EECS 373
Design of Microprocessor-Based Systems

Robert Dick
University of Michigan

Lecture 9: Serial buses, ADCs, and DACs

2 February 2017

Slides inherited from Mark Brehob.

Outline

- Context and review
- Serial buses
  - UART
  - SPI
  - I2C
- ADCs and DACs
  - Fundamentals
  - Operation

Context and review

- Timers
  - Oscillator, counter, ISR, (OS routines,) library, application.
  - PWM a useful application.
  - Virtualization: use SW to increase apparent number of HW timers.
- Hazards
  - Fundamental definitions.
  - Problems they cause.
  - Side-stepping them.
    - Synchronous systems.
    - Pre-syncing signals using flip-flops.
  - Directly addressing them.
    - Logic design techniques.
- Setup and hold times

Terse project proposals

- Two, each 0.5-1 page in length.
- Due tomorrow.
- See sticky Piazza post for example and uploading instructions.

Team formation meeting

- 7:30pm-9pm on 7 Feb.
- 1311 EECS.
- Come prepared to very briefly pitch your project idea if you want to recruit team members.
- Everybody should leave with a team and a topic.

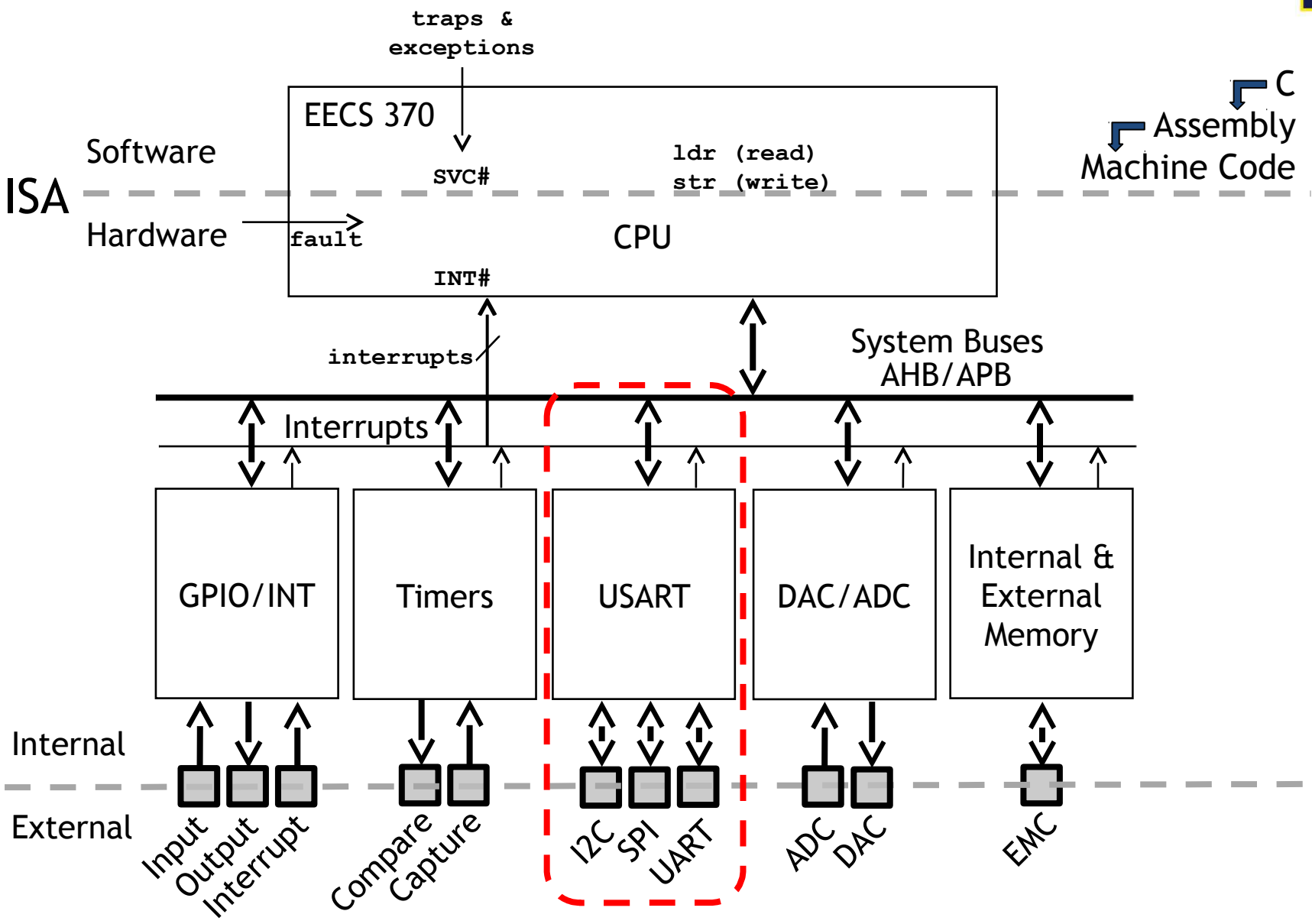
Student talks

- Each student will do 5-minute talk in March.
- Can team up with others by scheduling talks contiguously.
- See Piazza post for initial (random) schedule.
- Can change schedule, but follow rules.
- More on this later in course.

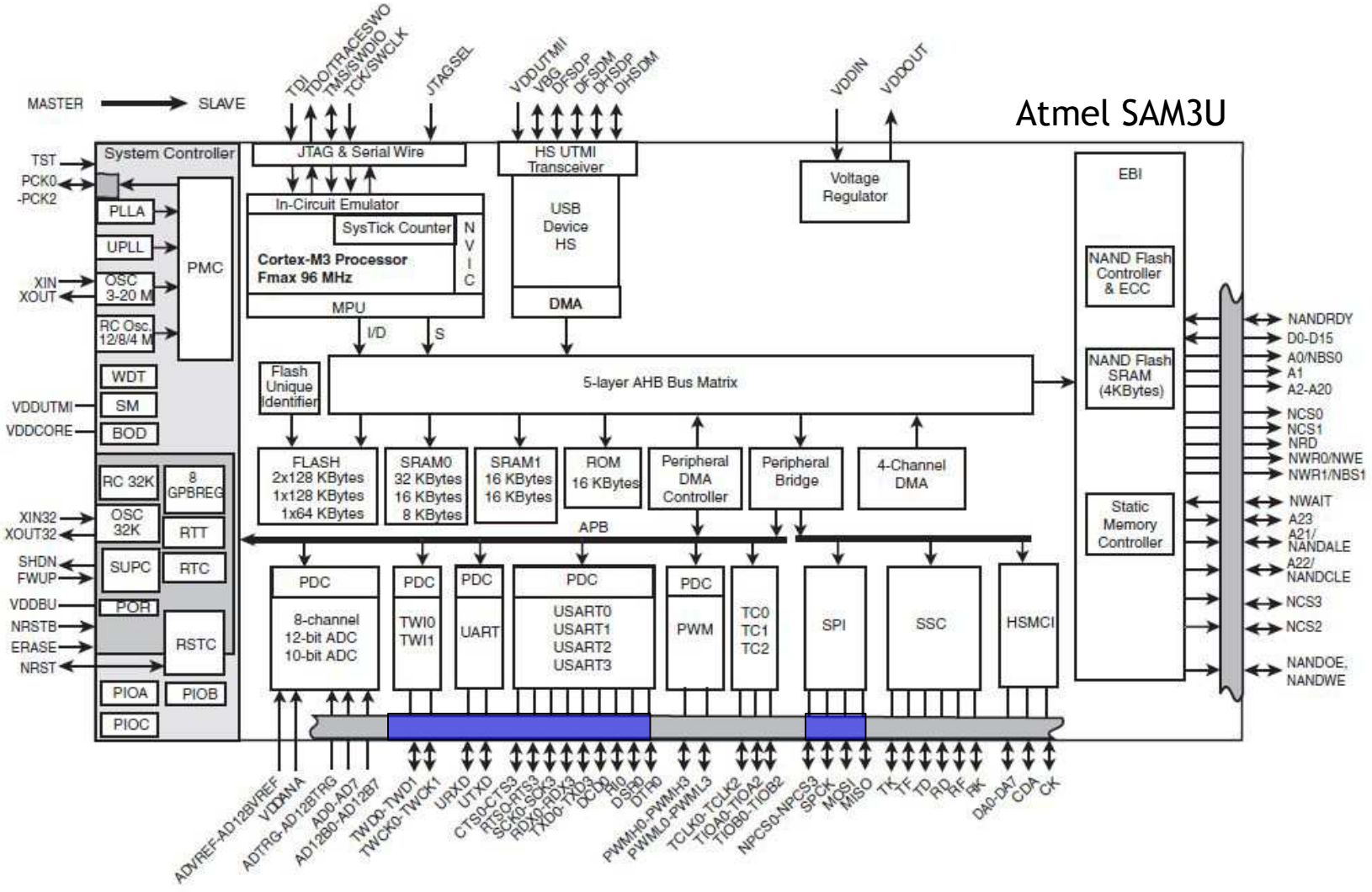
Outline

- ~~Context and review~~
- Serial buses
  - UART
  - SPI
  - I2C
- ADCs and DACs
  - Fundamentals
  - Operation

Serial interfaces

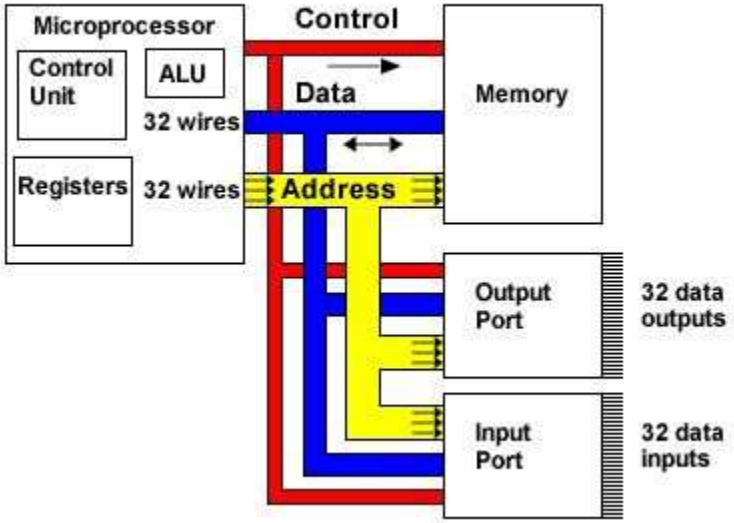


External memory attaches to the processor via the external memory controller and bus

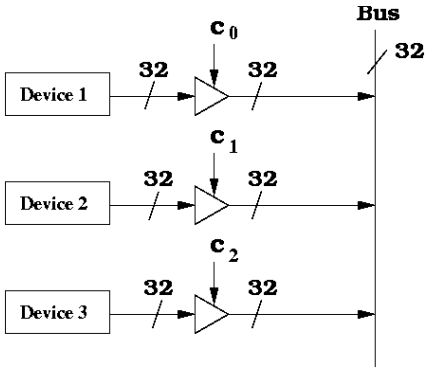


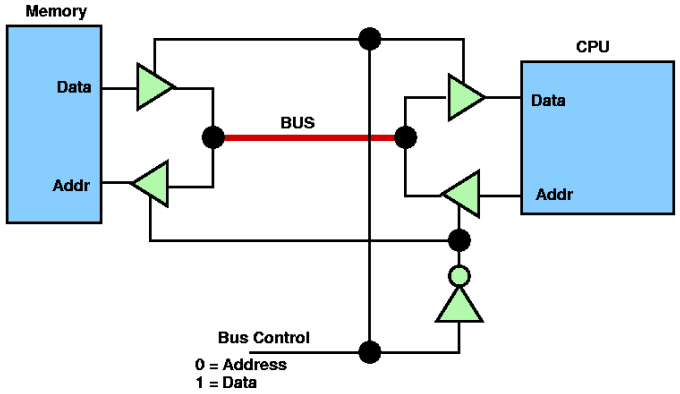
Bus organization

- Multidrop bus (MDB): all components are connected to the same set of wires.
- In the general case, a bus may have more than one device capable of driving it.
- That is, it may be a “multi-master” bus as discussed earlier.



