# EECS 373

## Introduction to Embedded System Design

Robert Dick
University of Michigan

Lecture 9: Serial buses, Datasheets, ADCs, and DACs

15 February 2024

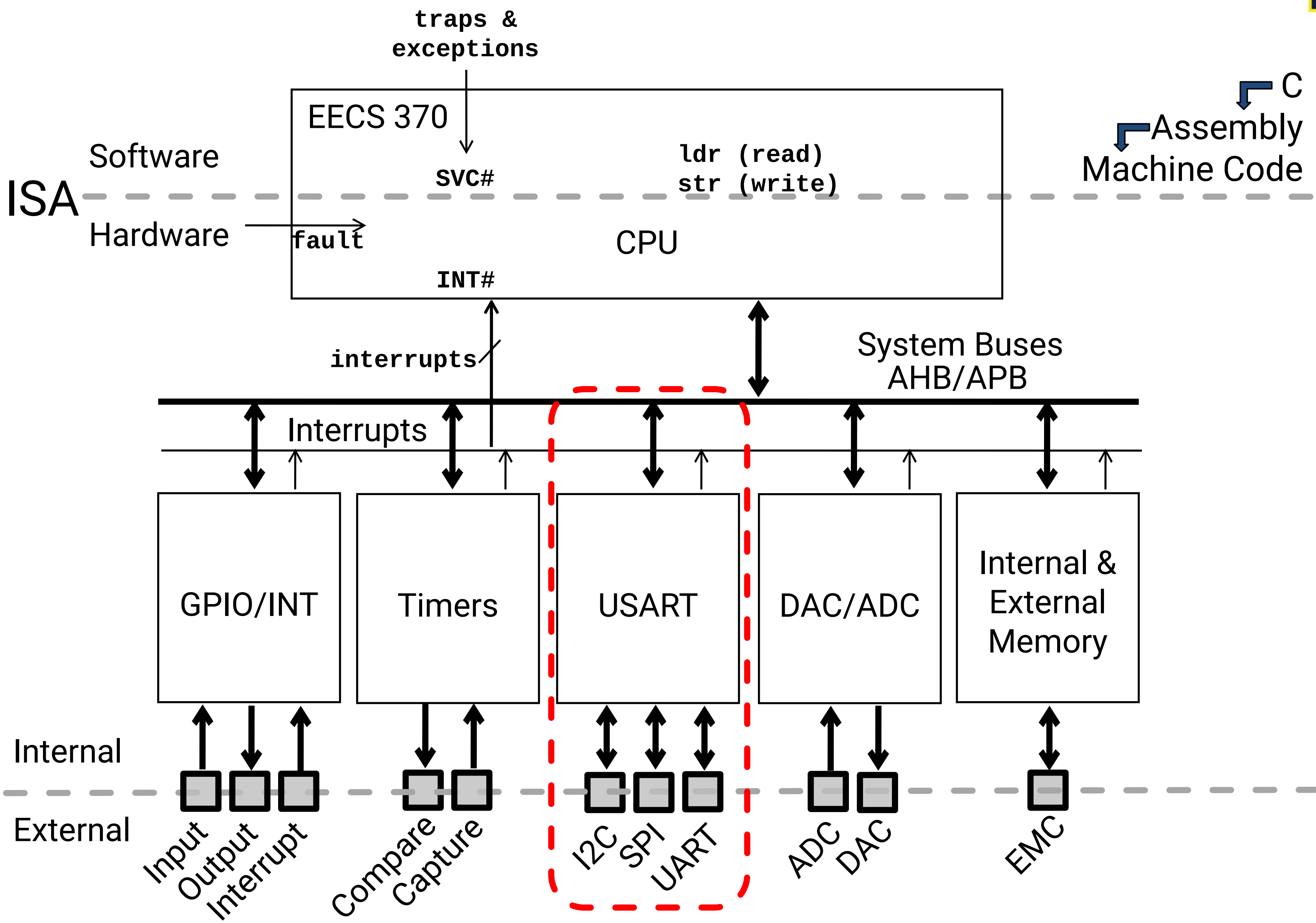
# Review

- Timers.
  - Compare: activate signal when counter = comparison register.
  - Capture: when signal received, copy counter to capture register.
- PWM.
- Hazards.
  - Definition.
  - Why they cause problems.
  - How to eliminate.
- Setup and hold times.
  - Definition.
  - Ways of honoring.

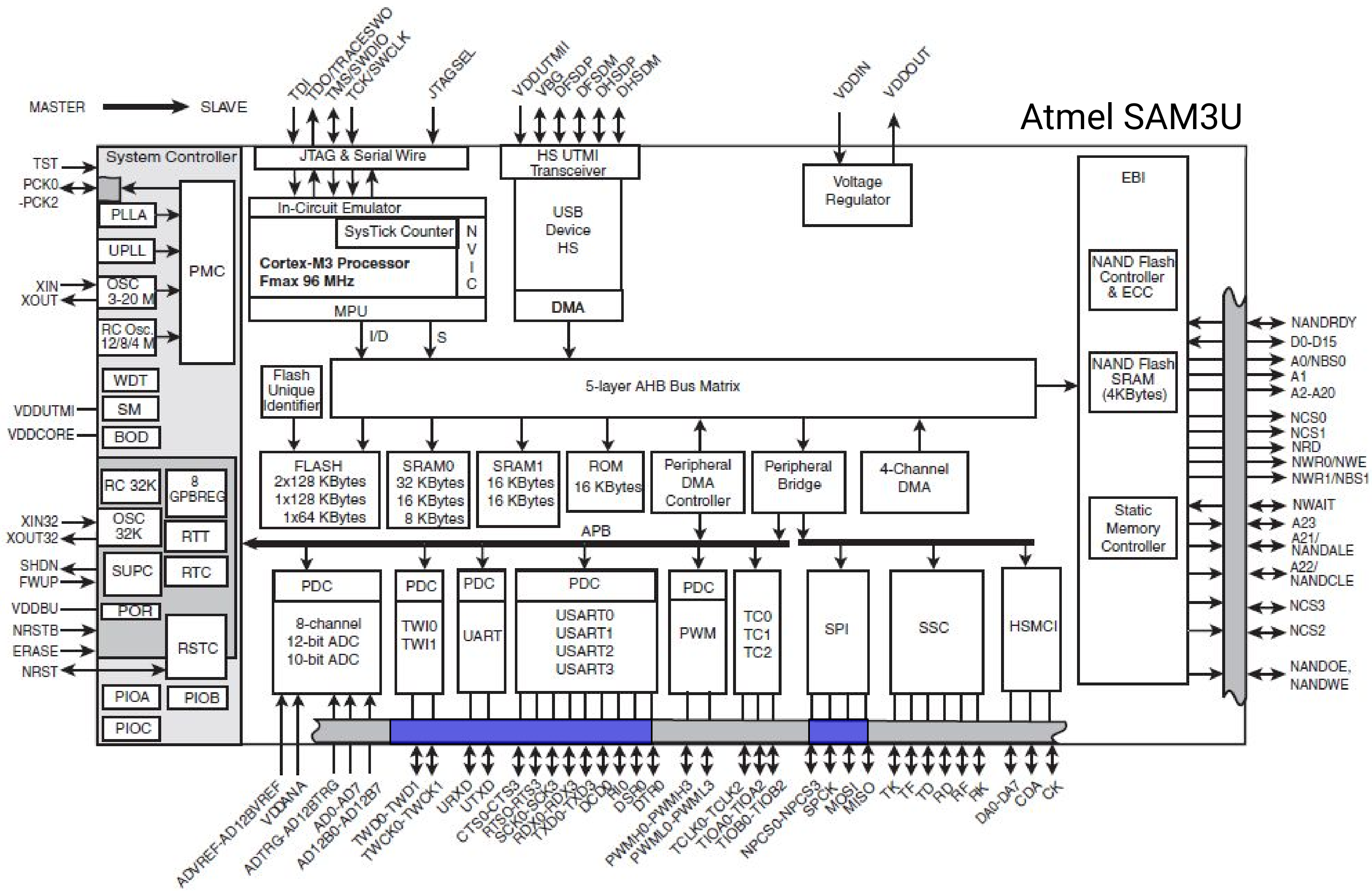
# Outline

- **Serial buses**
  - UART
  - I$^2$C
- Datasheets
- ADCs and DACs
- SPI (another serial bus)

# Serial interfaces

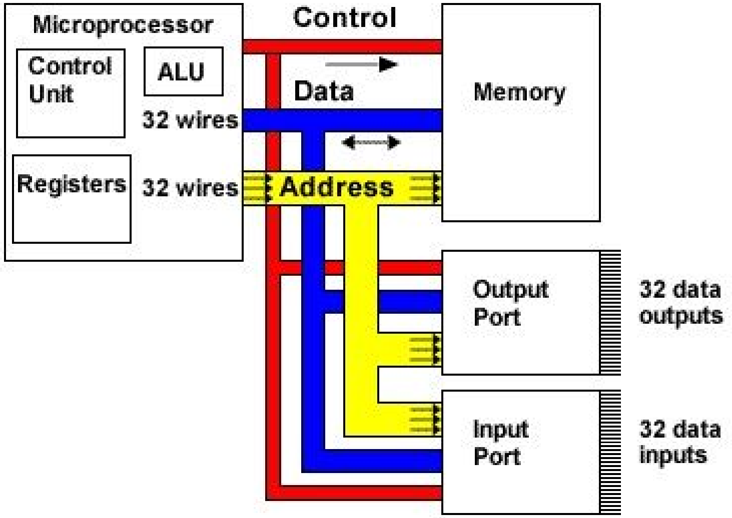


# External memory attaches to the processor via the external memory controller and bus



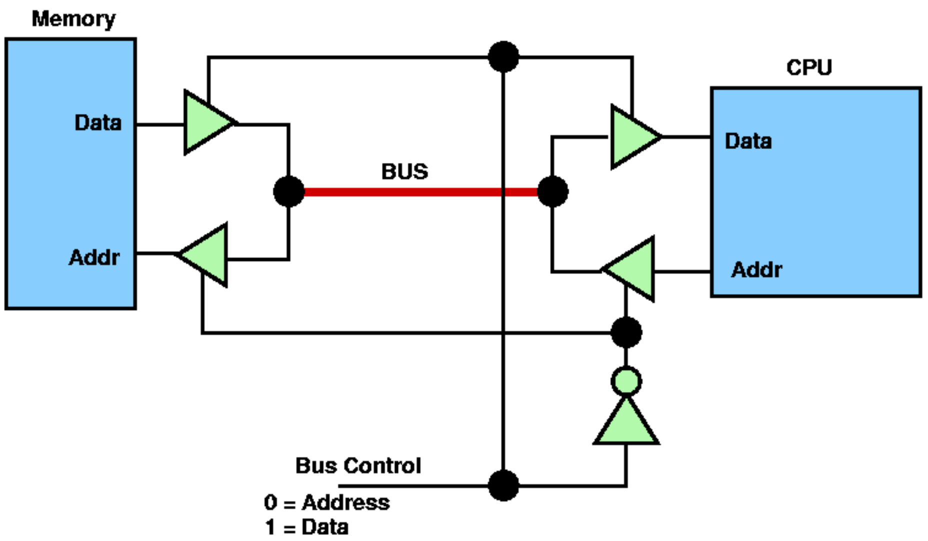
# Bus organization

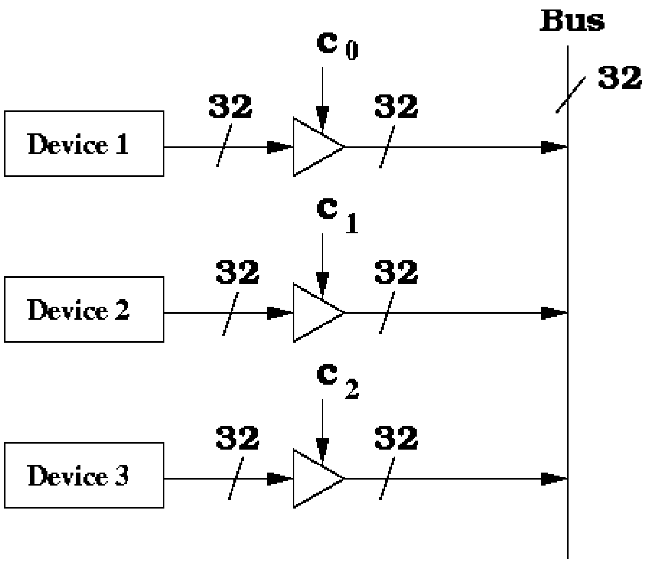
- Multidrop bus (MDB): components on same wires.
- More than one component can drive.
- Might be multi-initiator.



# Review: Multiple (potential) bus drivers (1)

- Tri-state devices: one on at a time.
- All can read.
- Pin-efficient, low-power.
- Potentially dangerous.



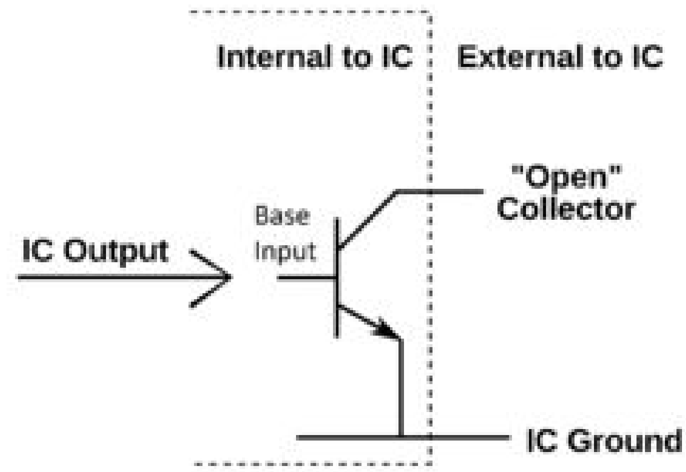



# Review: Multiple (potential) bus drivers (2)

- MUX
  - Many pins.
  - Consider a 32-bit bus with 6 potential drivers.
  - Generally impractical on PCB.
  - More practical on-chip.

# Review: Multiple (potential) bus drivers (3)

- "pull-up" aka "open collector" aka "wired OR"
- Pull high w. resistor.
- Any device can pull low.
- Safe.
- Fast or energy efficient, pick one.
- Used in $I^2C$, CAN.



# Outline

- Serial buses
  - **UART**
  - I²C
- Datasheets
- ADCs and DACs
- SPI

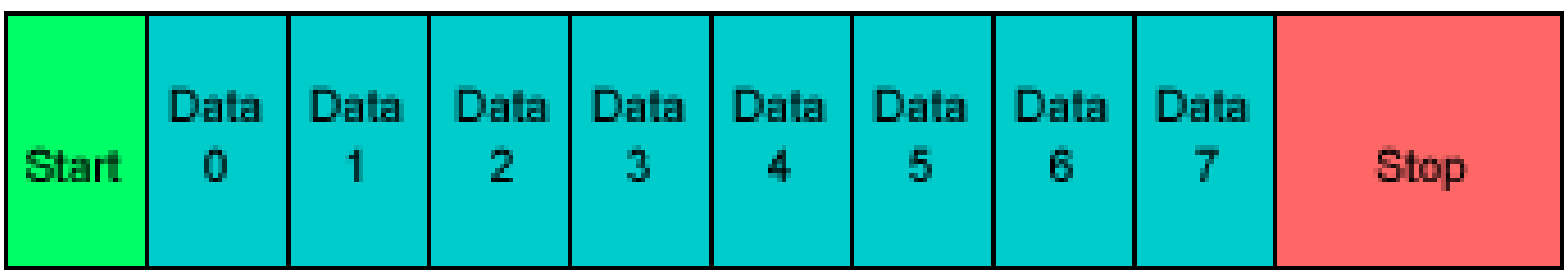
# UART

- Universal Asynchronous Receiver/Transmitter
- Translates data between parallel and serial forms.
- UARTs used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485.
- Universal
  - Configurable data format and speed.
  - Signaling levels/methods delegated.

# Protocol

- Each character is sent as
  - a logic *low* **start** bit
  - a configurable number of data bits (usually 7 or 8, sometimes 5)
  - an optional **parity** bit
  - *one or more logic high* **stop** bits.
  - with a particular bit timing ("**baud**" or "baudrate")

- Examples
  - "9600-N-8-1"  <baudrate><parity><databits><stopbits>
  - "9600-8-N-1"  <baudrate><databits><parity><stopbits>



# Variations

- U(S)ART is actually a generic term that includes a large number of different devices/standards.
- RS-232 is a standard.
- Specifies characteristics and timing of signals, the meaning of signals, and the physical size and pin out of connectors.

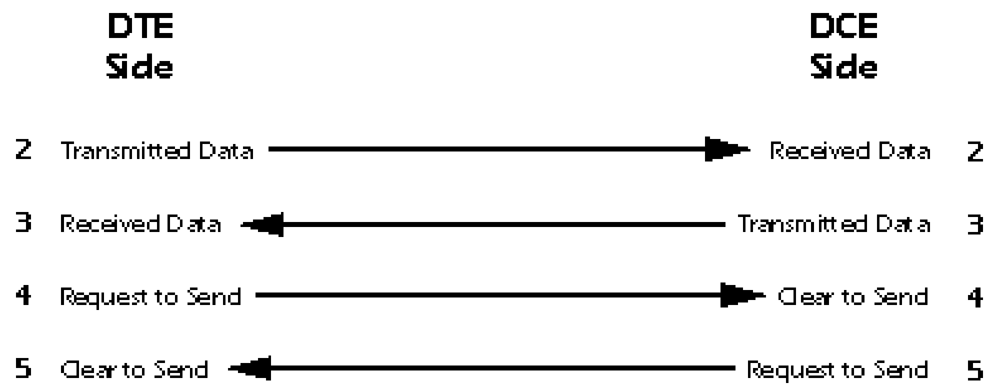
# Most commonly used signals

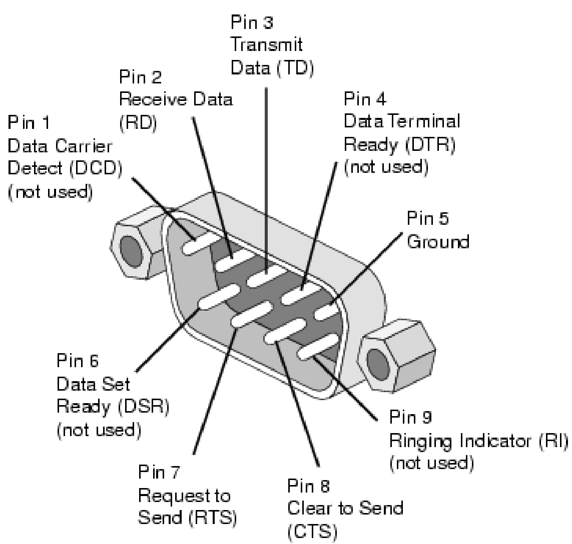
- Definitions
  - DTE: Data terminal equipment
  - DCE: Data circuit-terminating equipment
- **RXD**: for receiving data.
- **TXD**: for transmitting data
- Flow control.
  - **RTS#** (Request to Send): DTE calls for DCE to send data..
  - **CTS#** (Clear to Send): DCE tells DTE it is ready to accept data..

- RXD ↔️ TXD
- RTS ↔️ CTS

# DB9 stuff

- DTE vs. DCE.
- Pinout of a DCE?
- Common ground?
- Noise effects?

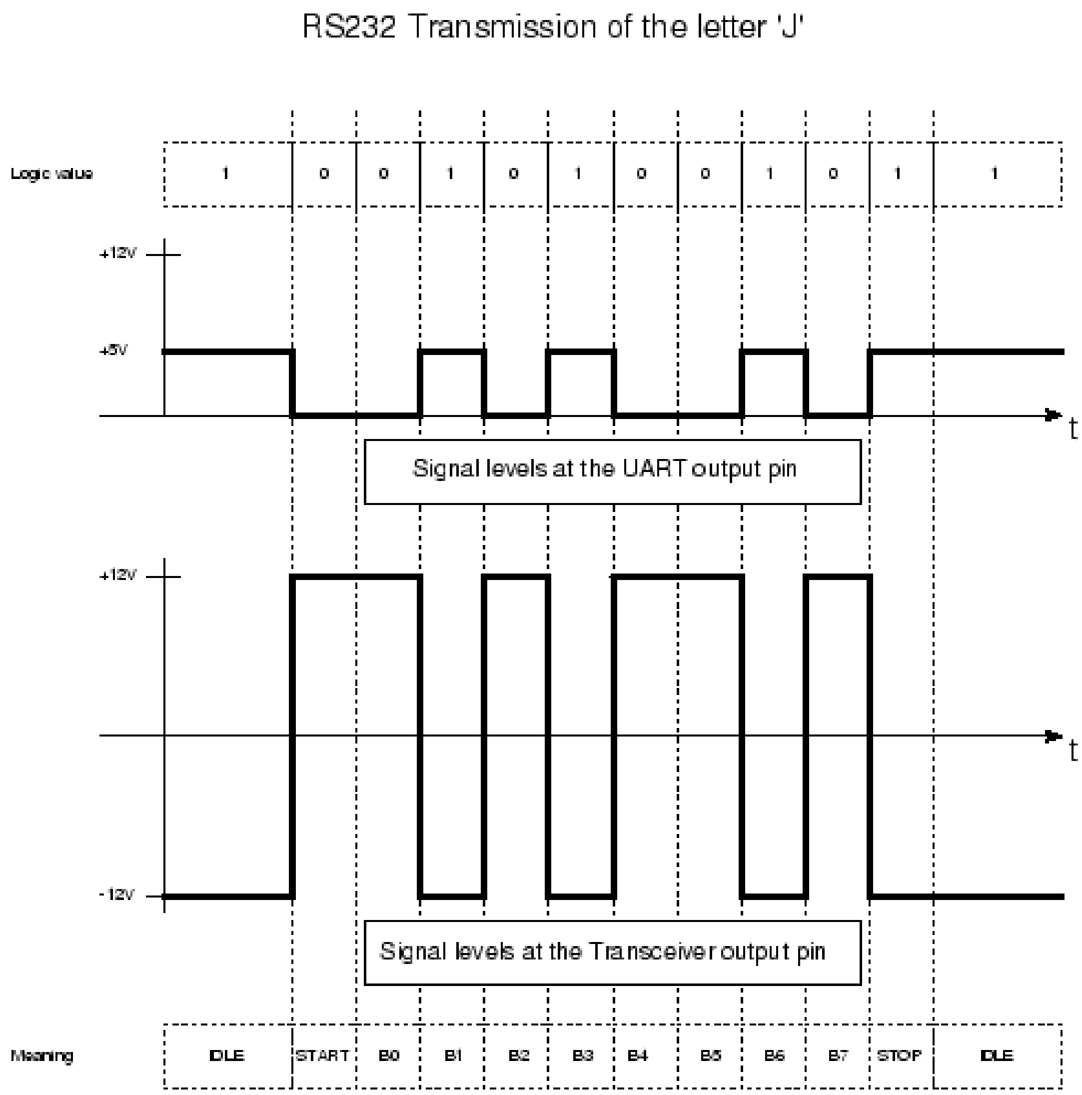



| Pin Number | Signal | Description |
|---|---|---|
| 1 | DCD | Data carrier detect |
| 2 | RxD | Receive Data |
| 3 | TxD | Transmit Data |
| 4 | DTR | Data terminal ready |
| 5 | GND | Signal ground |
| 6 | DSR | Data set ready |
| 7 | RTS | Ready to send |
| 8 | CTS | Clear to send |
| 9 | RI | Ring Indicator |
| | | |

DTE ↔️ DCE: wires connect pin x ↔️ x.
DTE ↔️ DTE: crossover or null modem cable needed.

# RS-232 transmission example



# Outline

- Serial buses
  - UART
  - **I²C**
- Datasheets
- ADCs and DACs
- SPI

## I$^2$C summary

- Inter integrated circuit bus.
- Two-wire protocol.
- From Philips in early 1980s.

# I$^2$C applications

- Initially in TV sets.
- Common for peripherals from many companies.
- Real-time clocks.
- Temperature sensors.
- Many others.

# I²C technical description

- Two-wire serial protocol.
- Addressing.
- Up to 3.4 Mb/s.
- Multi-initiator, multi-target.

# I$^2$C wiring

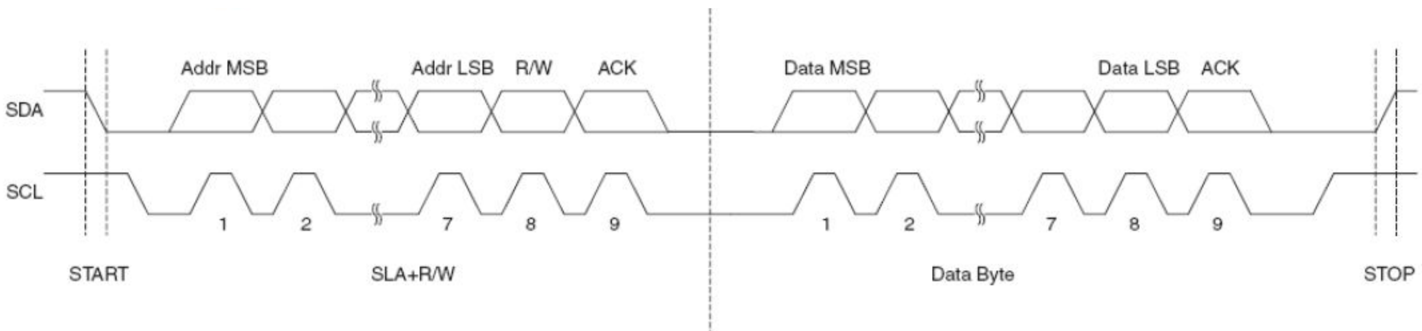
- SDA: data.
- SCL: clock.
- Open collector.
- Simple interfacing among voltage domains.

# $I^2C$ clocking

- Unconventional.
- Quiescent state is high.
- Initiator pulses low during transmission.
- Target holds clock low to extend transmission cycle.
  - Open-collector design enables this.

# I²C transaction

- Start.
- Address.
- Data.
- Ack.
- Stop.



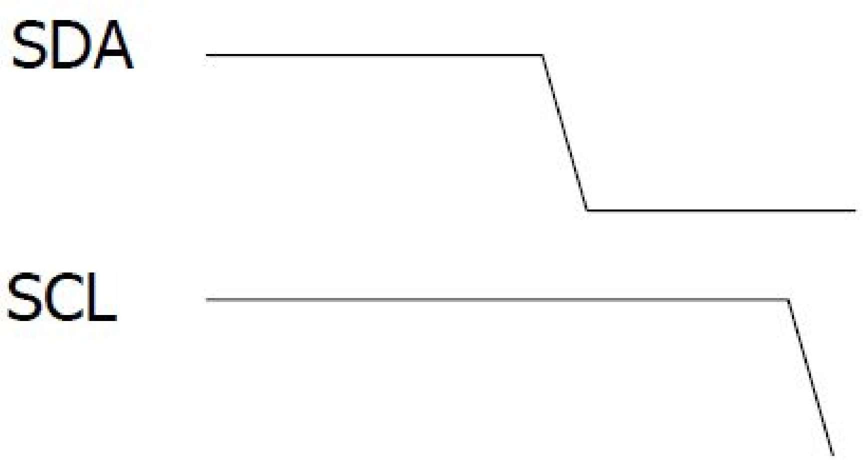

Source: ATMega8 Handbook

# I$^2$C roles

- Transmitter/receiver not same as initiator/target.
- Initiator starts transaction.
- Transmitter sends data on SDA.
- Receiver acks.
- Read: target is transmitter.
- Write: initiator is transmitter.

# I²C starting

- Initiator drives SDA low while SCL remains high.
- During other parts of transactions, SDA changes when SCL is low.

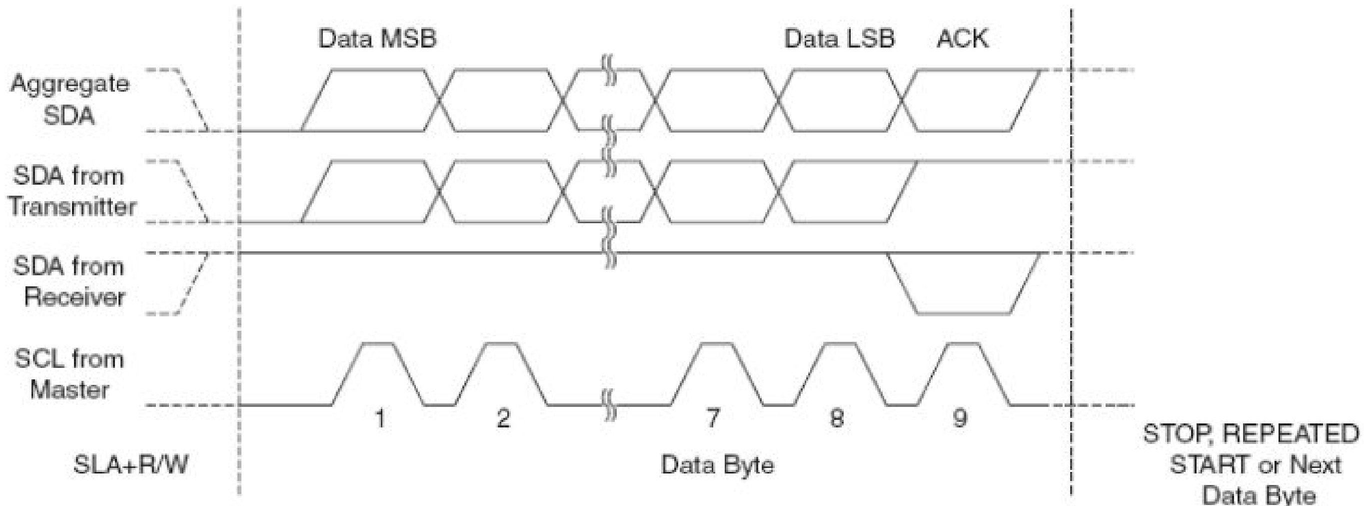


# I$^2$C address

- Sampled on rising SCL.
- 7-bit address.
- 8th bit
  - Low: write.
  - High: read.
- Philips/NXP can assign standard addresses for a fee.
- Devices with hard-coded addresses troublesome.
  - What if >1 device needed?
  - Segment bus?
  - Add select line outside bus protocol?
  - Add custom select/MUX circuitry?