Review: Multiple (potential) bus drivers (1)





Tri-state devices, just have one device drive at a time.

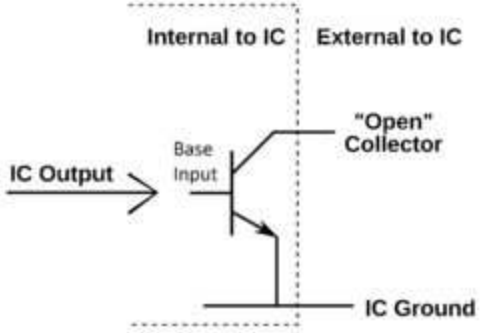
- Everyone can read though
  - Pros:
    - Fairly fast, pin-efficient.
  - Cons:
    - Tri-state devices can be slow.
      - Especially drive-to-tristate?
    - Need to be sure two folks not driving at the same time
      - Let out the magic smoke.
  - Most common solution (at least historically)
    - Ethernet, PCI, etc.

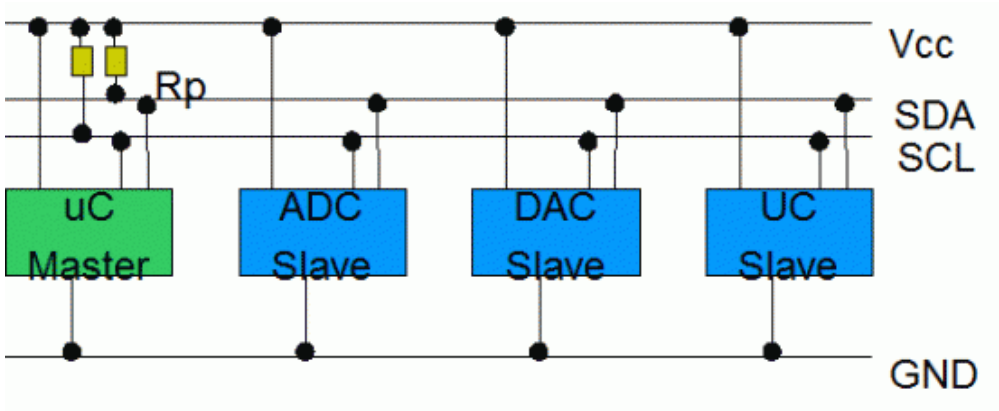
Review: Multiple (potential) bus drivers (2)

- 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"
  - Wire is pulled high by a resistor.
  - If any device pulls the wire low, it goes low.
- Pros:
  - If two devices both drive the bus, it still works!
- Cons:
  - Rise-time is long.
  - Constant power consumption.
- Used in I2C, CAN





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
  - Data format and transmission speeds are configurable
  - Electric signaling levels and methods (such as differential signaling etc.) handled by an external driver.

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

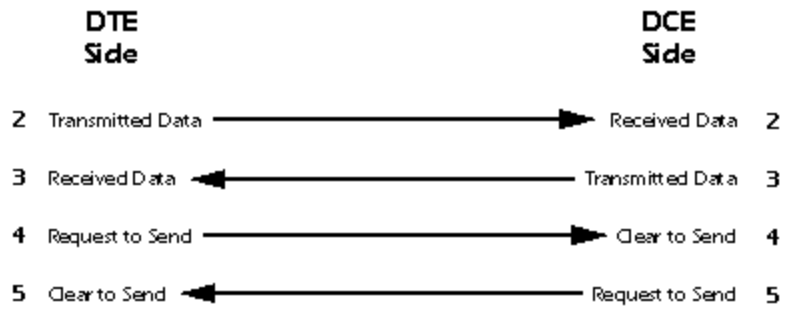
- UART is actually a generic term that includes a large number of different devices/standards.
- RS-232 is a standard.
- Specifies characteristics and timing of signals, the meaning of signals, and the physical size and pin out of connectors.

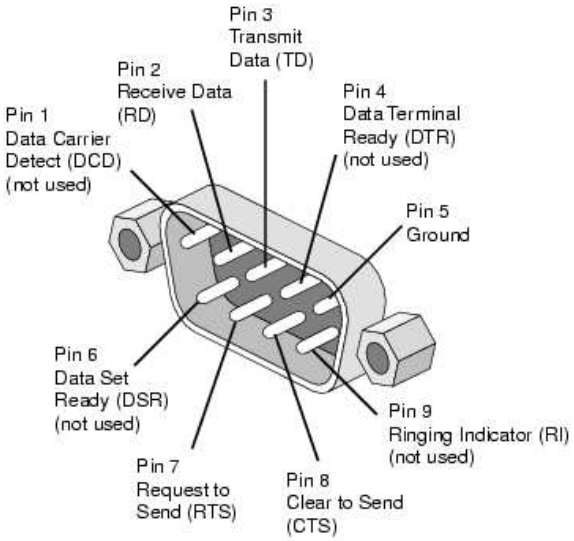
Signals (only most common)

- The **RXD** signal of a UART is the signal receiving the data. This will be an input and is usually connected to the TXD line of the downstream device.
- The **TXD** signal of a UART is the signal transmitting the data. This will be an output and is usually connected to the RXD line of the downstream device.
- The **RTS#** (Ready to Send) signal of a UART is used to indicate to the downstream device that the device is ready to receive data. This will be an output and is usually connected to the CTS# line of the downstream device.
- The **CTS#** (Clear to Send) signal of a UART is used by the downstream device to identify that it is OK to transmit data to the upsteam device. This will be an input and is usually connected to the RTS# line of the upstream device.

DB9 stuff

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

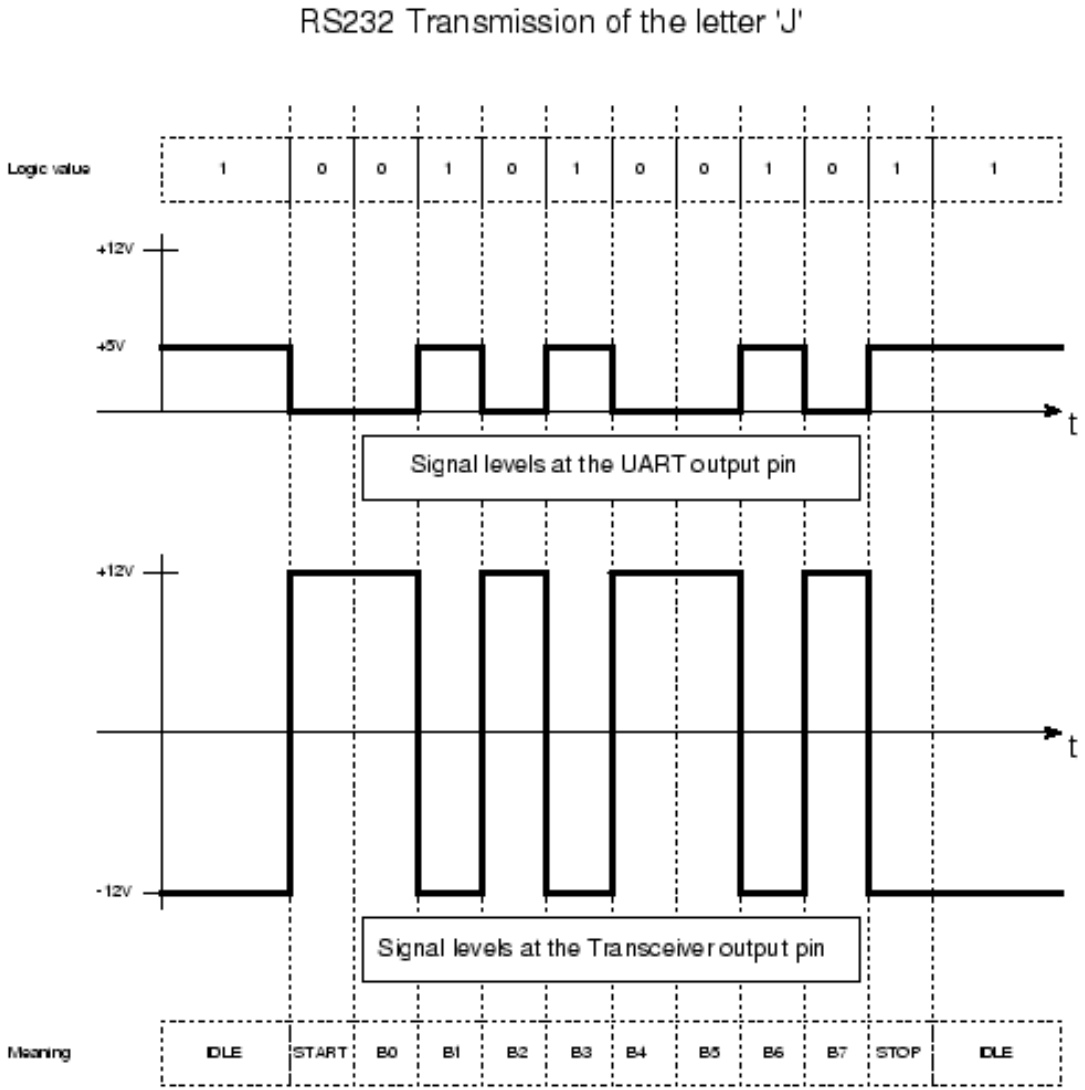




Pin Number	Signal	Description
1	DCD	Data carrier detect
2	RxD	Receive Data
3	TxD	Transmit Data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Ready to send
8	CTS	Clear to send
9	RI	Ring Indicator

Wiring a DTE (terminal) device to a DCE (modem) device for communication is easy.
The pins are a one-to-one connection, meaning all wires go from pin x to pin x.
A straight through cable is commonly used for this application.
In contrast, wiring two DTE devices together requires crossing the transmit and receive wires.
This cable is known as a null modem or crossover cable. Also easy to wire.