# $I^2C$ data

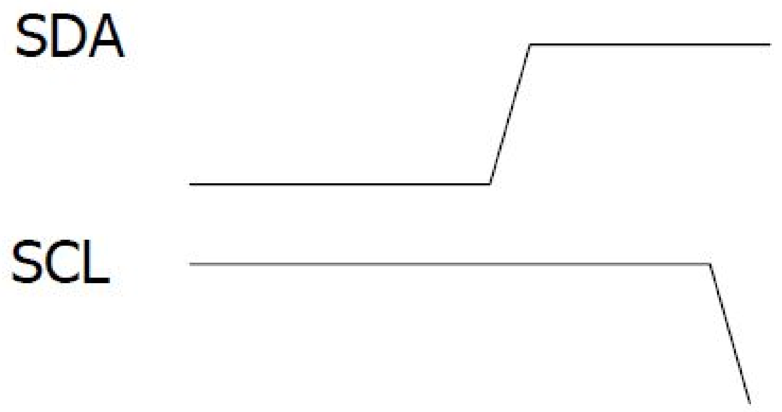
- Sampled on rising SCL.
- 8-bit.
- Write: initiator transmits, target acks.
- Read: target transmits, initiator acks.
- Continues until initiator signals to stop.



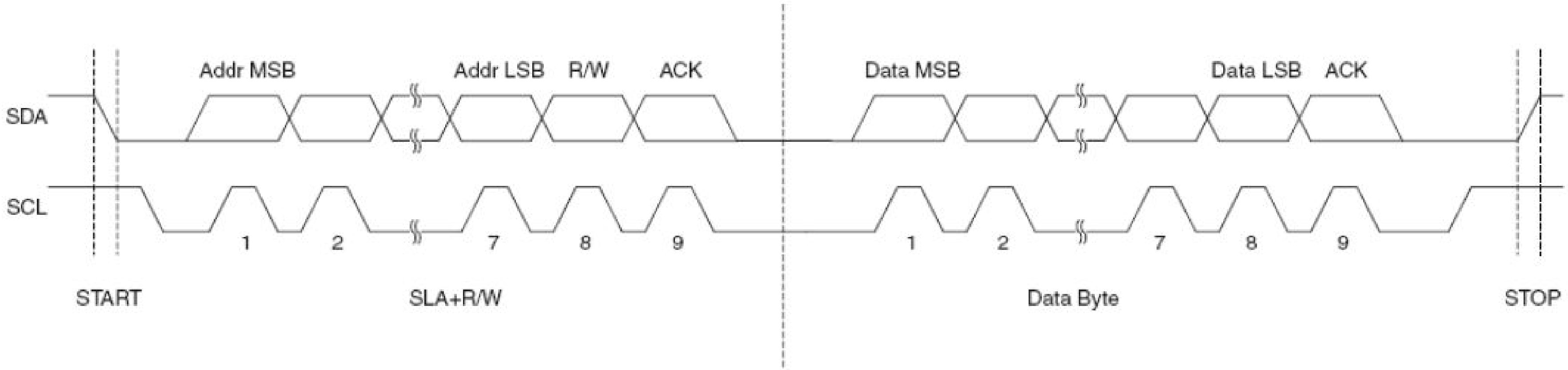

Source: ATMega8 Handbook

# I$^2$C stopping

- Initiator allows SDA to go high while SCL high.
- Stops or aborts transactions.



# I$^2$C timing diagram




Source: ATMega8 Handbook

# Outline

- Serial buses
  - UART
  - $I^2C$
- **Datasheets**
- ADCs and DACs
- SPI

# Datasheets

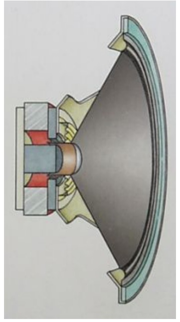
# Outline

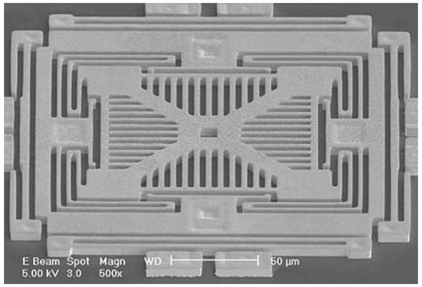
- Serial buses
  - UART
  - I2C
- Datasheets
- **ADCs and DACs**
- SPI

# Many signals effectively analog

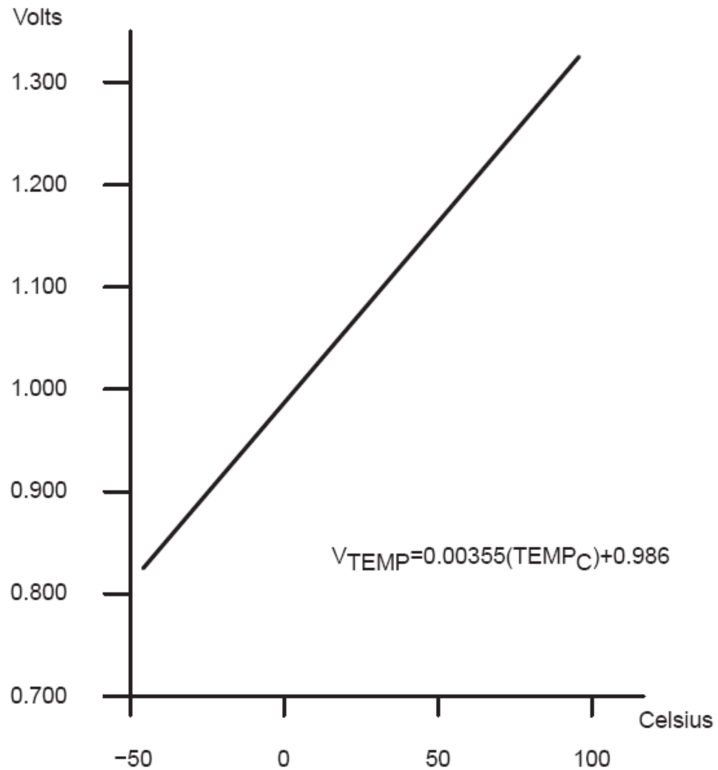
- Many signals analog.
- Sound, light, temperature, pressure, voltage, etc.

- Path to digital system.
  - Source → continuous voltage → discrete value.
- Transducers: converts one type of energy to another
  - Electro-mechanical, optical, electrical, …
- Examples.
  - Microphone/speaker.
  - Thermocouples.
  - Accelerometers.





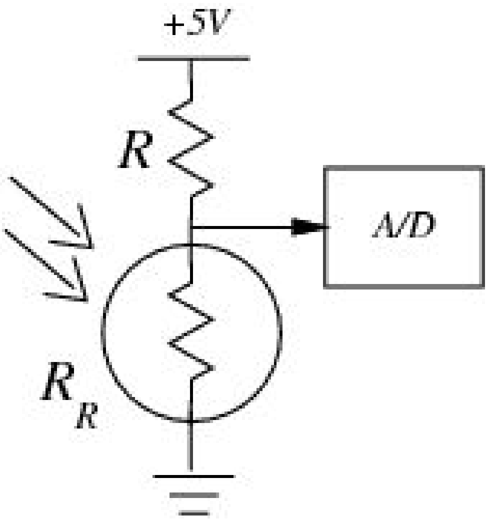
# Transducers convert one form of energy into another



# Convert light to voltage with a CdS photocell

- $V_{signal} = (+5V)\ R_R/(R + R_R)$.

- Choose $R = R_R$ at median of range.
- Cadmium Sulfide (CdS).
- Cheap, low current.
- $T_{RC} = (R+R_R)\ C_l$.
- Typically $R \approx 50\text{-}200\ k\ \Omega$.
- $C \approx 20$ pF.
- So, $T_{RC} \approx 20\text{-}80$ uS.
- $f_{RC} \approx 10\text{-}50$ kHz.




Source: Forrest Brewer

# Many other common sensors (some digital, 1/2)

- Force.
  - Strain gauges - foil, conductive ink.
  - Conductive rubber.
  - Rheostatic fluids.
    - Piezorestive.
  - Piezoelectric films.
  - Capacitive force.
- Sound.
  - Microphones: current or charge.
  - Sonar: usually piezoelectric.
- Position.
  - Switches.
  - Shaft encoders.
  - Gyros.
- Atmospheric pressure.

# Many other common sensors (some digital, 2/2)

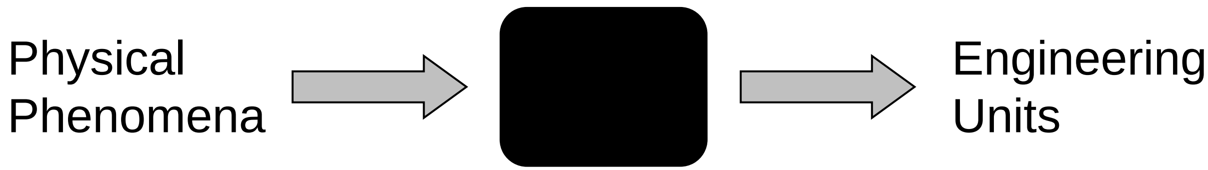
- Acceleration
  - MEMS
  - Pendulum
- Monitoring
  - Battery energy
  - Motor velocity
  - Temperature
- Field
  - Antenna
  - Magnetic
    - Hall effect
    - Flux gate
- Location
  - Permittivity
  - Dielectric
- Conductivity
- Many, many more

# Sensor strengths and weaknesses

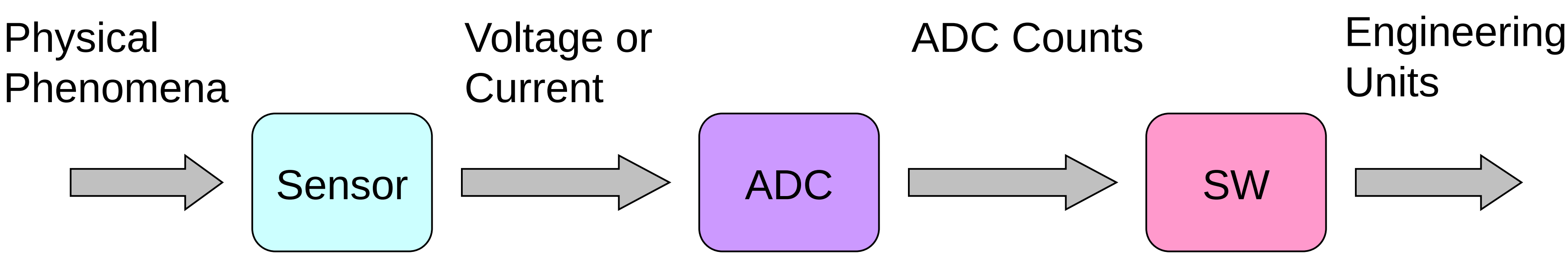
- Common for different sensors to provide information relevant to query of interest.
- Sensors generally perform well under some circumstances and poorly under others.
- E.g., cameras work well in daylight but not at night or in fog, sonar has different characteristics.
- Sensor fusion can cover more varied circumstances.
- Fusion of sensors that are all correlated to the query of interest but not correlated with each other can improve accuracy a lot.

# Analog to digital

- Goal

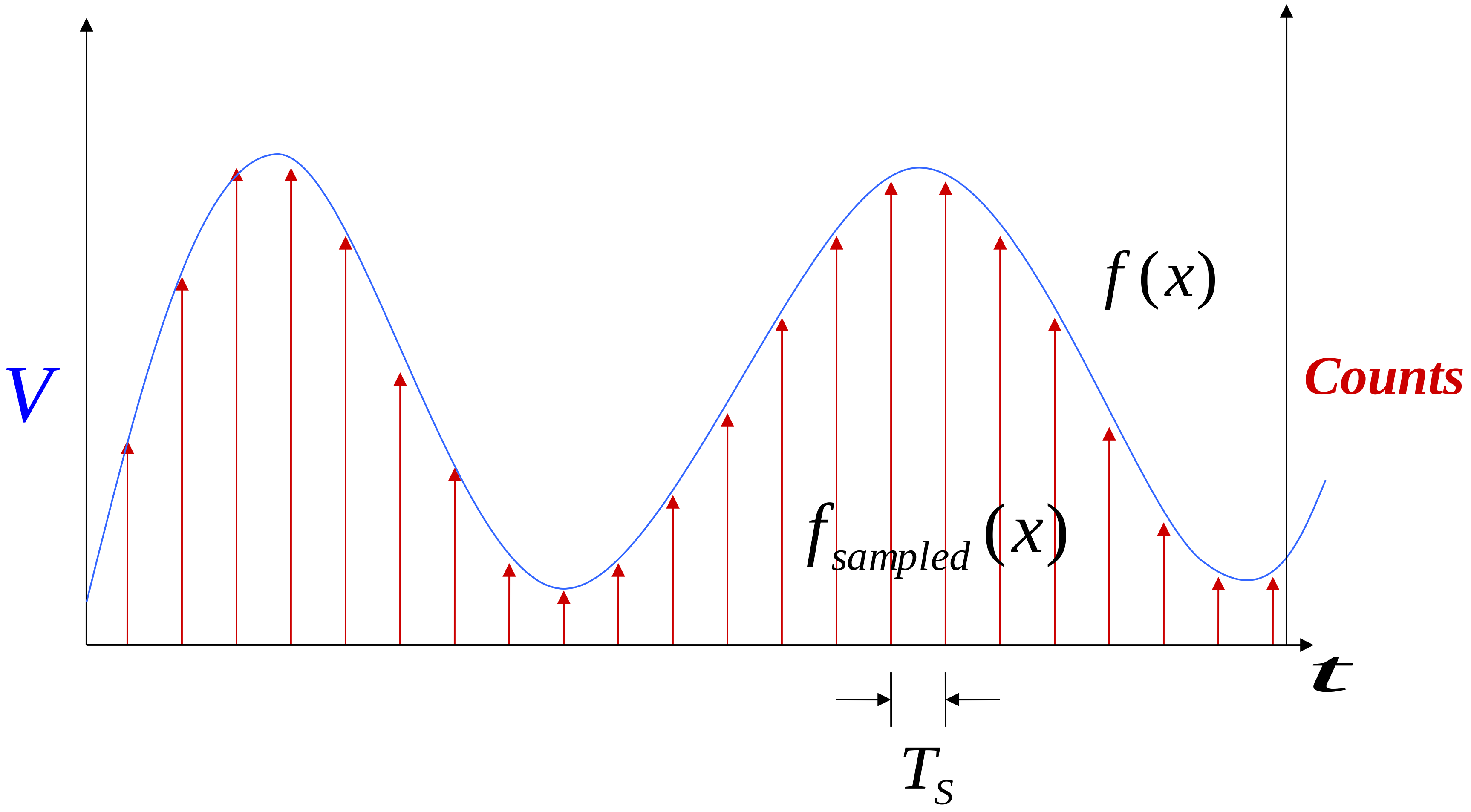



- Process



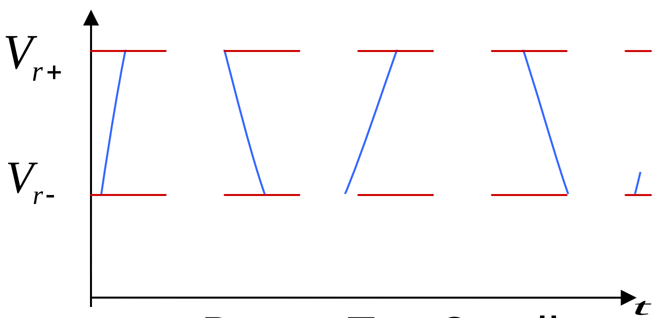
# Digital representation of analog signal

- Discretize in time and value.