RS-232 transmission example

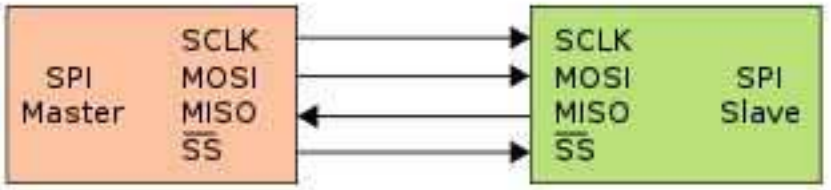


Outline

- ~~Context and review~~
- Serial buses
  - ~~UART~~
  - SPI
  - I2C
- ADCs and DACs
  - Fundamentals
  - Operation

Introduction

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

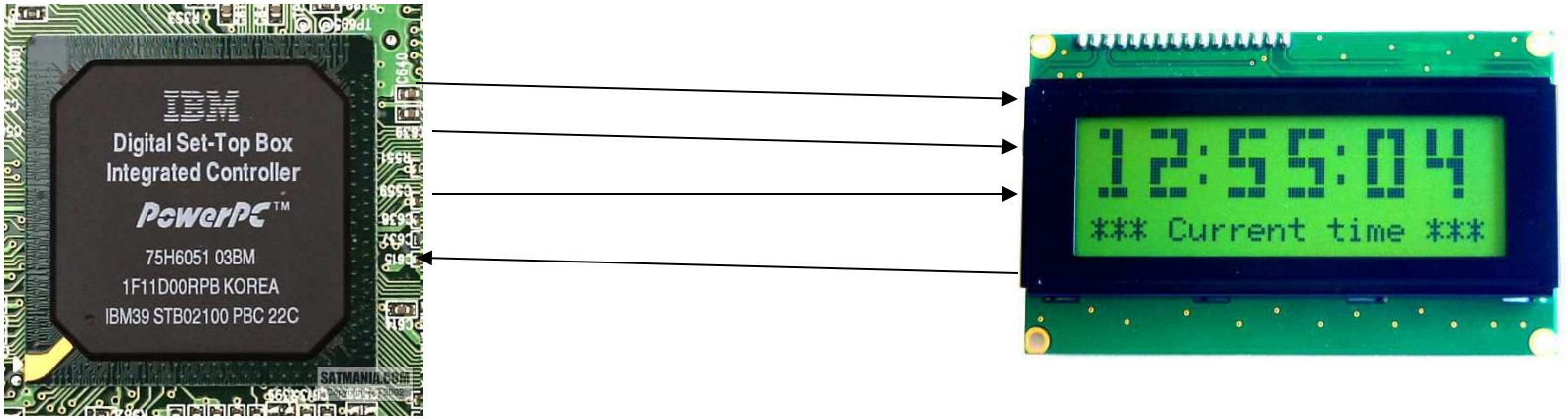


Serial Peripheral Interface

http://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/SPI_single_slave.svg/350px-SPI_single_slave.svg.png

What is SPI?

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



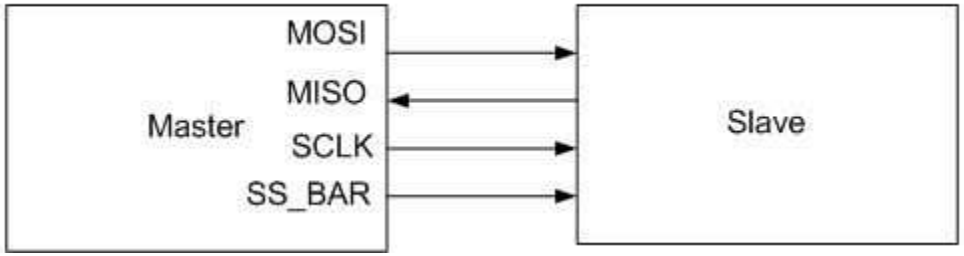
SPI basics

- A communication protocol using 4 wires
  - Also known as a 4 wire bus
- Used to communicate across small distances
- Multiple slaves, single master
- Synchronized

Capabilities of SPI

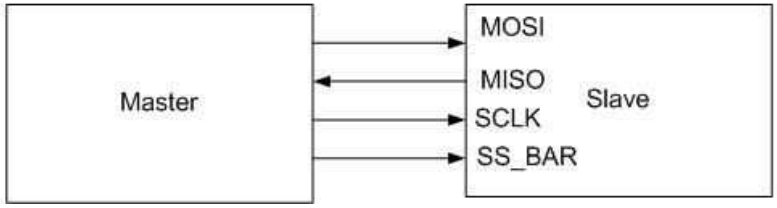
- Always Full Duplex
  - Communicating in two directions at the same time
  - Transmission need not be meaningful
- Multiple Mb/s transmission speed
- Transfers data in 4 to 16 bit characters
- Multiple slaves
  - Daisy-chaining possible

Protocol



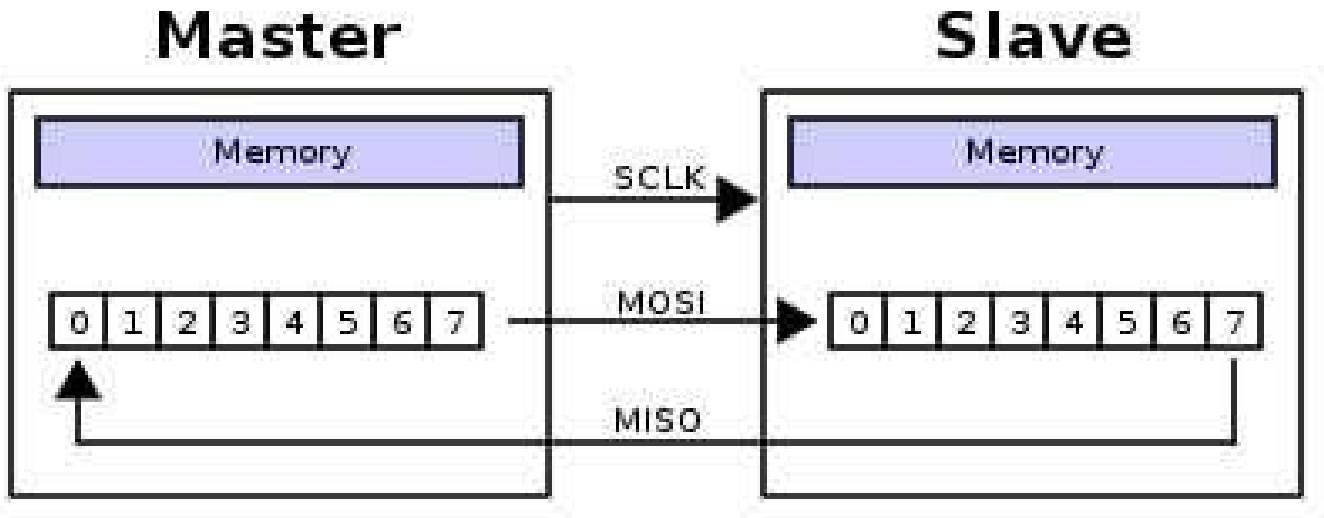
- Wires:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - System Clock (SCLK)
  - Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data

Wires in detail



- MOSI – Carries data out of Master to Slave
- MISO – Carries data from Slave to Master
  - Both signals used for every transmission
- SS_BAR – Unique line to select a slave
- SCLK – Master produced clock to synchronize data transfer

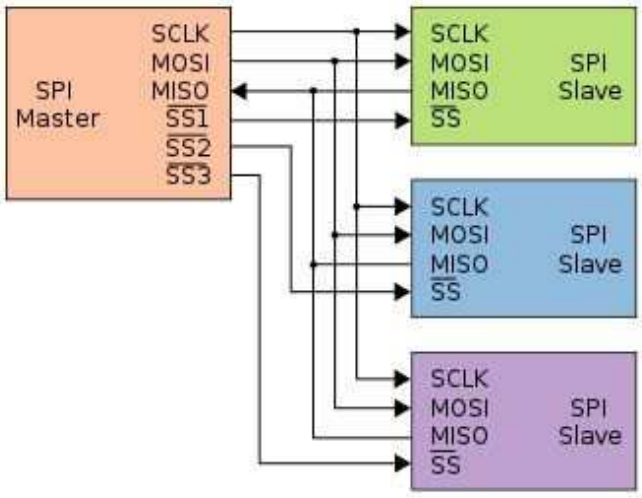
Shifting protocol



Master shifts out data to Slave, and shift in data from Slave

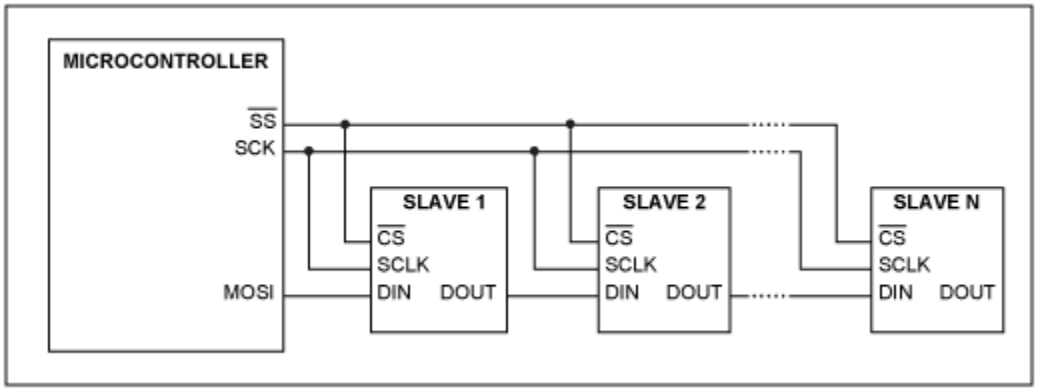
http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

Diagram



Master and multiple independent slaves

http://upload.wikimedia.org/wikipedia/commons/thumb/f/fc/SPI_three_slaves.svg/350px-SPI_three_slaves.svg.png



Some wires have been renamed

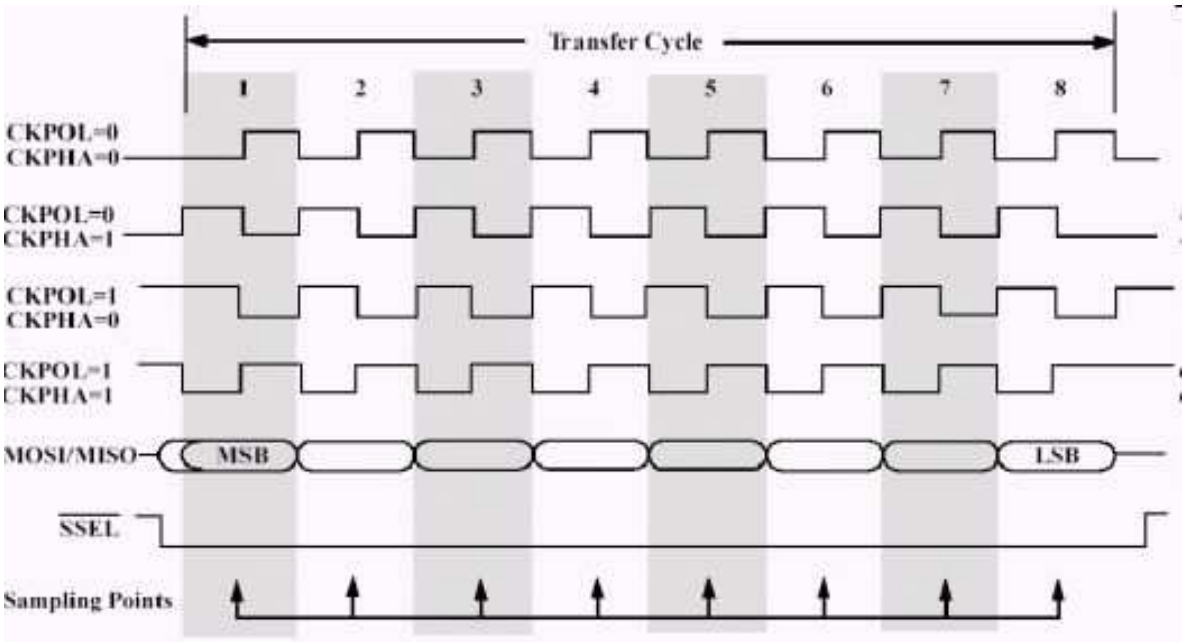
Master and multiple daisy-chained slaves

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

Clock phase (advanced)

- Two phases and two polarities of clock.
- Four modes.
- Master and selected slave must be in same mode.
- Master must change polarity and phase to communicate with slaves of different numbers.
- Data transmission and latching happen on alternating clock edges.
- Why? Increases implementation flexibility for bus users.

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 addressing/Simple to implement
- Full duplex
- Widely supported

Cons:

- SS signal makes multiple slaves complicated
- No ack capability
- No inherent arbitration
- No flow control
- Four wires

Uses

- 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 slaves, one master
- Best for point-to-point streaming data
- Easily supported

Outline

- ~~Context and review~~
- Serial buses
  - ~~UART~~
  - ~~SPI~~
  - I2C
- ADCs and DACs
  - Fundamentals
  - Operation

I^2C summary

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

I^2C applications

- Initially in TV sets.
- Common for peripherals from many companies.
- Real-time clocks and temperature sensors on general-purpose computer motherboards.

I²C technical description

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

I²C wiring

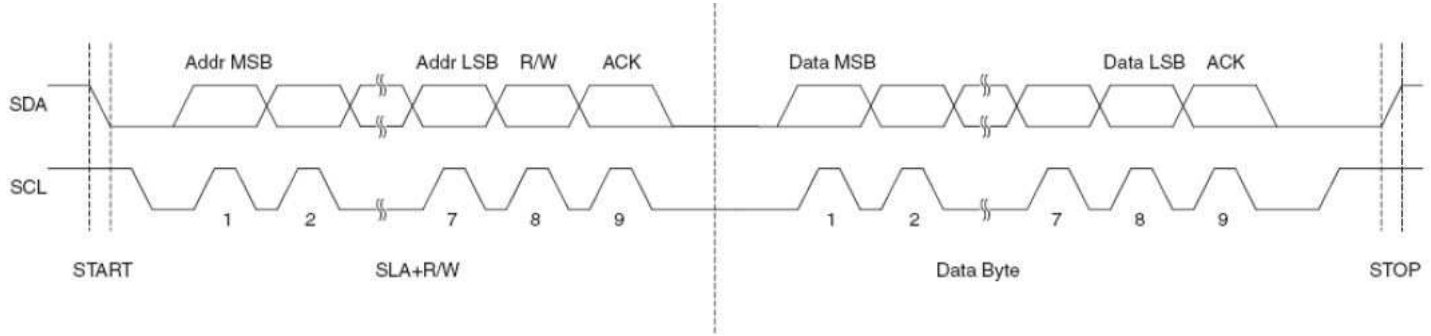
- SDA: data.
- SCL: clock.
- Open collector.
- Simple interfacing among voltage domains.

I²C clocking

- Unconventional.
- Quiescent state is high.
- Master pulses low during transmission.
- Slave holds clock low to extend transmission cycle.
  - One advantage of open collector design.

I²C transaction

- Master initiates.
- Start.
- Address.
- Data.
- Ack.
- Stop.



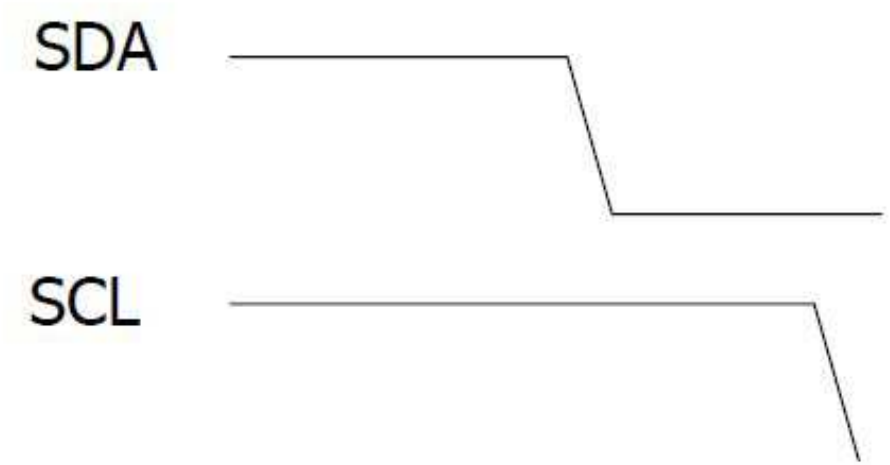
Source: ATMega8 Handbook

I²C roles

- Transmitter/receiver not same as master/slave.
- Master initiates transactions.
- Transmitter sends data on SDA.
- Receiver acks.
- Read: slave is transmitter.
- Write: master is transmitter.

I²C starting

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

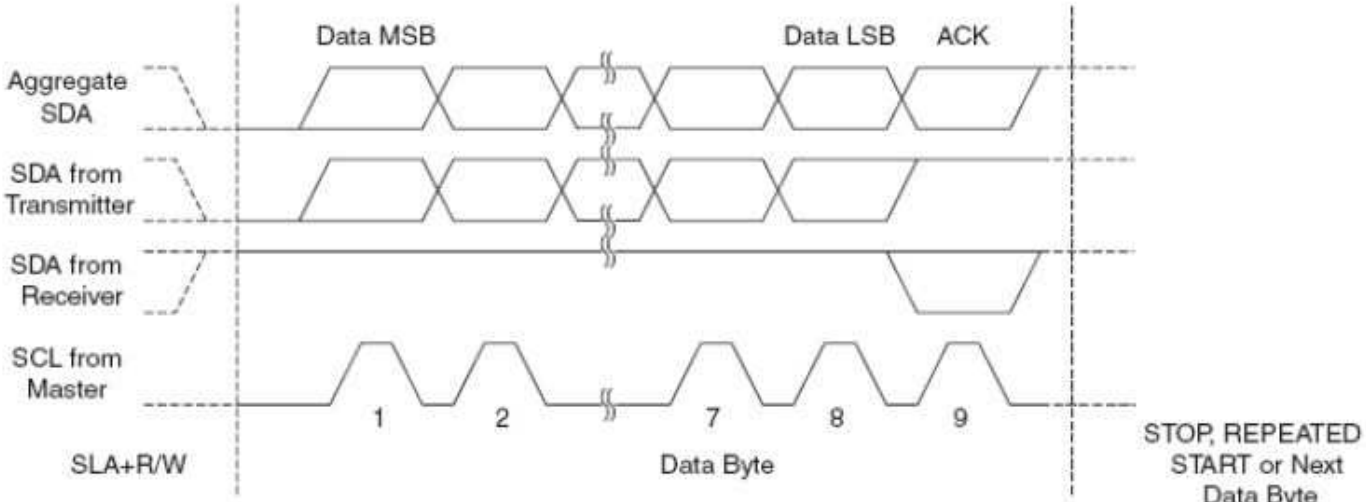


I²C address

- Sampled on rising SCL.
- 7-bit address.
- 8th bit
  - Low: write.
  - High: read.
- Philips/NXP can assign standard addresses for a fee.

I²C data

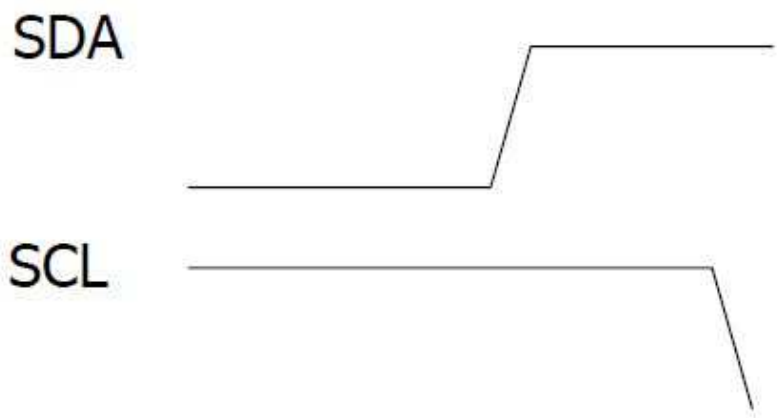
- Sampled on rising SCL.
- 8-bit.
- Write: master transmits, slave acks.
- Read: slave transmits, master acks.
- Continues until master signals to stop.



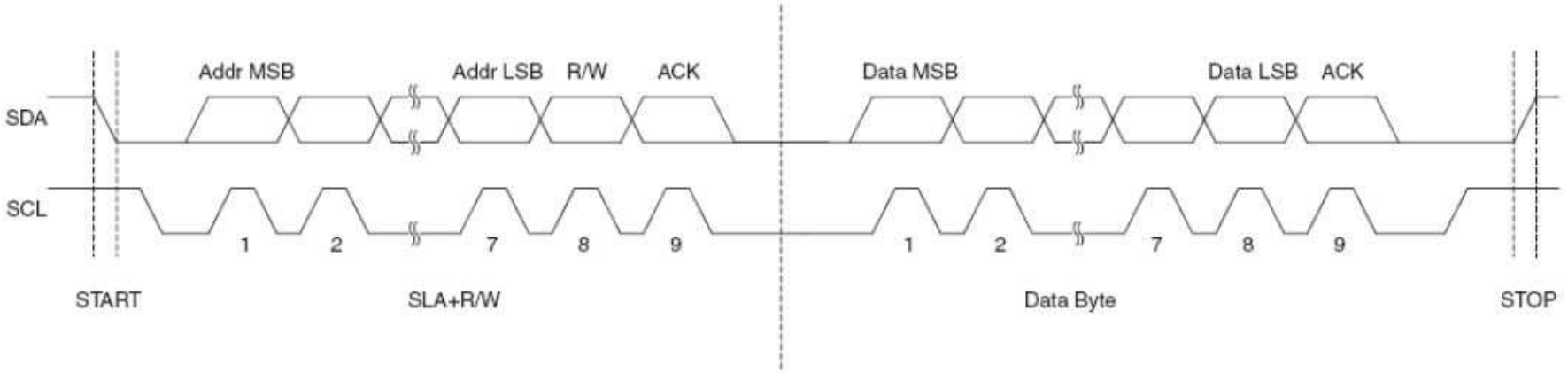
Source: ATMega8 Handbook

I²C stopping

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



I²C timing diagram



Source: ATMega8 Handbook

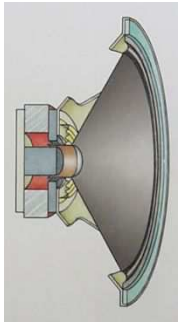
Outline

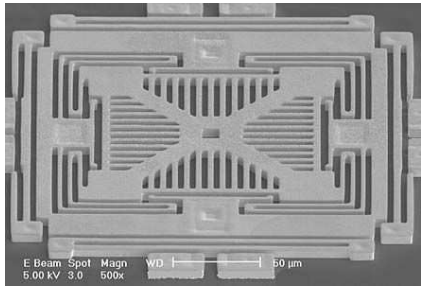
- ~~Context and review~~
- ~~Serial buses~~
  - ~~UART~~
  - ~~SPI~~
  - ~~I2C~~
- ADCs and DACs
  - Fundamentals
  - Operation