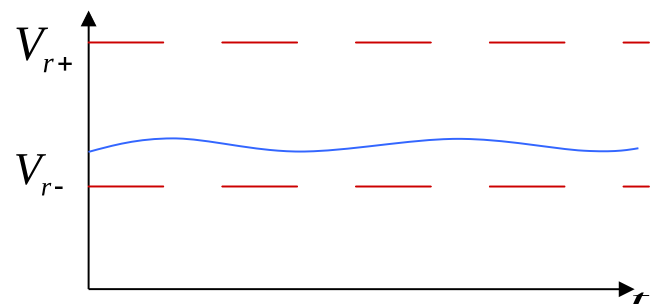


# Choosing the value range

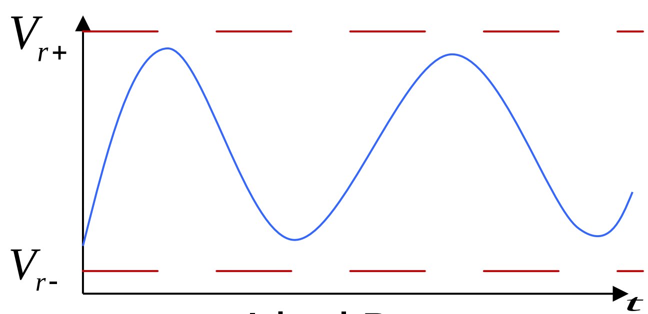
- What do the sample values represent?
- Some fraction within the range of values




Range Too Small



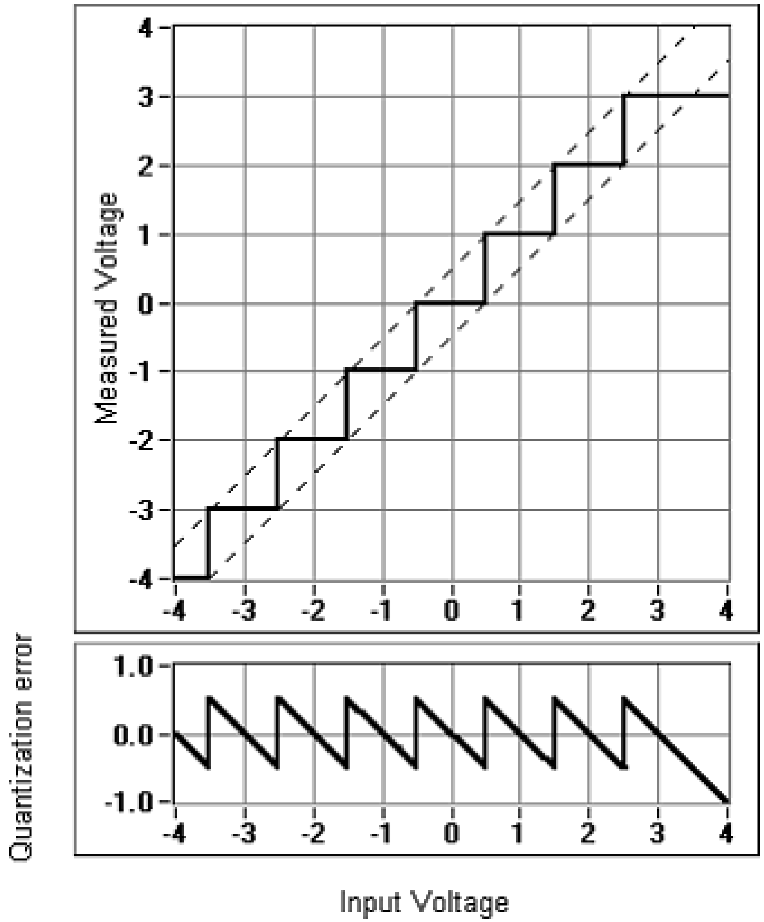

Range Too Big




Ideal Range

# Choosing the value resolution

- Resolution
  - Number of discrete values that represent a range of analog values.
  - MSP430: 12-bit ADC
    - 4096 values
    - Range / 4096 = Step

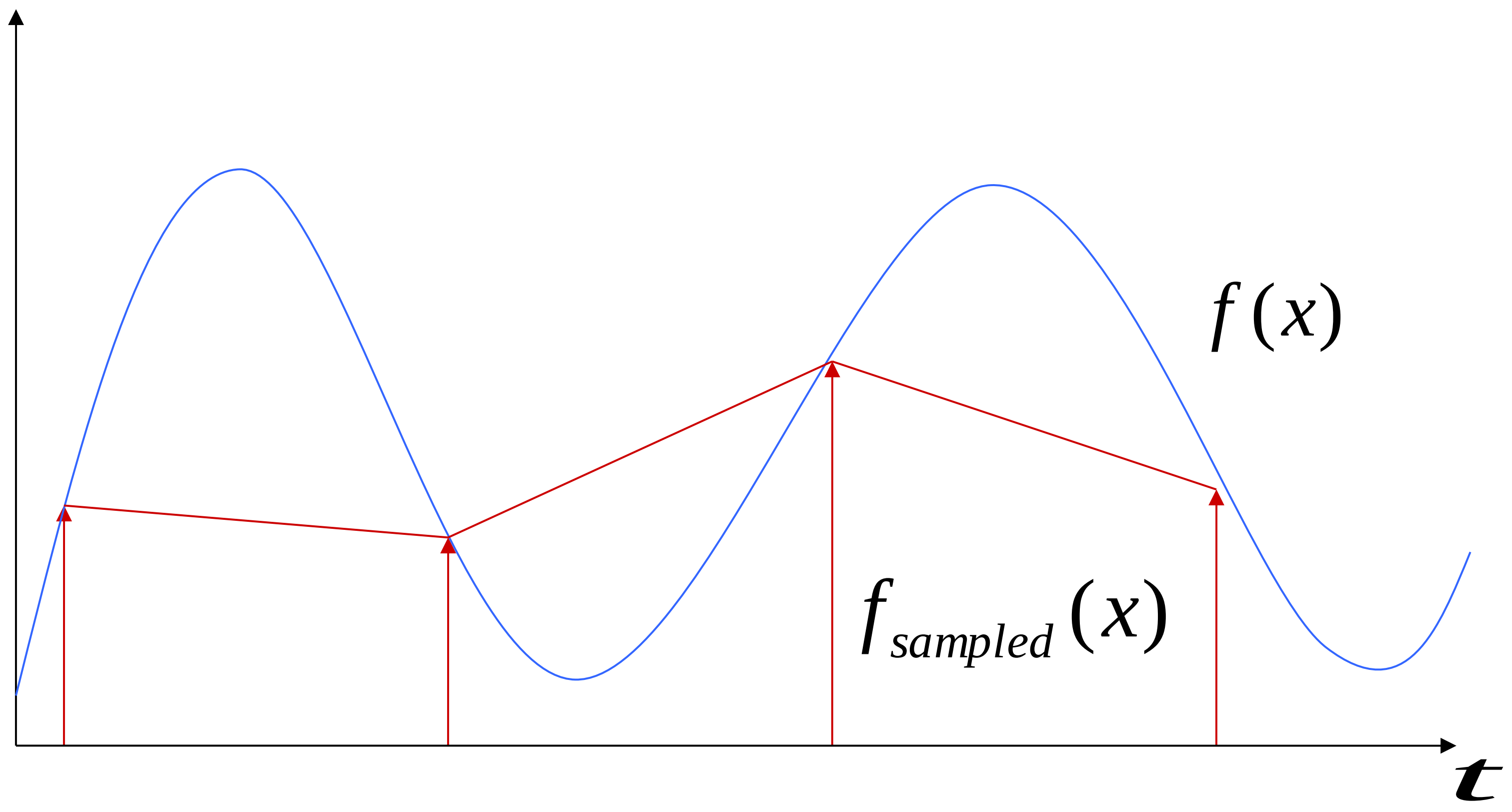



- Quantization Error
  - How far off discrete value is from actua
  - ½ LSB → Range / 8192

**Larger range → lower resolution**

# Choosing the temporal resolution

- Too low: we can't reconstruct the signal.
- Too high: waste computation, energy, resources.



# Shannon-Nyquist sampling theorem

- If a continuous-time signal $f(x)$ contains no frequencies higher than $f_{\max}$, it can be completely determined by discrete samples taken at a rate:

$$f_{\text{samples}} > 2\, f_{\max}$$

- Example:
  - Humans can process audio signals 20 Hz – 20 kHz.
  - Audio CDs: sampled at 44.1 Khz.

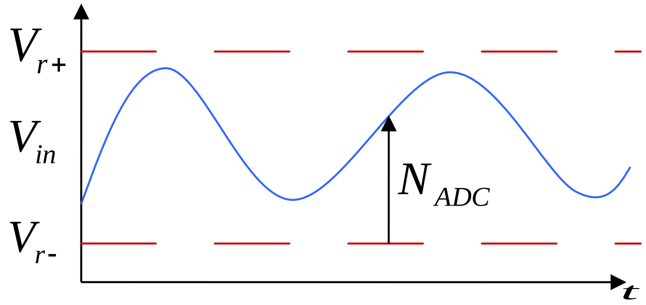
- Caveat: additional samples can have value if signal contains high-frequency noise.
  - Allows low-pass filtering.
  - Improves accuracy.
  - Decreases effective samping rate.
  - Can sometimes fix with filter at measurement side.

# Compressed sensing

- Can sometimes reconstruct well at below Nyquist rate.
- Relies on understanding (predictable) properties of signal.

# Converting between voltages, ADC counts, and engineering units

- Converting: ADC counts → Voltage




$$N_{ADC} = 4095 \times \frac{V_{in} - V_{r-}}{V_{r+} - V_{r-}}$$

$$V_{in} = N_{ADC} \times \frac{V_{r+} - V_{r-}}{4095}$$

- Converting: Voltage → Engineering Units

$$V_{\mathrm{TEMP}} = 0.00355(\mathrm{TEMP_C}) + 0.986$$

$$\mathrm{TEMP_C} = \frac{V_{\mathrm{TEMP}} - 0.986}{0.00355}$$

# A note about sampling and arithmetic

- Common error converting values

$$V_{TEMP} = N_{ADC} \times \frac{V_{r+} - V_{r-}}{4095} \qquad \text{TEMP}_C = \frac{V_{TEMP} - 0.986}{0.00355}$$

```
volatile const int adccount;
float vtemp = adccount / 4095 * 1.5;
float tempc = (vtemp-0.986) / 0.00355;
```

- Learn associativity and type conversion rules of ANSI C.
- Example.
- Fixed point operations
  - Overflow and underflow dangers complicate design.
  - Can use logarithmic representation for dynamic range.
  - This is deep. See reference. Talk to me.
- Floating point operations
  - Often software emulated.
  - Often slow, power-hungry on embedded processors.

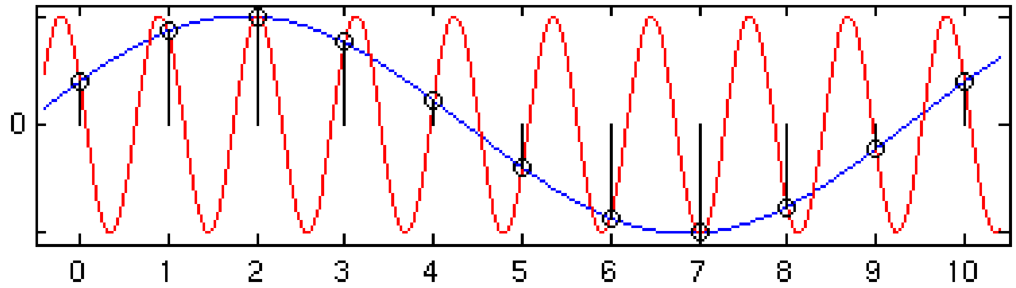
# Floating point

See W. Kahan and J. D. Darcy, “How Java’s Floating-Point Hurts Everyone Everywhere,” Wkshp. on Java for High-Performance Network Computing, Mar. 1998.

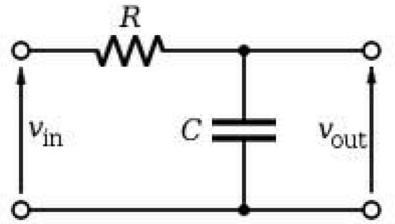
- In many ANSI C implementations, floats are auto-promoted to double by virtue of floating point unit register width.
- Cuts back down in case of [float] * [float].
- What is this: 5.5?
- Bottom line: “double” makes many numerically questionable algorithms work right in practice.
- Unless it’s slow (e.g., floating point emulation), use double except when size matters, e.g., big matrices or arrays.

# Use anti-aliasing filters on ADC inputs to ensure that Shannon-Nyquist is satisfied

- Aliasing.
  - Different frequencies are indistinguishable when they are sampled.



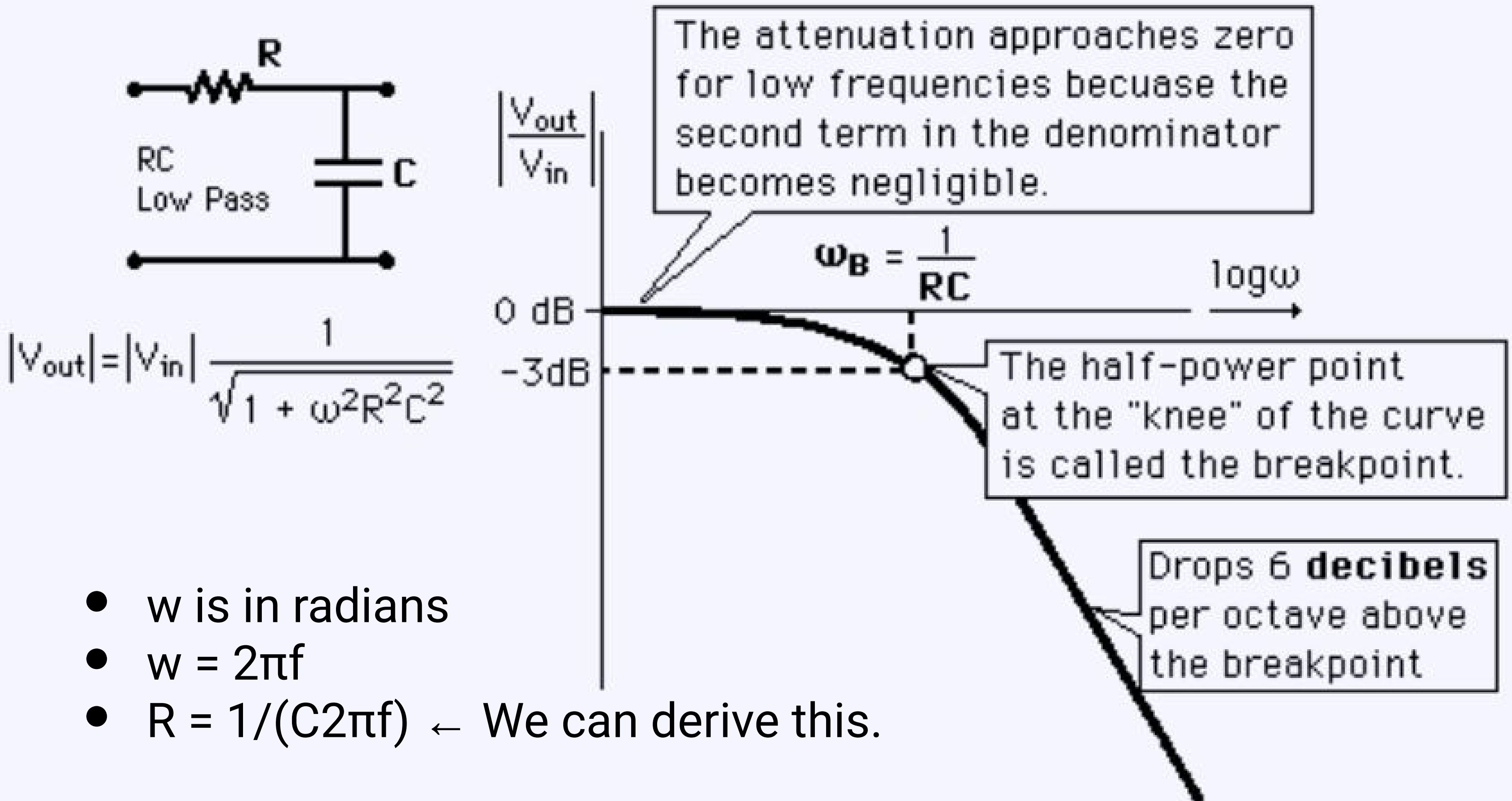

- Condition the input signal using a low-pass filter.
  - Removes high-frequency components.
  - A.K.A. anti-aliasing filter.



# Do I really need to condition my input signal?

- Often.
- Many ADCs have analog filter built in.
- Those filters typically have a cut-off frequency just above ½ their ***maximum*** sampling rate.
- Which is great if you are using the maximum sampling rate, less useful if you are sampling at a slower rate.
- Can sometimes use random sampling phase offsets to avoid some aliasing problems.

# Designing the anti-aliasing filter




- w is in radians
- w = 2πf
- R = 1/(C2πf) ← We can derive this.

- Goal: cutoff f = 30 Hz. Given: C = 0.1 μF.
- Question: R = ?
- Example.

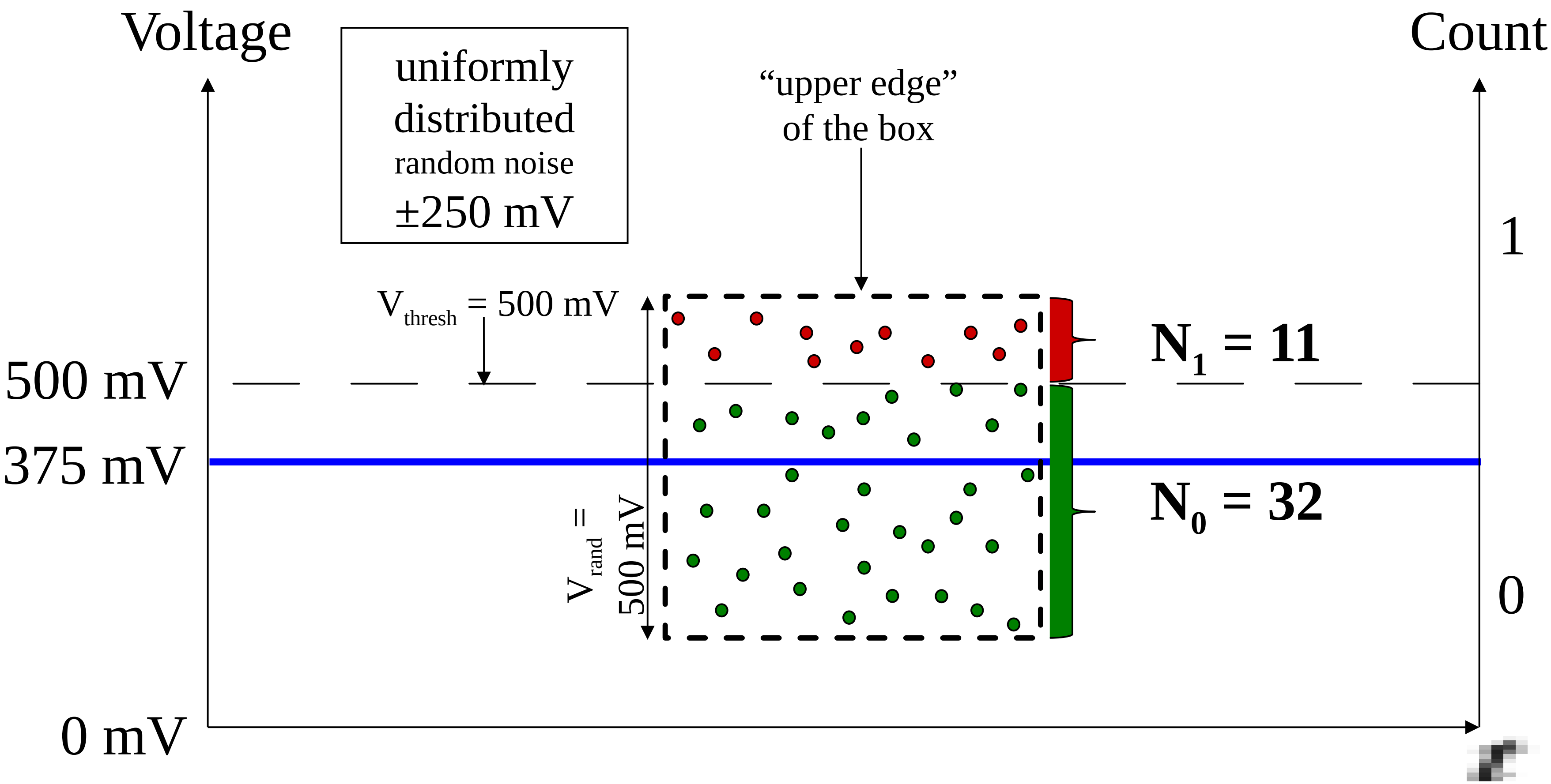
# Quantization error

- Illustrate on paper.

- How to solve?
- Dither: add or or leave noise and oversample.
- Independent.
- High-frequency.
- Reduces quantization error.

- Trades off temporal for value resolution.

# Oversampling a 1-bit ADC w/ noise & dithering (cont)




**Note:**

$N_1$ is the # of ADC counts that = 1 over the sampling window

$N_0$ is the # of ADC counts that = 0 over the sampling window

See example.

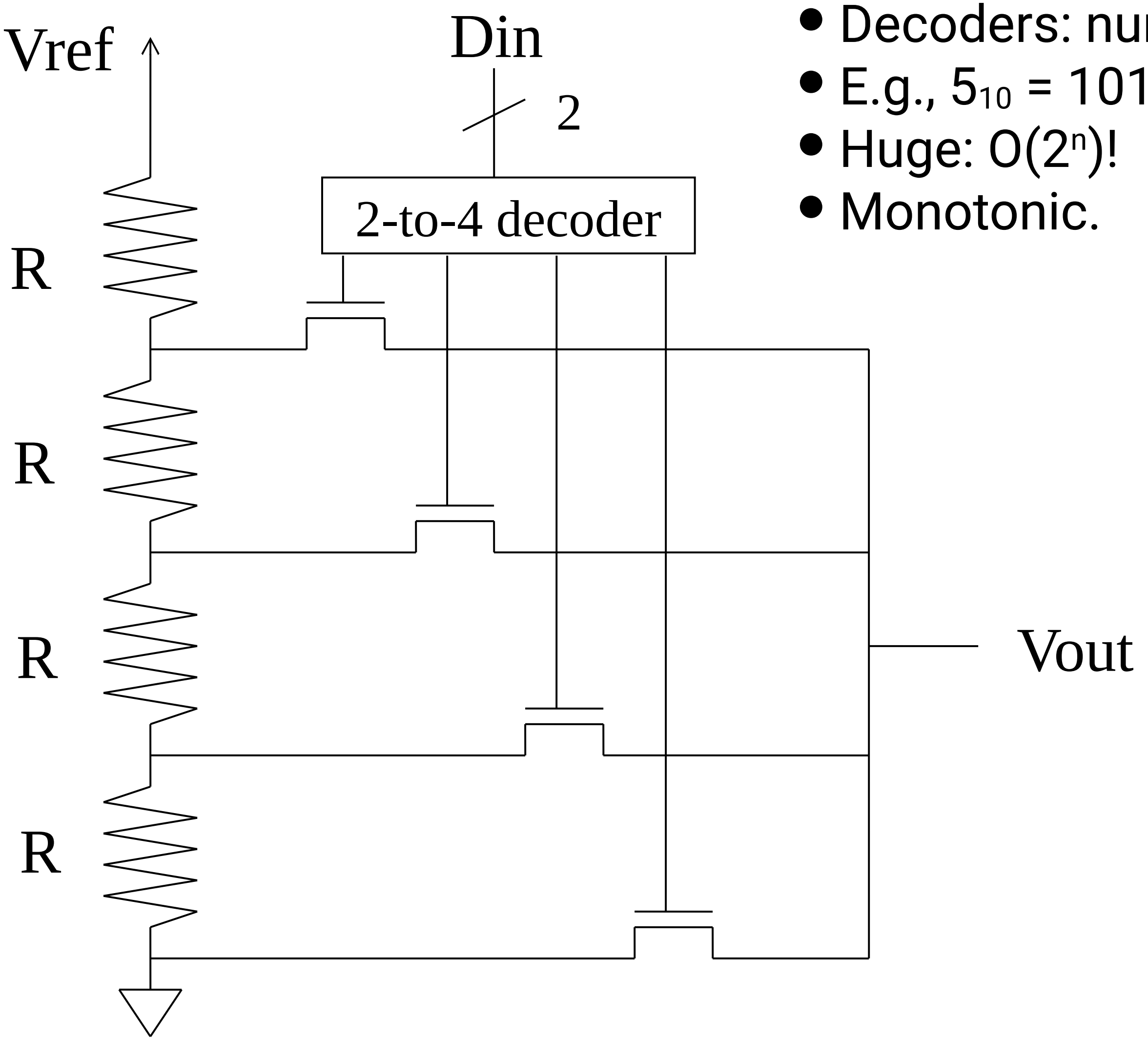
# Other potential problems

- Might need anti-imaging filter.
  - Especially during analog signal reconstruction.
  - Remove high-f components from stair stepping.

- Cost and power play a role.

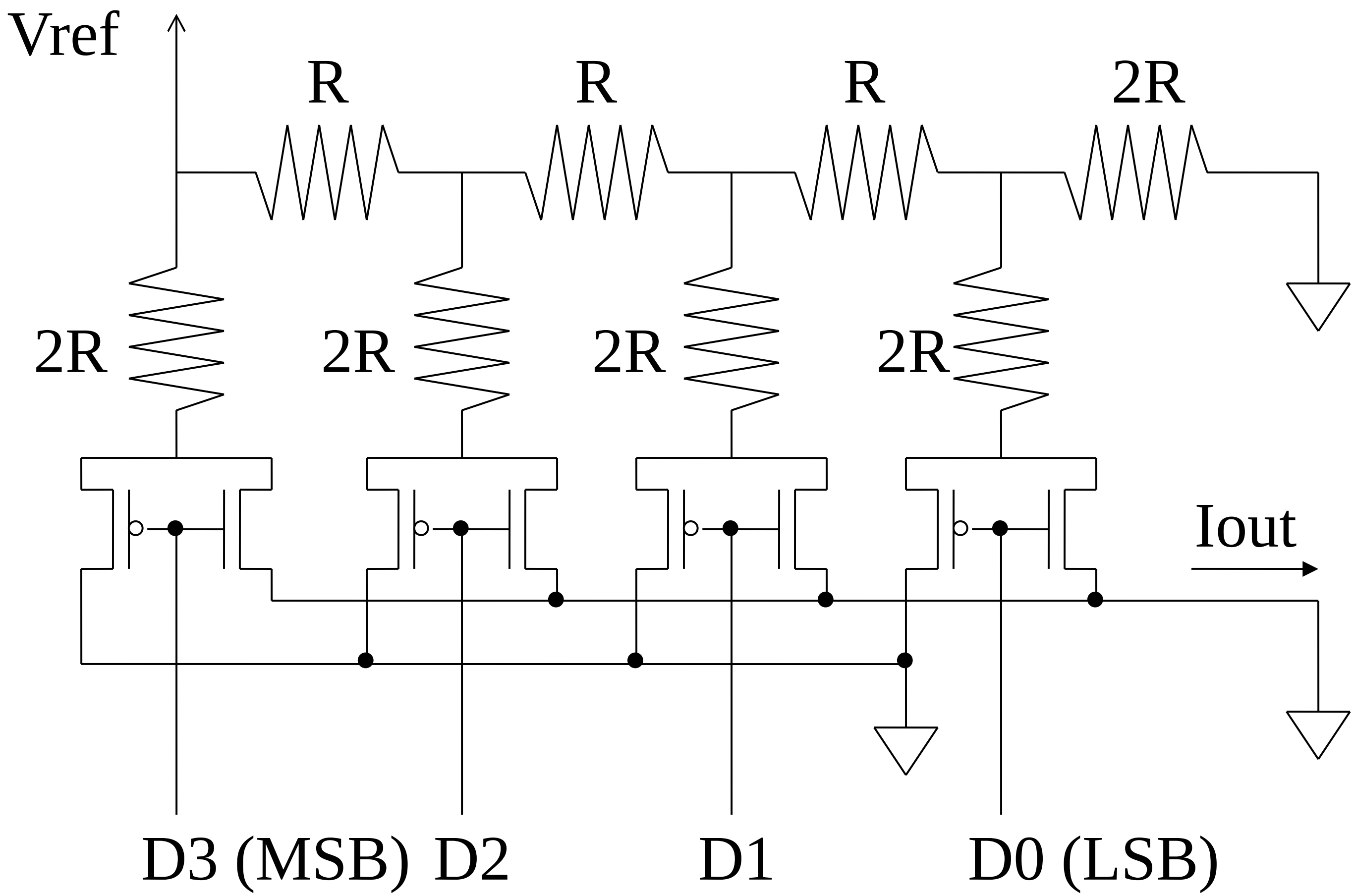
- Might be able to avoid ADCs/DACs.
  - E.g., PWM.