Many signals effectively analog

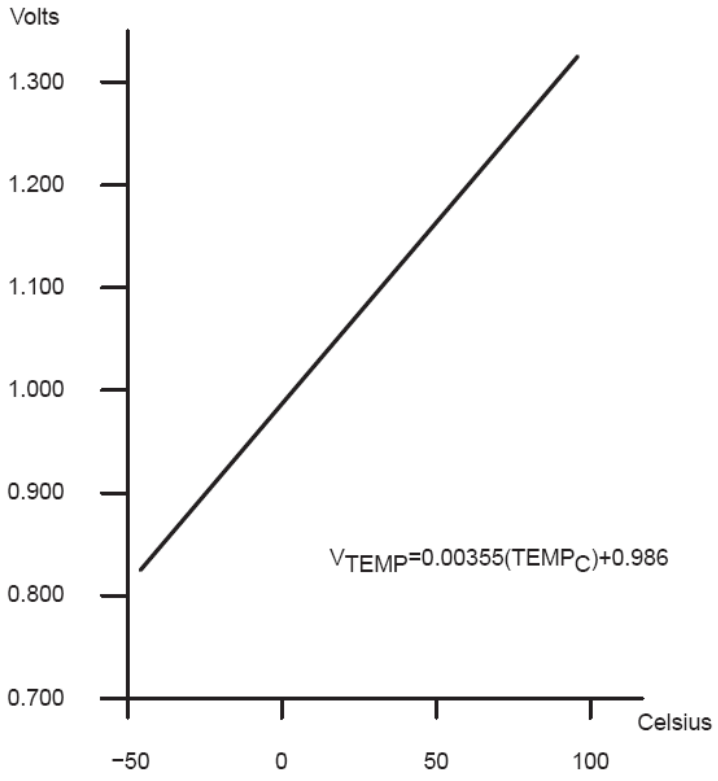
- At the macroscopic level, many signals have so many possible values they are effectively continuous/analog.
  - Sound, light, temperature, pressure, voltage, etc.

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



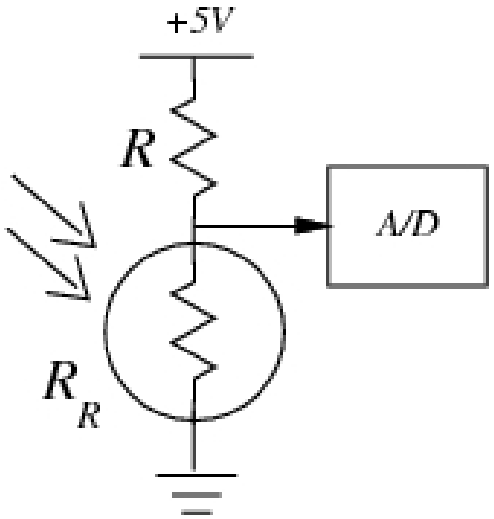


Transducers convert one form of energy into another



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~50-200kW.
- C~20pF.
- So, T_{RC}~20-80uS.
- f_{RC} ~ 10-50kHz.



Source: Forrest Brewer

Many other common sensors (some digital)

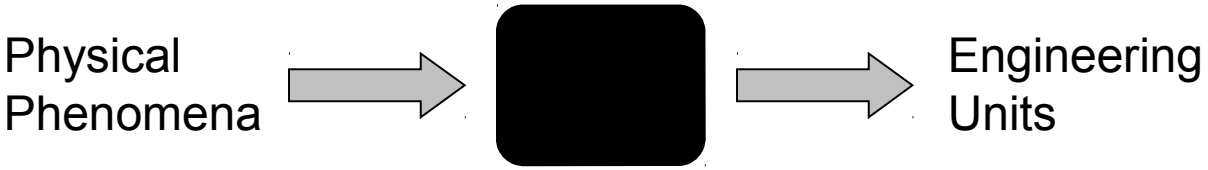
- Force
  - Strain gauges - foil, conductive ink
  - Conductive rubber
  - Rheostatic fluids
    - Piezorestive (needs bridge)
  - Piezoelectric films
  - Capacitive force
- Sound
  - Microphones: current or charge
  - Sonar: usually piezoelectric
- Position
  - Switches
  - Shaft encoders
  - Gyros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

Many other common sensors (some digital)

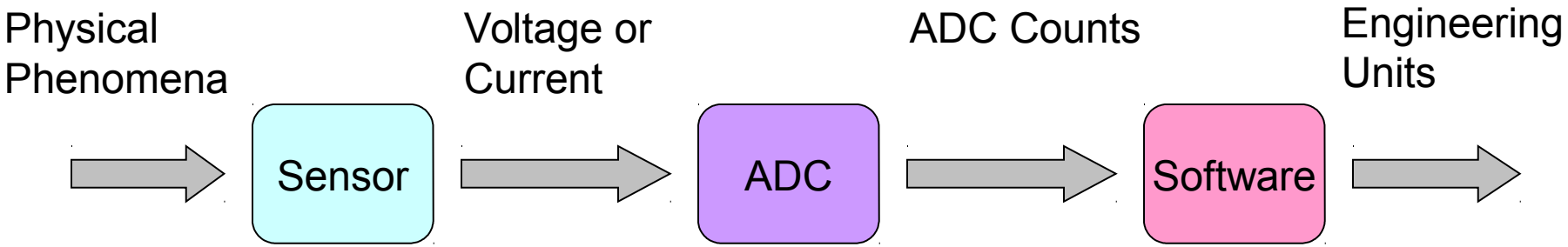
- Acceleration
  - MEMS
  - Pendulum
- Monitoring
  - Battery energy
  - Motor velocity
  - Temperature
- Field
  - Antenna
  - Magnetic
    - Hall effect
    - Flux gate
- Location
  - Permittivity
  - Dielectric
- Conductivity
- Many, many more

Analog to digital

- Goal

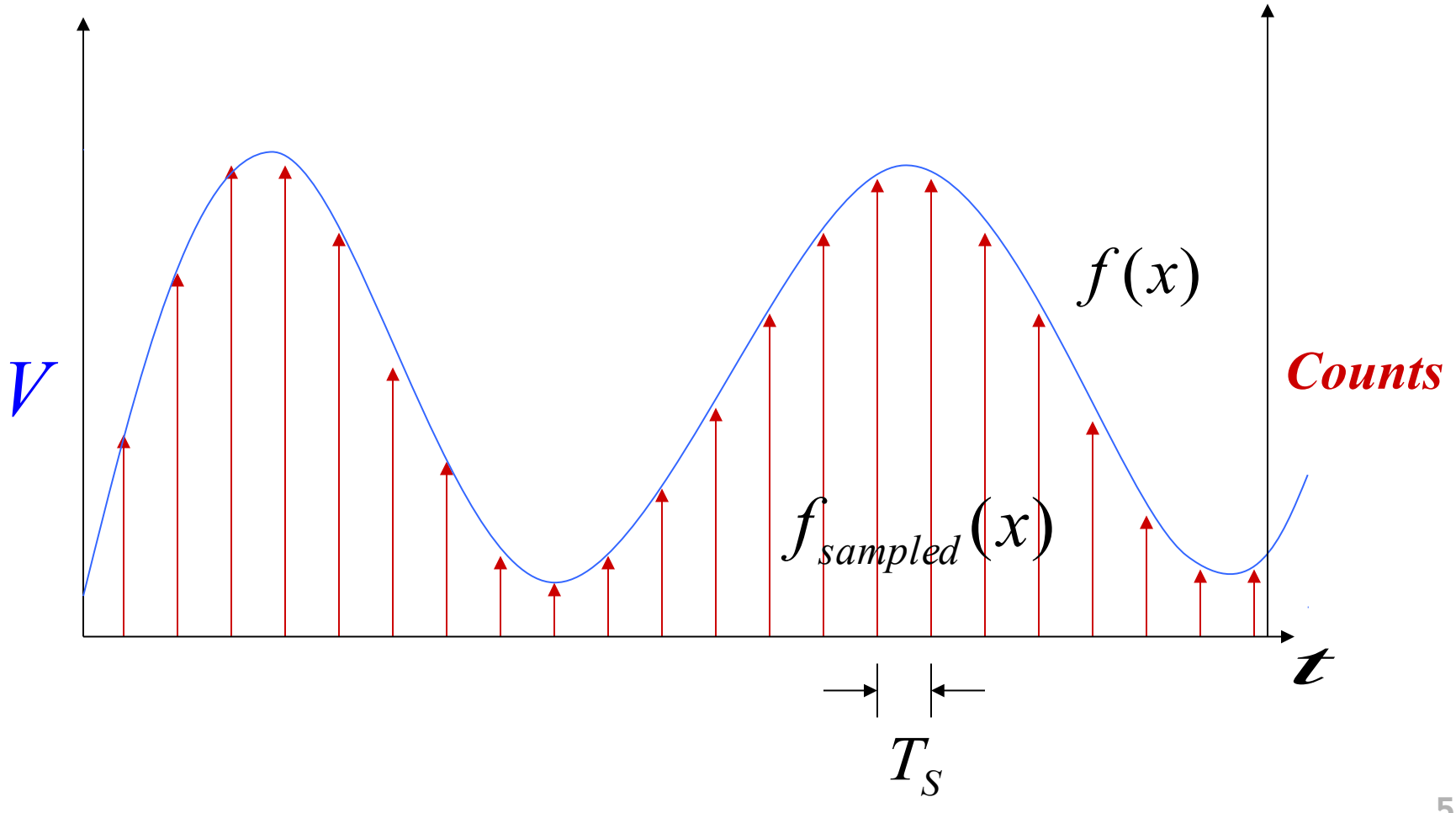


- Process



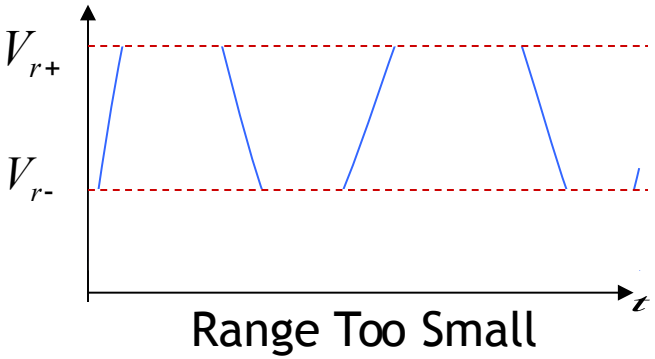
Digital representation of analog signal

- Discretize in time and value.
- Time series of discrete values

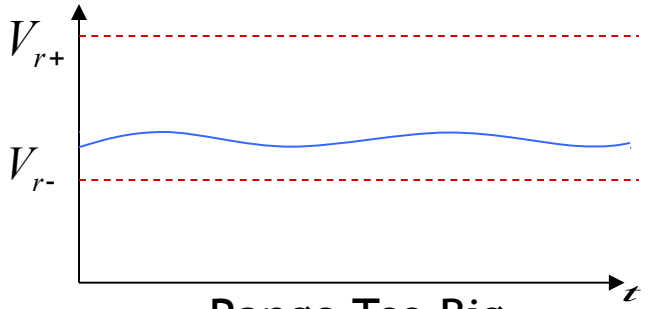


Choosing the value range

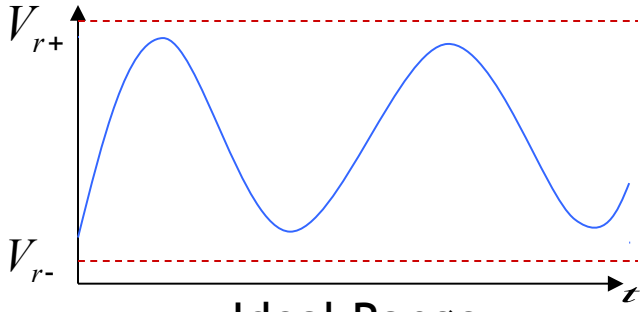
- What do the sample values represent?
  - Some fraction within the range of values