# DAC #1: voltage divider



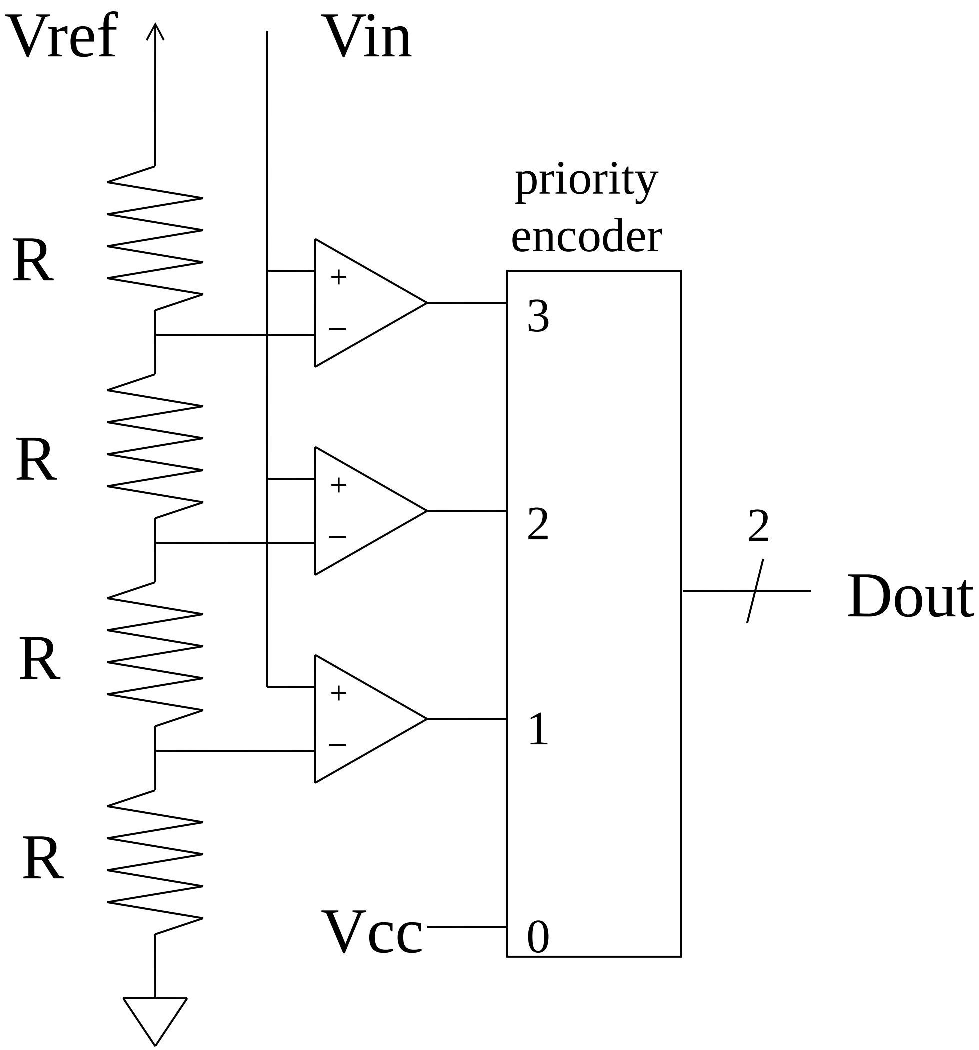

- Fast.
- Decoders: number → single on-bit.
- E.g., $5_{10} = 101_2 \rightarrow 00010000$.
- Huge: $O(2^n)$!
- Monotonic.

# DAC #2: R/2R ladder



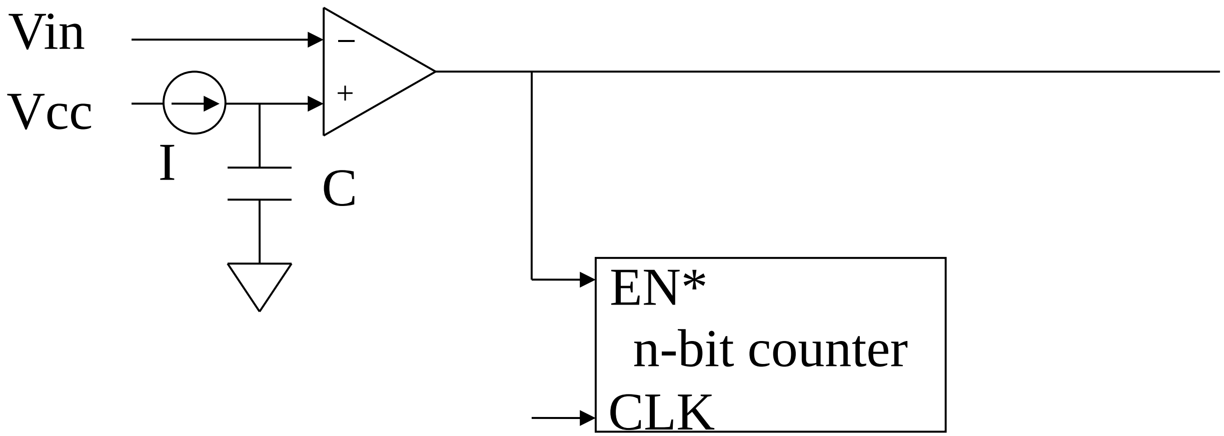

- Small: O(n).
- Monotonicity? (Consider 0111 -> 1000)

# ADC #1: flash



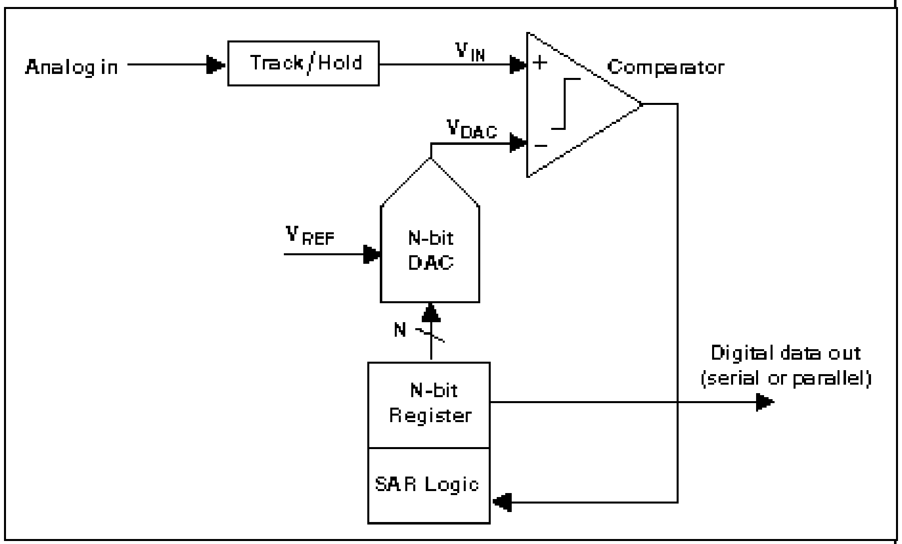

- Compare input with linear range of voltages.
- Use priority encoder.
- Only leave on highest input bit.
  - 001111111 → 001000000.
- Convert input bit index to binary number.
- 001000000 → 110
  - Noting that 000000000 → 000.
- Huge: $O(2^n)$.

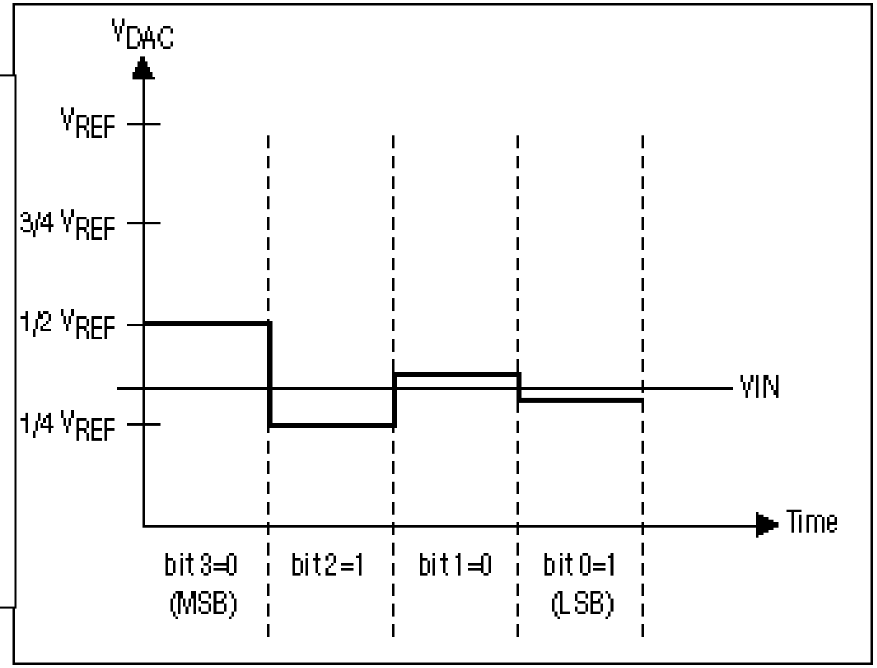
# ADC #2:  single-slope integration




- Start: Reset counter, discharge C.
- Charge C at fixed current I until Vc > Vin.
- Final counter value is Dout.
- Slow: $O(2^n)$.
  - Conversion may take several milliseconds.
- Good differential linearity (dI/dO).
- Absolute linearity depends on precision of C, I, and clock.

# ADC #3: successive approximation



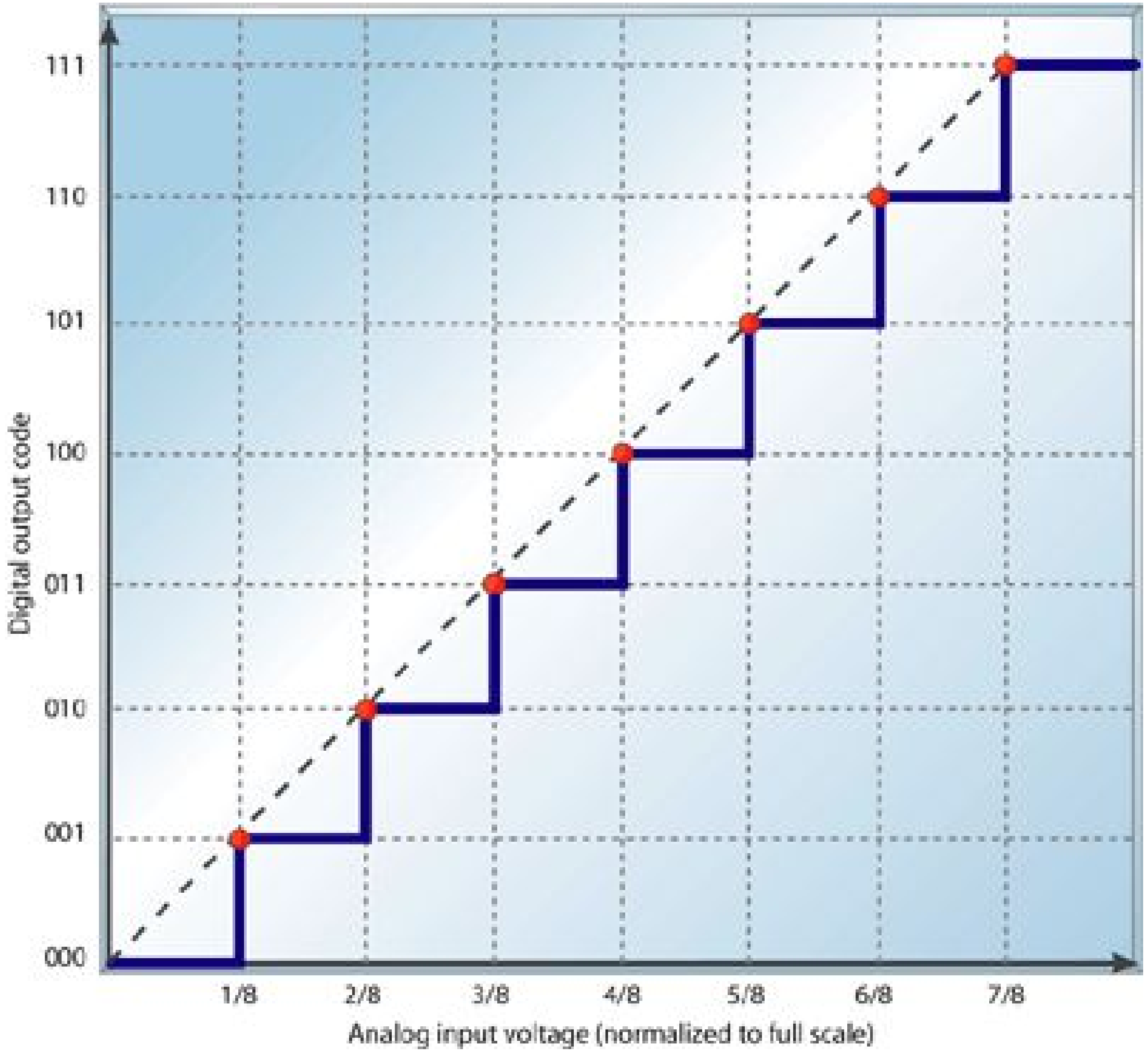

1 Sample → Multiple cycles

- Uses DAC for guessing.
- Somewhat fast: O(n).
- Goes from MSB to LSB.

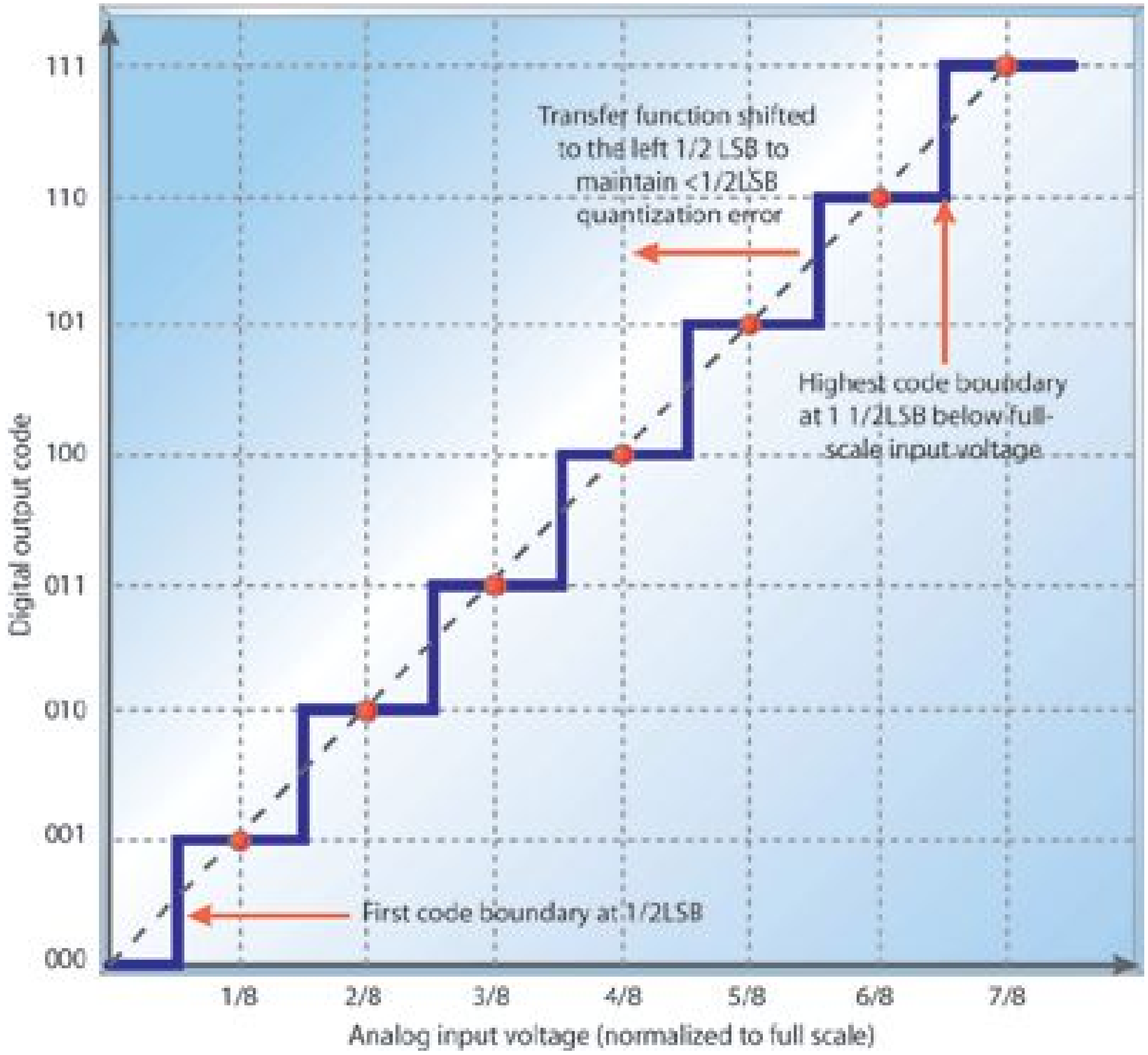
# Errors and ADCs

- Figures and some text from “Understanding Analog to Digital Converter Specifications,” Staller, Len, EE Times Asia.
- Key concept: specification provides *worst-case* values.