Range Too Small



Range Too Big



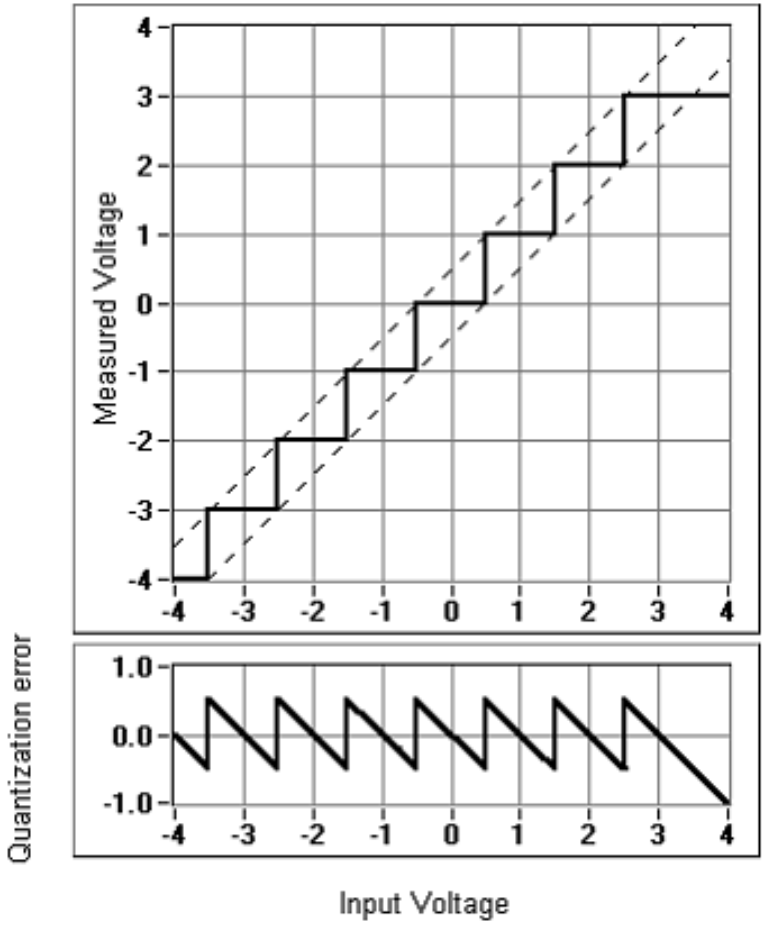
Ideal Range

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

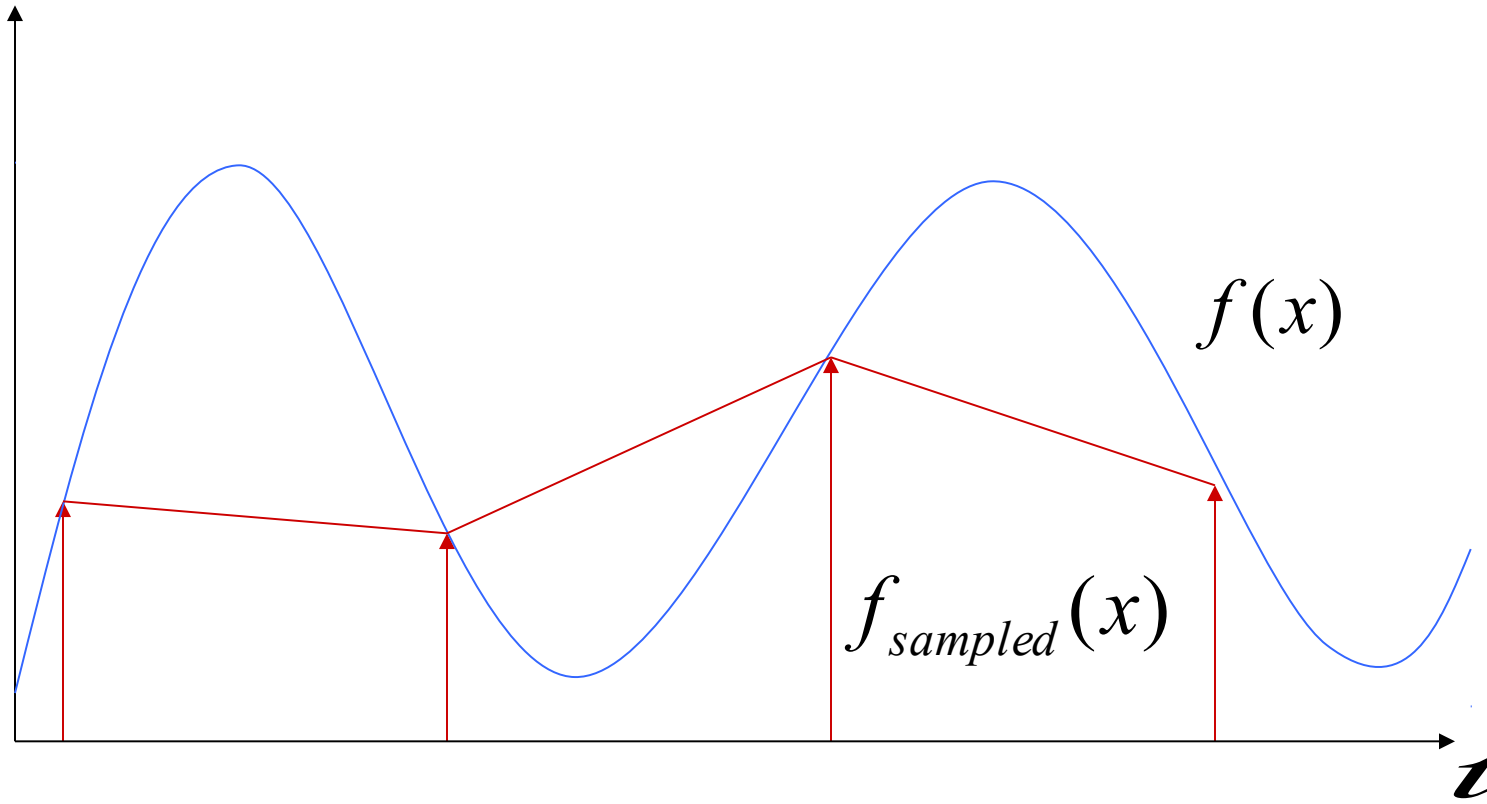
- Quantization Error
  - How far off discrete value is from actual
  - ½ LSB → Range / 8192

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:

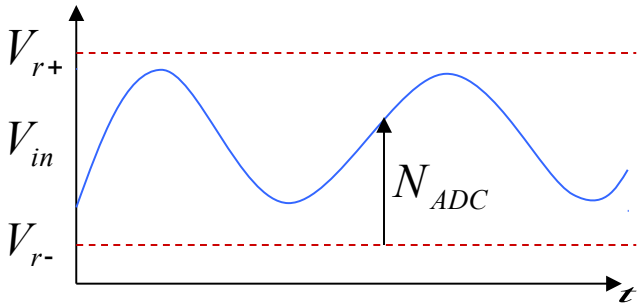
$$f_{\text{samples}} > 2 f_{\max}$$

- 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 filter to improve accuracy at cost of decreased effective samping rate.

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

$$V_{\text{TEMP}} = 0.00355(\text{TEMP}_{\text{C}}) + 0.986$$

$$\text{TEMP}_{\text{C}} = \frac{V_{\text{TEMP}} - 0.986}{0.00355}$$

A note about sampling and arithmetic

- Common error converting values

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

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

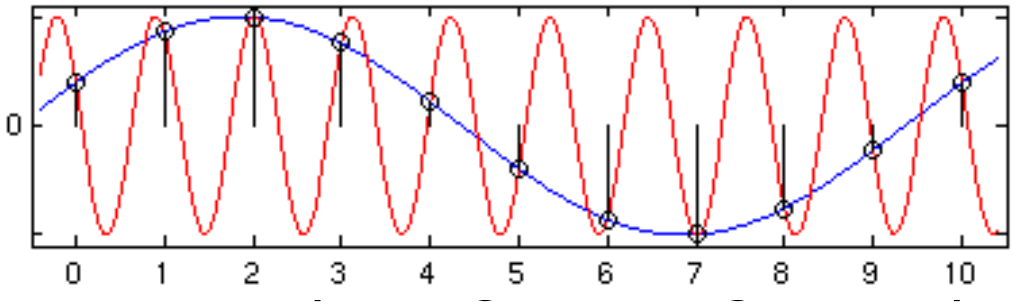
Learn associativity and type conversion rules of ANSI C.

- Fixed point operations
  - Overflow and underflow dangers complicate design.
  - This is deep. Talk to me if you want to try it.

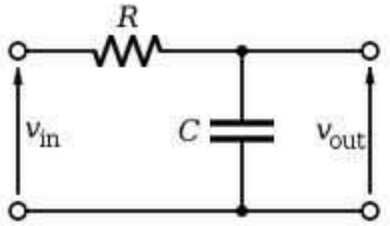
- Floating point operations
  - Often software emulated.
  - Often slow, power-hungry on embedded processors.

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

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



- Cond s filter
  - Removes high-frequency components
  - (a.k.a. anti-aliasing filter)

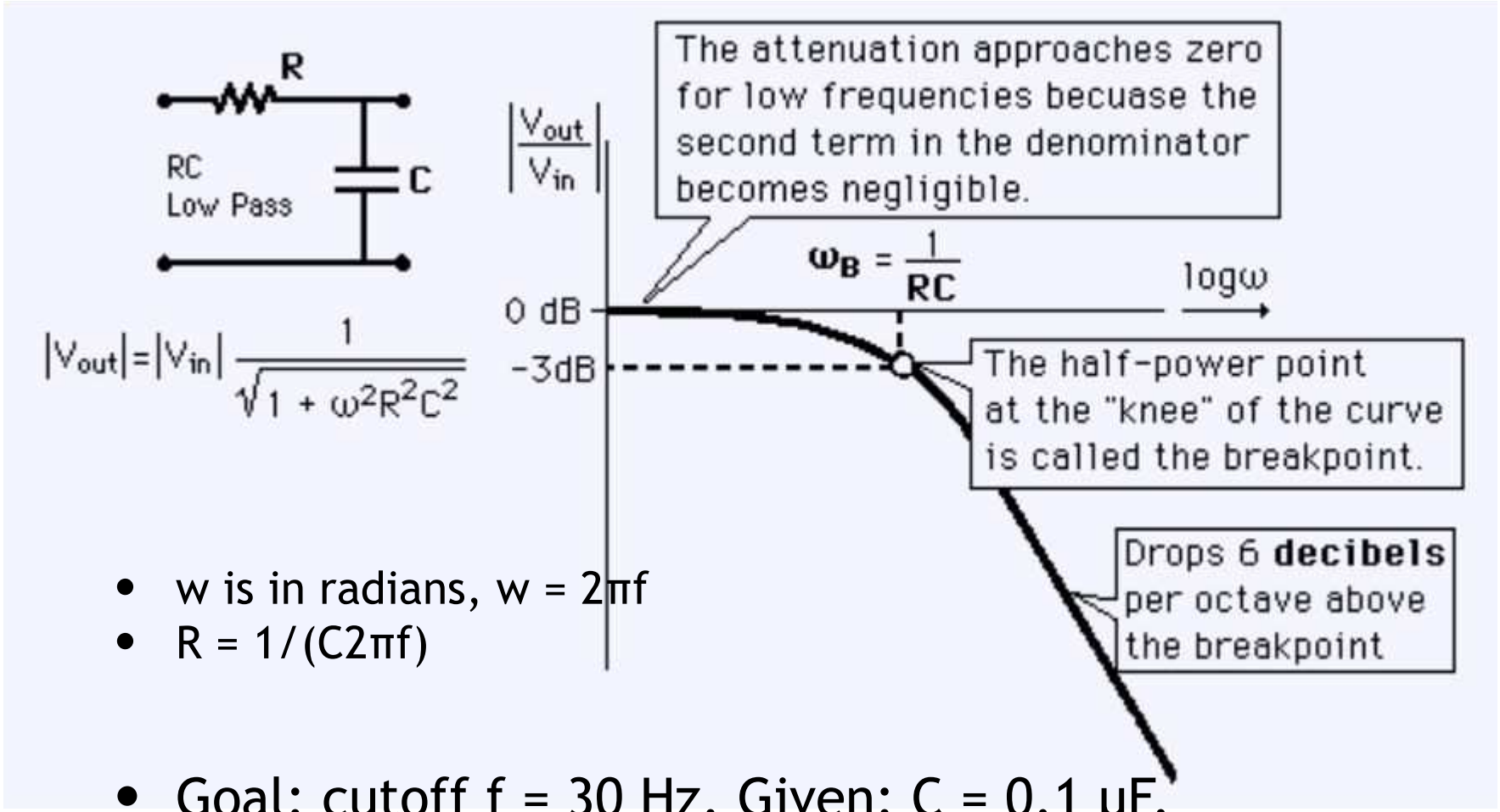


Do I really need to condition my input signal?

- 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.

Designing the anti-aliasing filter



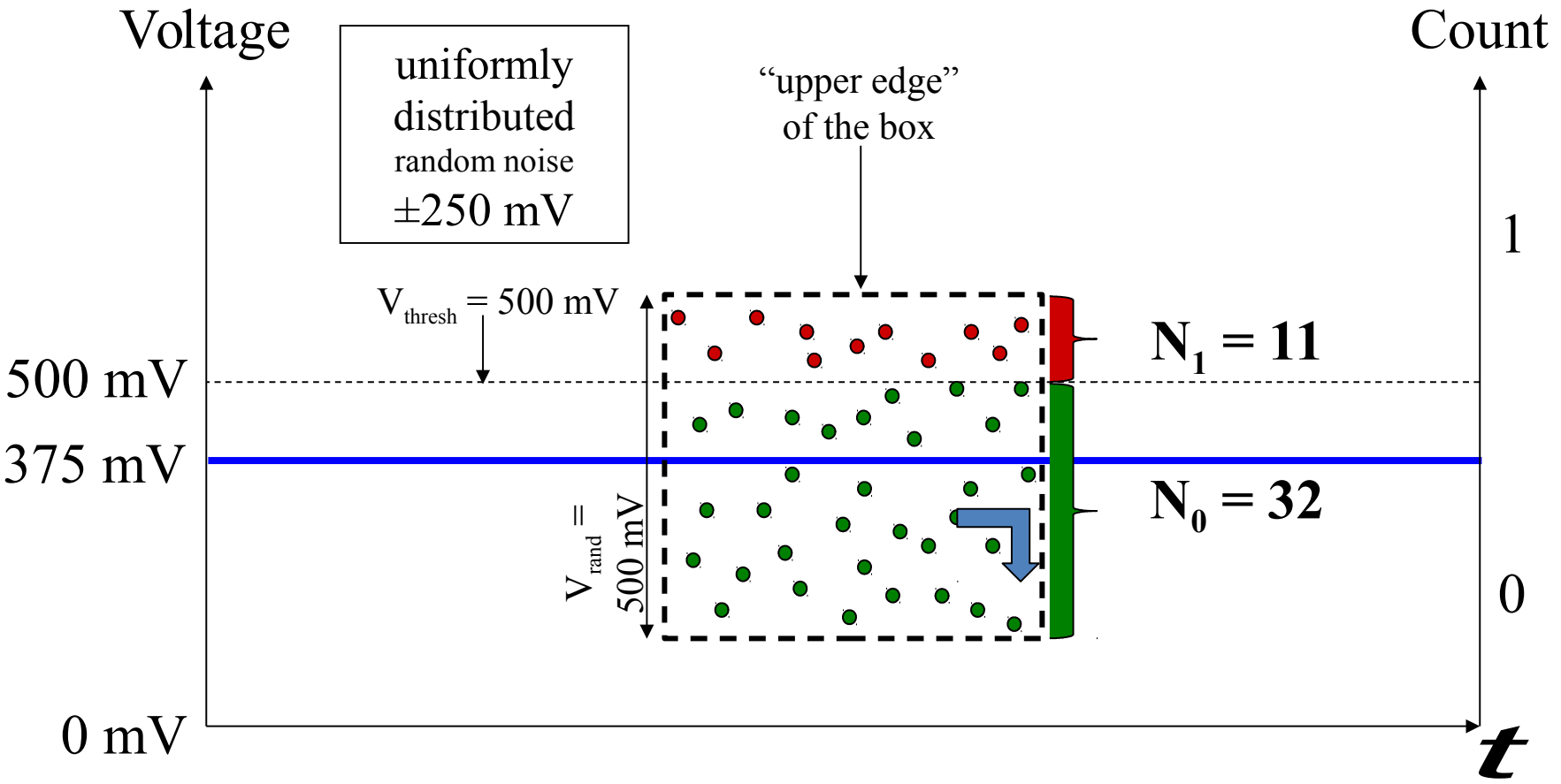
- w is in radians, w = 2πf
- R = 1/(C2πf)

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

Oversampling

- Can intentionally introduce or leave high-frequency noise and oversample.
- Reduces quantization error.

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

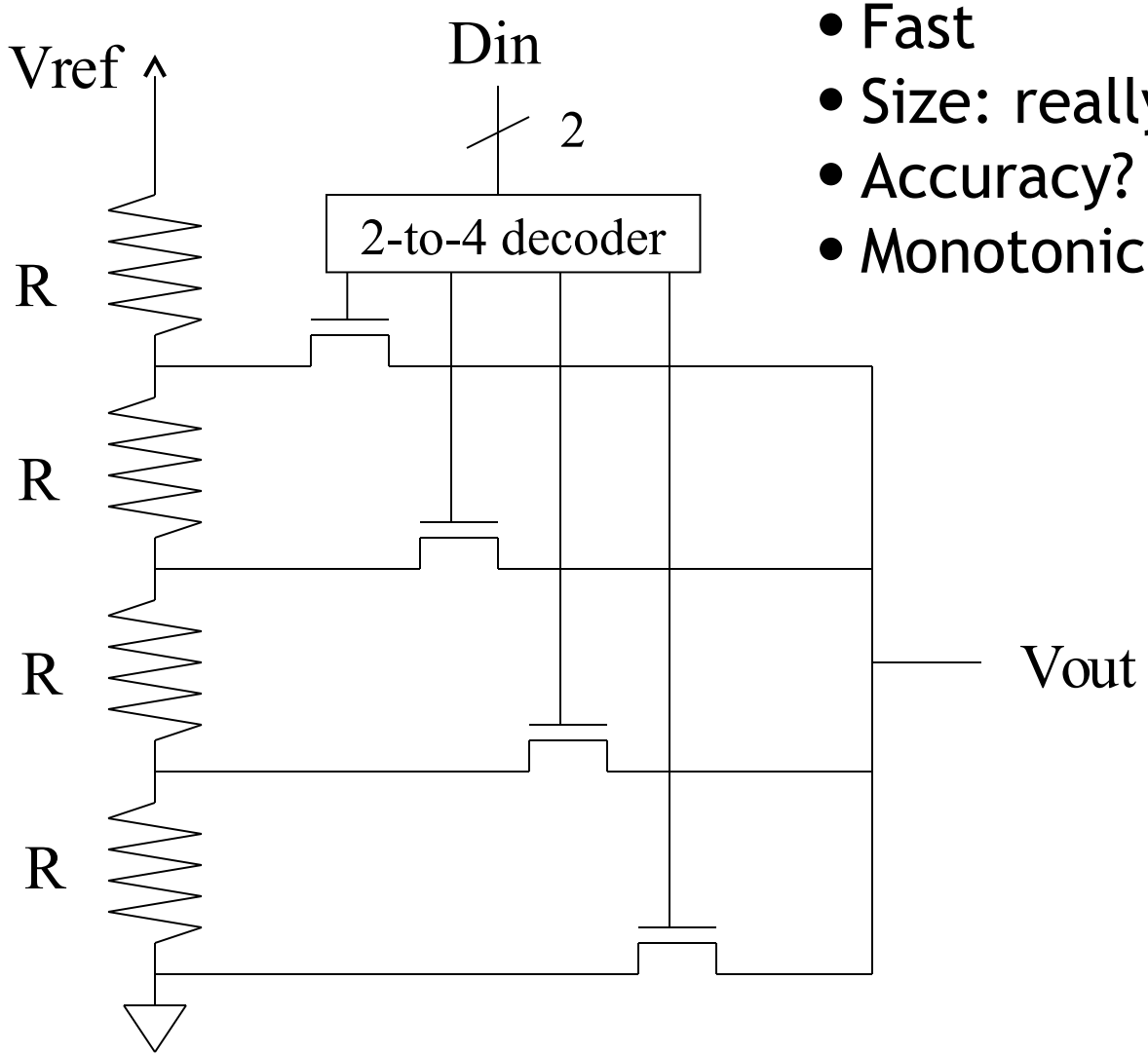
Lots of other issues

- 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.