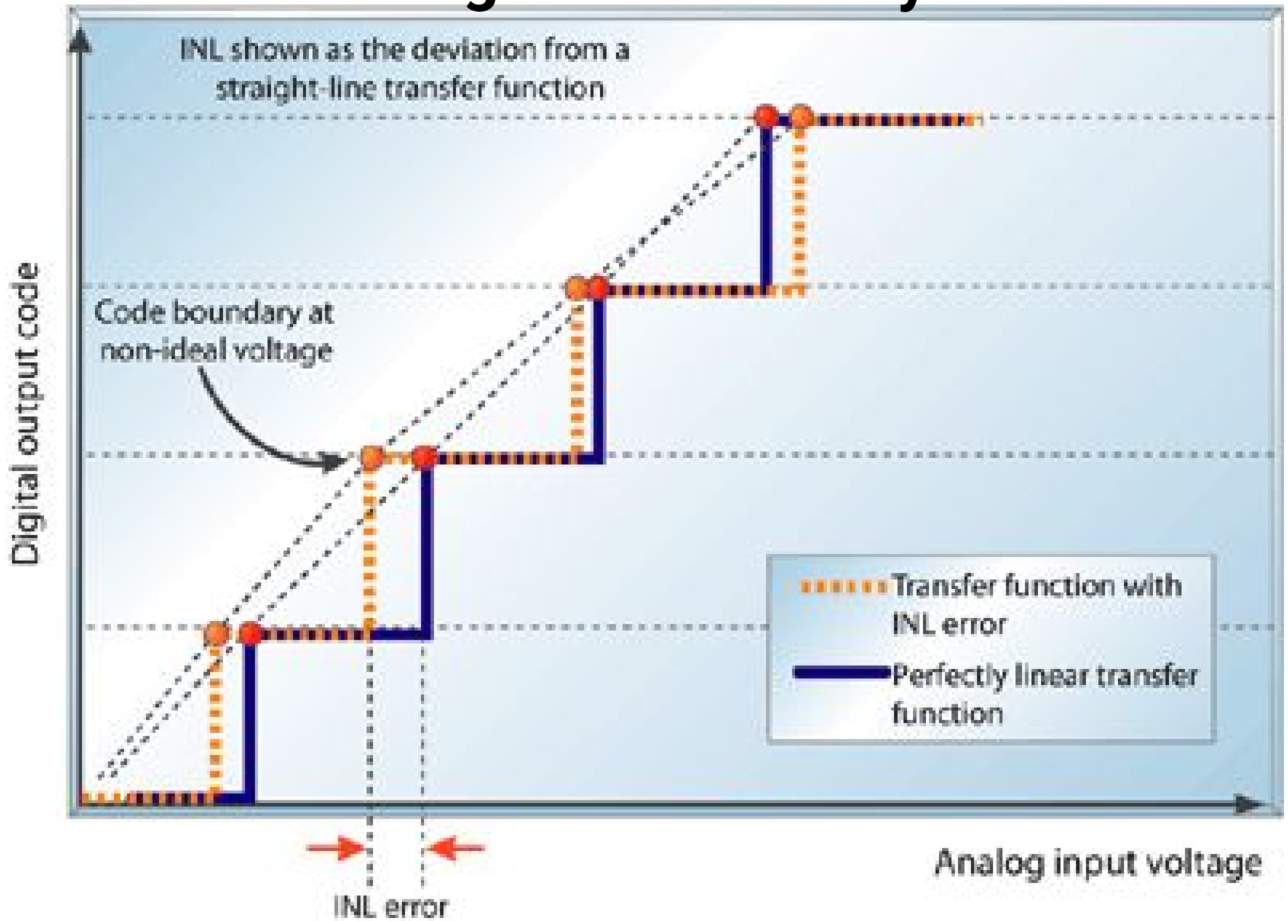
# Built-in ½ LSB error



# Built-in ½ LSB error corrected

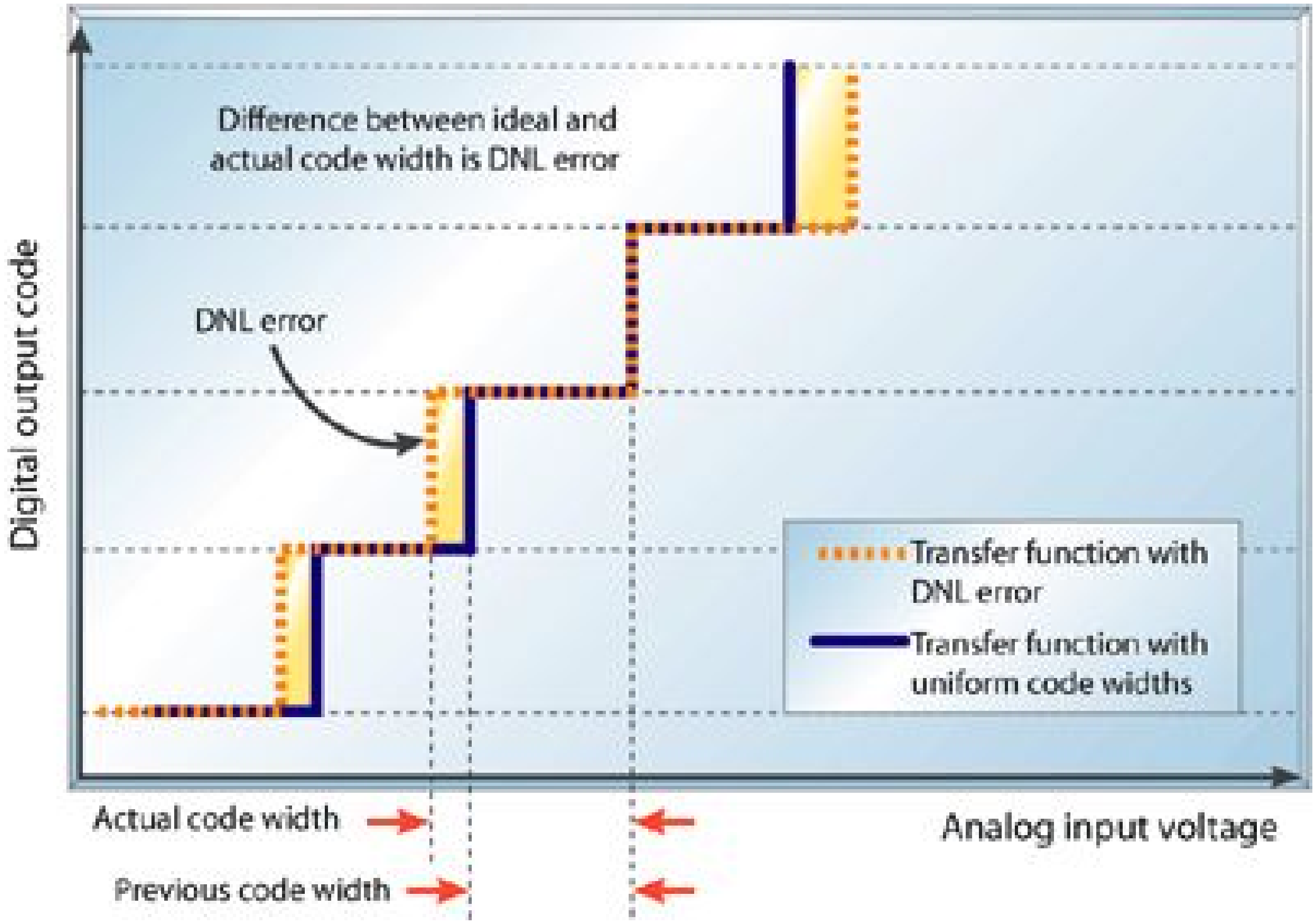


# Integral nonlinearity



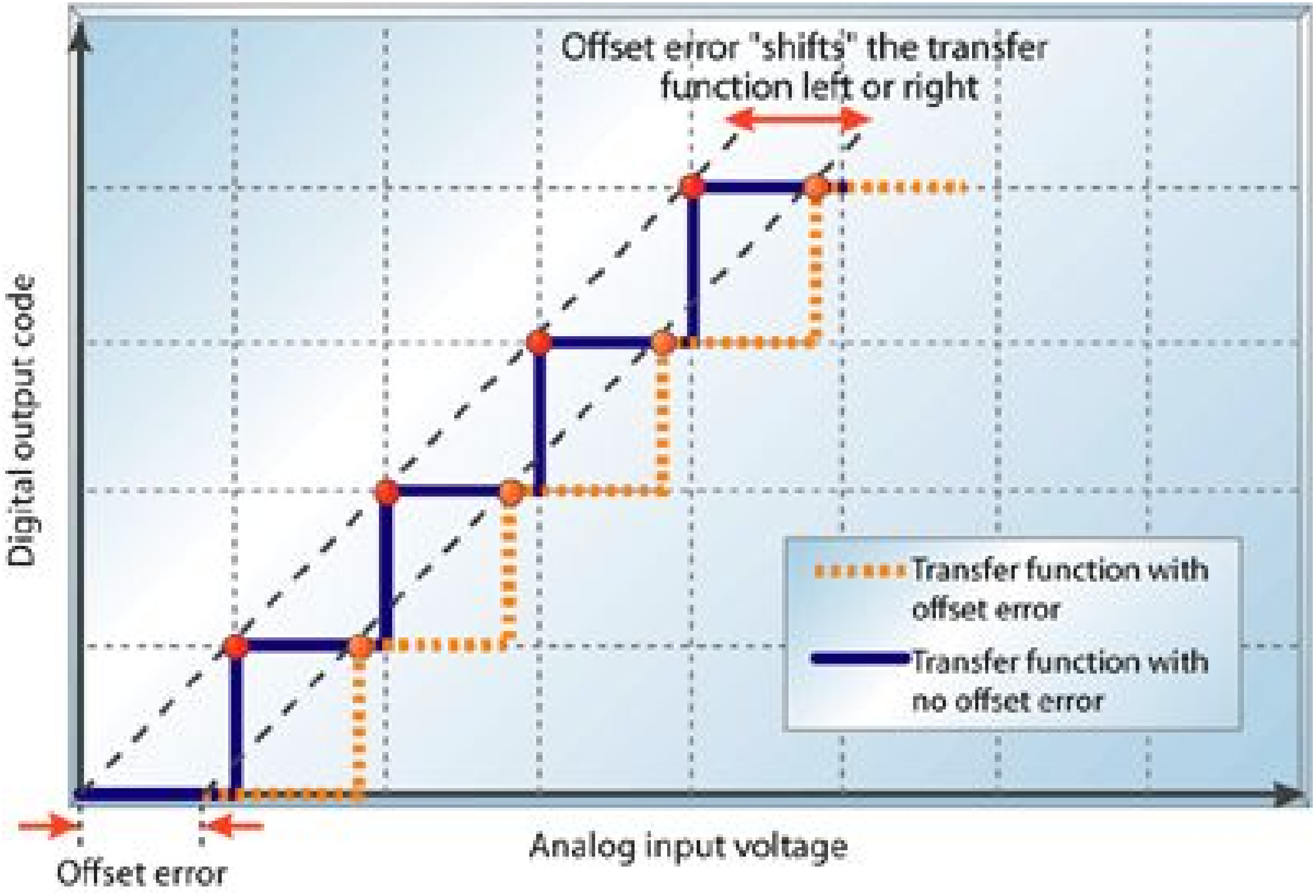

- Deviation of an ADC transfer function from a straight line.
- Best-fit line or highest to lowest points.
- Worst-case voltage deviation over all transitions.
- Express in LSB.
- INL error at any point in transfer function is integral of all lower DNL errors (next page).

# Differential nonlinearity

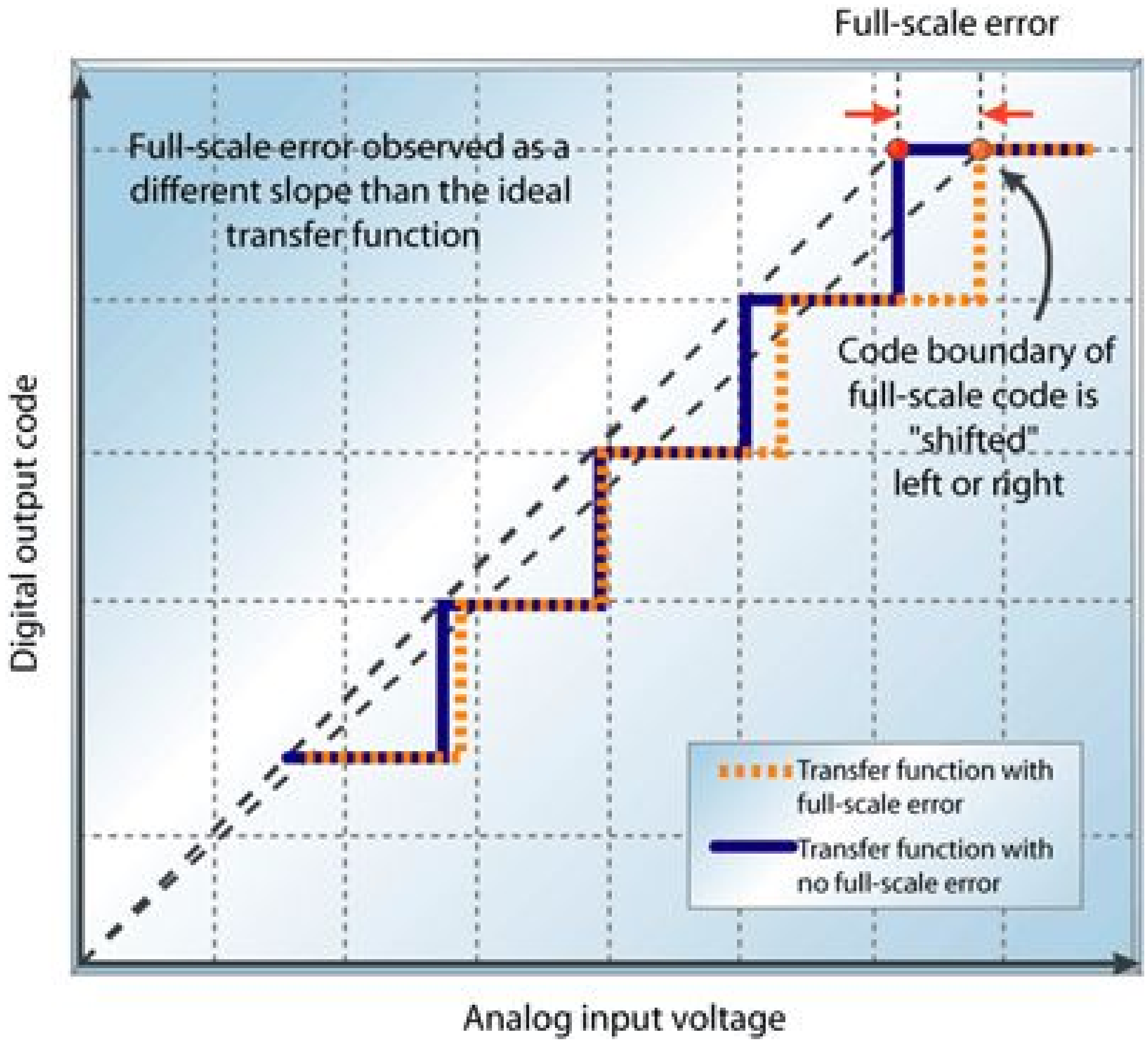



Worst-cases deviation of step size from ideal.

# Offset error



# Full-scale error is also sometimes called "gain error"




Difference between ideal and actual transition to highest code when offset error is zero.

# Errors

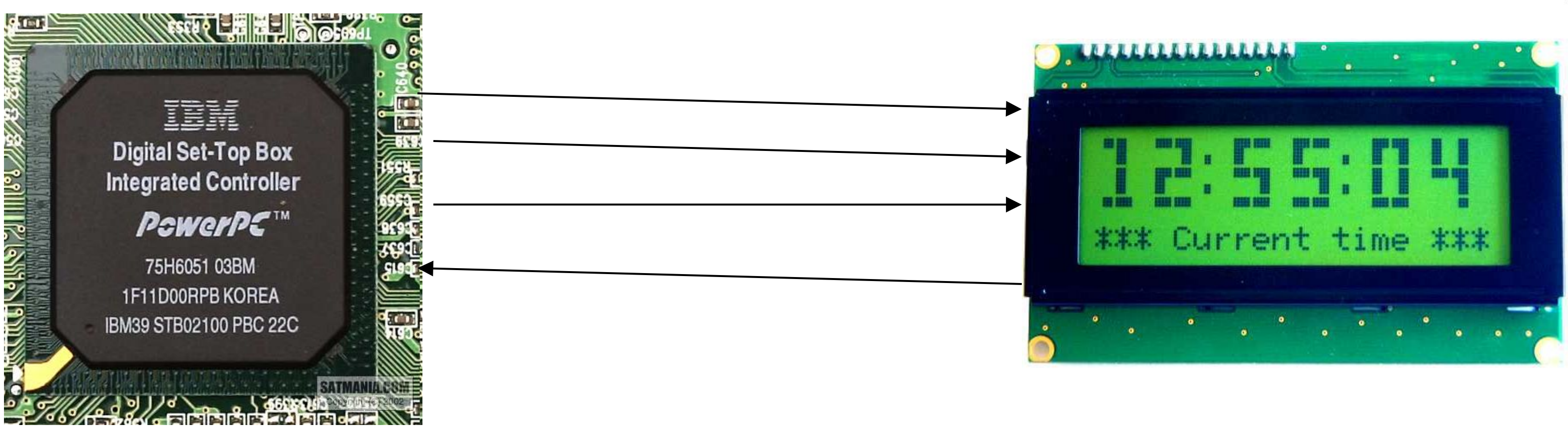
- Errors in a specification are worst-case.
  - So if you have an INL of ±0.25 LSB, you “know” that the device will never have more than 0.25 LSB error.
  - Temperature, input voltage, input current, etc.
- Some errors can be compensated for.
  - Nonlinearity.
  - Piece-wise linear lookup table.
  - Device-wise calibration can be expensive.
  - Automated or manufacturing process?
  - What about drift?

# Outline

- Serial buses
  - UART
  - I2C
- Datasheets
- ADCs and DACs
- **SPI**

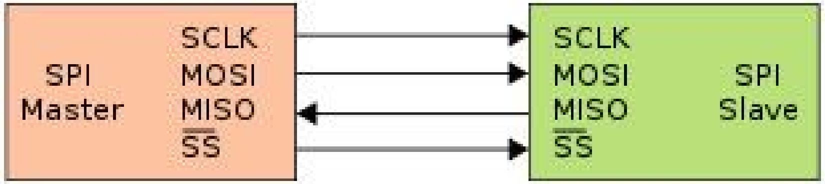
# What is SPI?

- Serial bus protocol.
- Fast, easy to use, simple.
- Widely supported.



# Introduction

- What is it?
- Basic Serial Peripheral Interface (SPI).
- Capabilities.
- Protocol.
- Pros / Cons and Competitor.
- Uses.
- Conclusion.




Serial Peripheral Interface

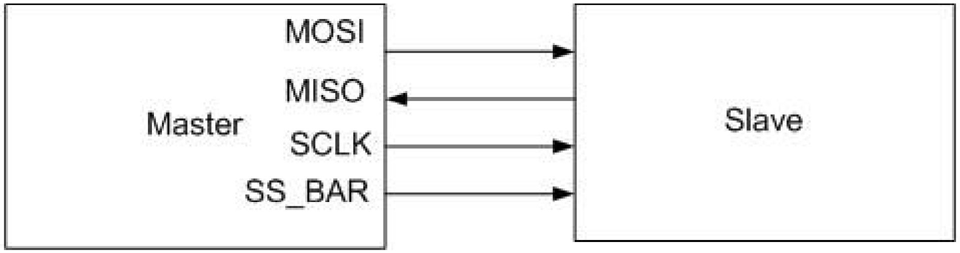
# SPI basics

- 4-wire bus.
- Short-range.
- Multiple targets, single initiator.
- Synchronous.

# Capabilities of SPI

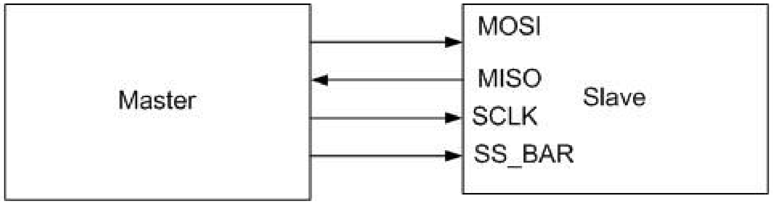
- Always full duplex.
  - Parallel bidirectional communication.
- Multiple Mb/s transmission speed.
- Transfers data in 4 to 16 bit characters.
- Multiple targets.
  - Daisy-chaining possible.

# Protocol




- Wires:
  - Master Out Slave In (MOSI).
  - Master In Slave Out (MISO).
  - System Clock (SCLK).
  - Slave Select 1…N (many of these).
- Initiator sets SS low.
- Initiator generates SCKL.
- Shift registers shift in and out data.

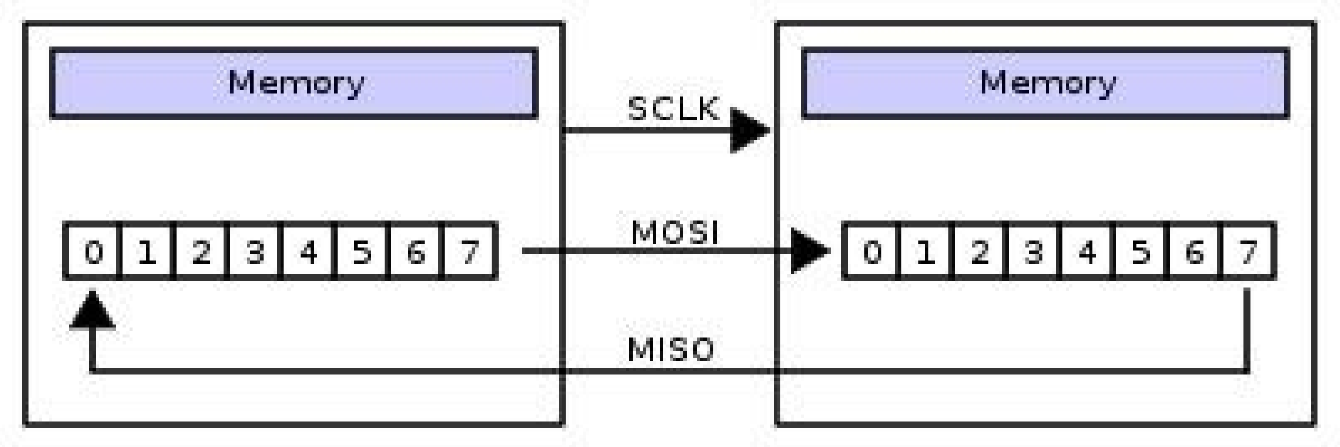
# Wires in detail




- MOSI – Carries data out of Initiator to Target.
- MISO – Carries data from Target to Initiator.
  - Both signals used for every transmission.
- SS_BAR – Unique line to select a target.
- SCLK – Initiator produced clock to synchronize data transfer.

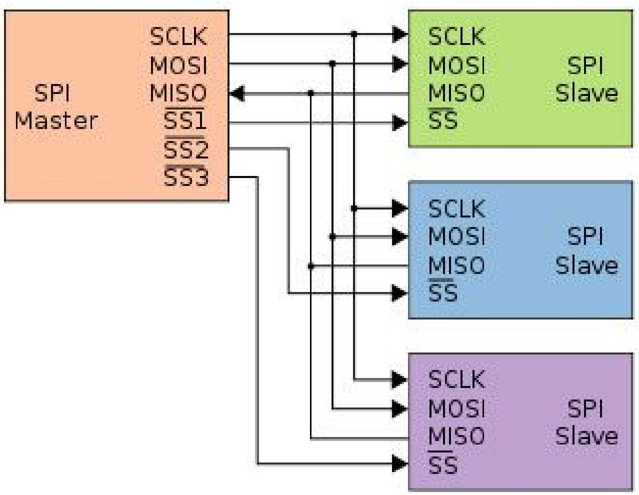
# Shifting protocol

Initiator Target

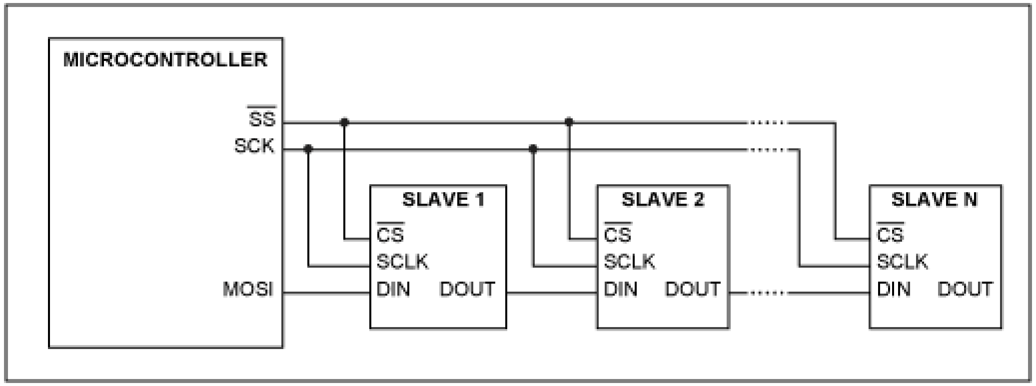



Initiator shifts out data to Target, and shift in data from Target

# Diagram




Initiator and multiple independent Targets




Some wires have been renamed

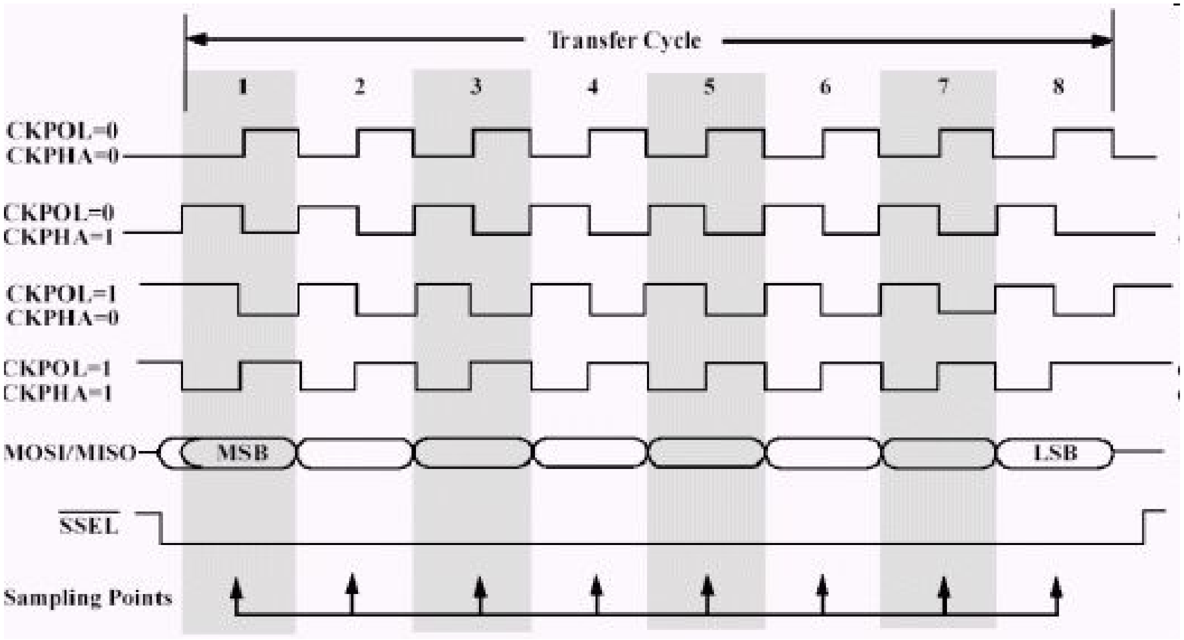
Initiator and multiple daisy-chained Targets

http://www.maxim-ic.com/appnotes.cfm/an_pk/3947

# Clock phase (advanced)

- Two phases and two polarities of clock.
- Four modes.
- Initiator and selected Target must be in same mode.
- Initiator must change polarity and phase to communicate with Targets with different modes.
- Data transmission and latching happen on alternating clock edges.
- Why the complexity?
  - Increases peripheral implementation flexibility.

# Timing Diagram




Timing Diagram – Showing Clock polarities and phases
http://www.maxim-ic.com.cn/images/appnotes/3078/3078Fig02.gif

# Pros and cons

Pros:

- Fast and easy.
  - Fast for point-to-point connections.
  - Easily allows streaming / constant data inflow.
  - No address bit / simple to implement.
- Full duplex.
- Widely supported.

Cons:

- SS signal makes multiple targets wiring-intensive.
- No ack capability.
- No inherent arbitration.
- No flow control.
- Four wires.

# Uses for clocking modes

- Some serial encoders/decoders, converters, serial LCDs, sensors, etc.
- Pre-SPI serial devices

# Summary

- SPI – 4 wire serial bus protocol
  - MOSI MISO SS SCLK wires.
- Full duplex.
- Multiple Targets, one Initiator.
- Best for point-to-point streaming data.
- Easily supported.

Done.