Outline

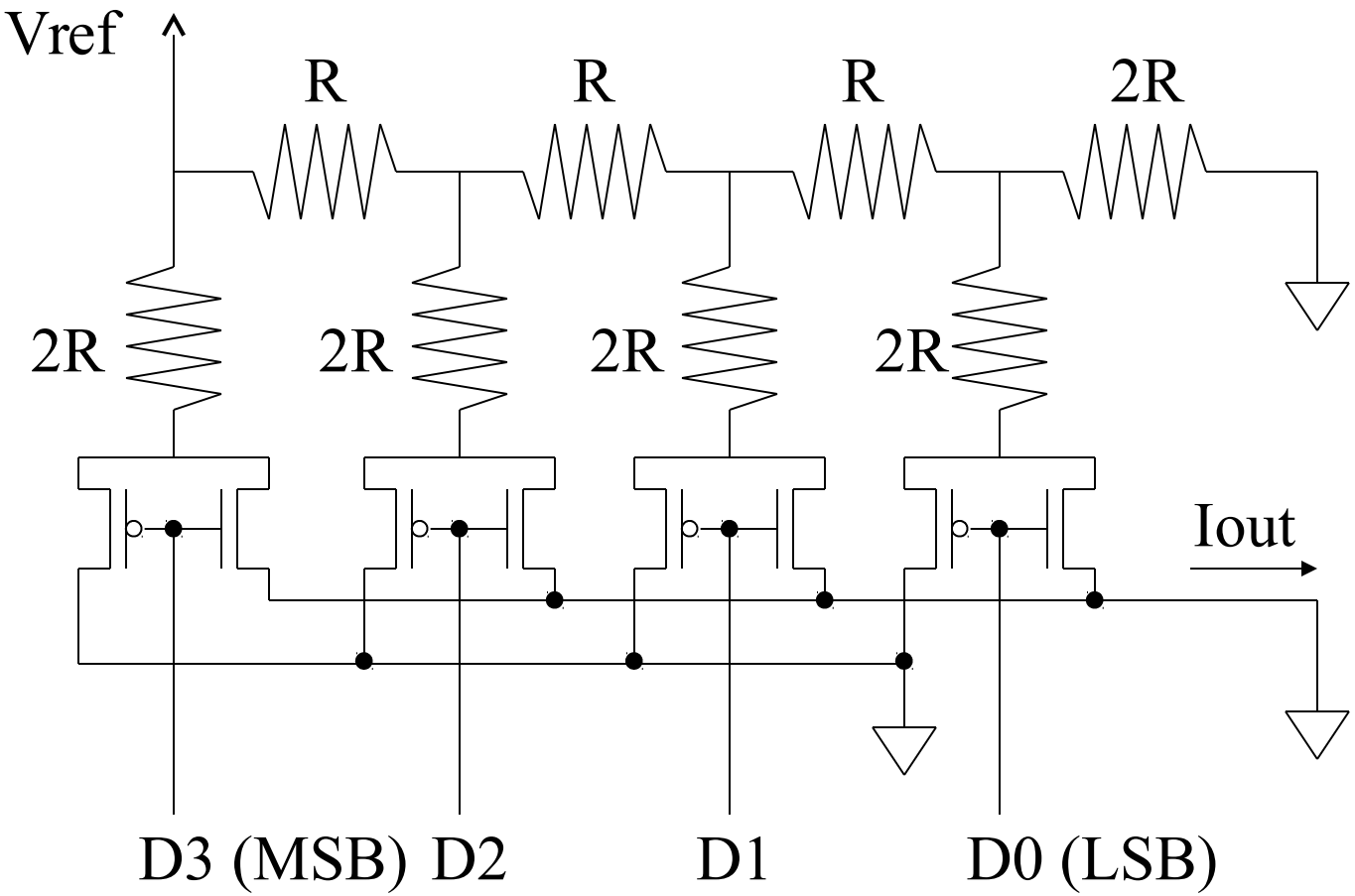
- ~~Context and review~~
- ~~Serial buses~~
  - ~~UART~~
  - ~~SPI~~
  - ~~I2C~~
- ADCs and DACs
  - ~~Fundamentals~~
  - Operation

DAC #1: voltage divider



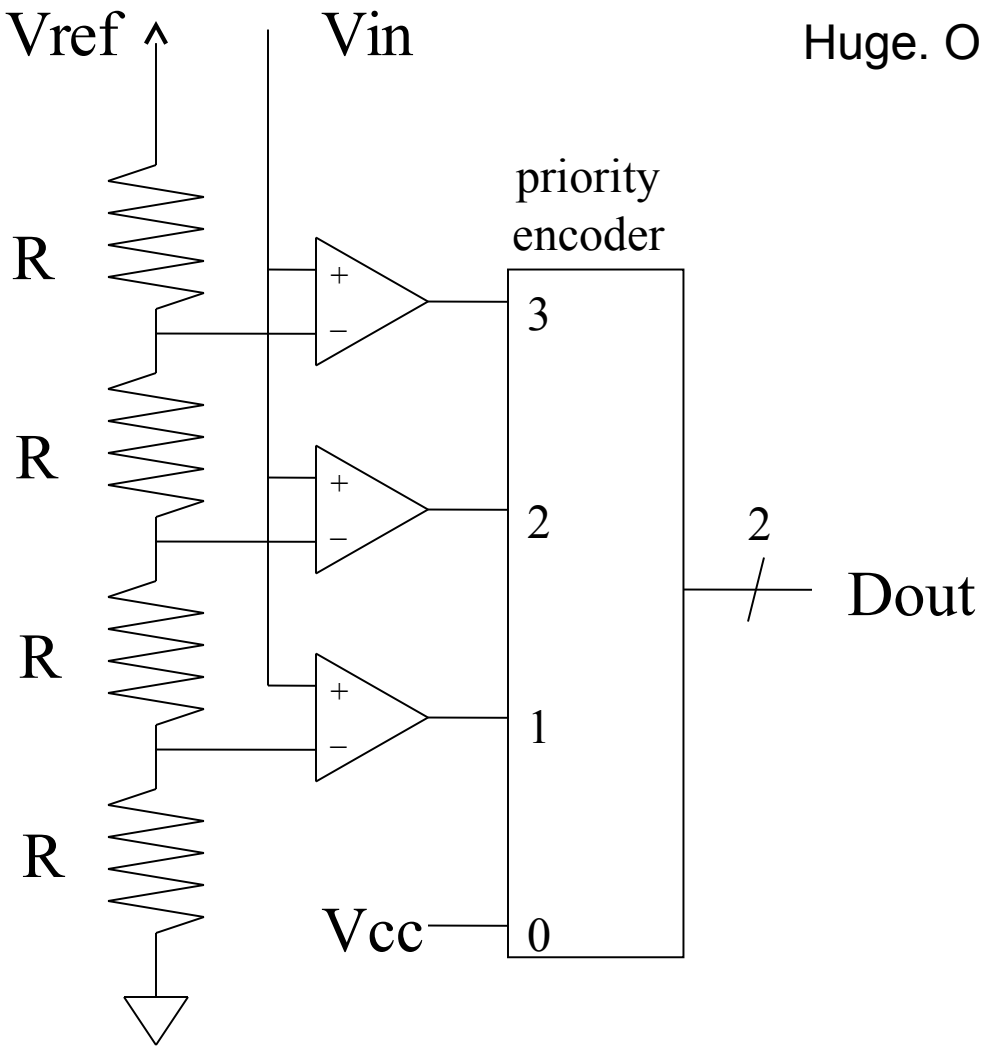
- Fast
- Size: really big: $O(2^n)$
- Accuracy?
- Monotonicity?

DAC #2: R/2R ladder



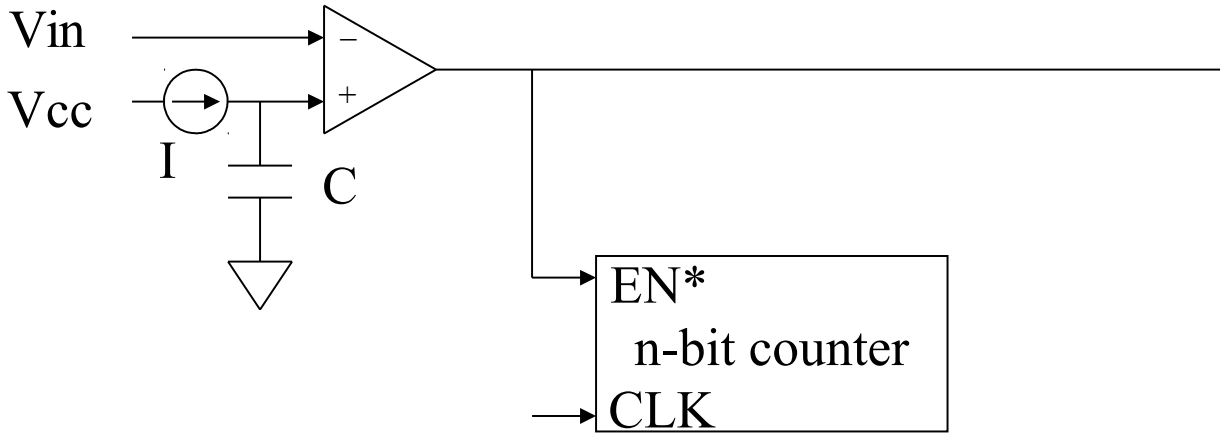
- Size: small O(n)
- Accuracy?
- Monotonicity? (Consider 0111 -> 1000)

ADC #1: flash



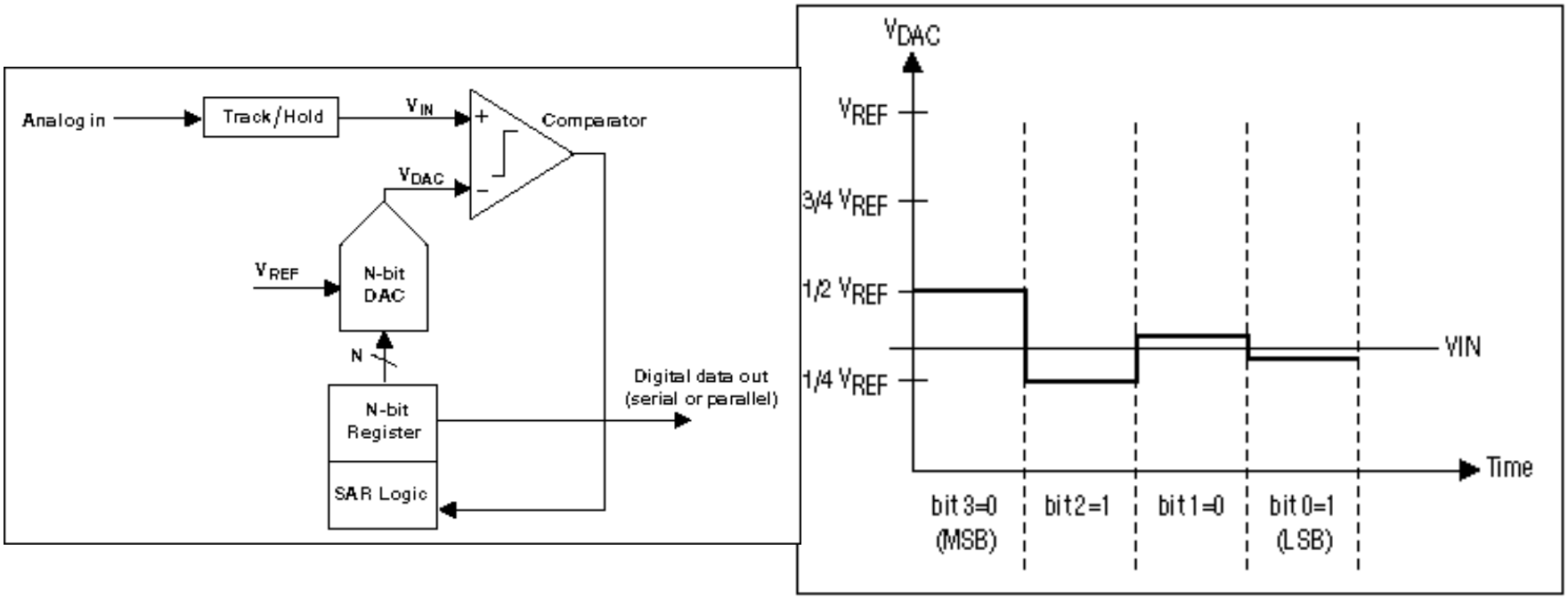
Huge. $O(2^n)$.

ADC #2: single-slope integration



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

ADC #3: successive approximation

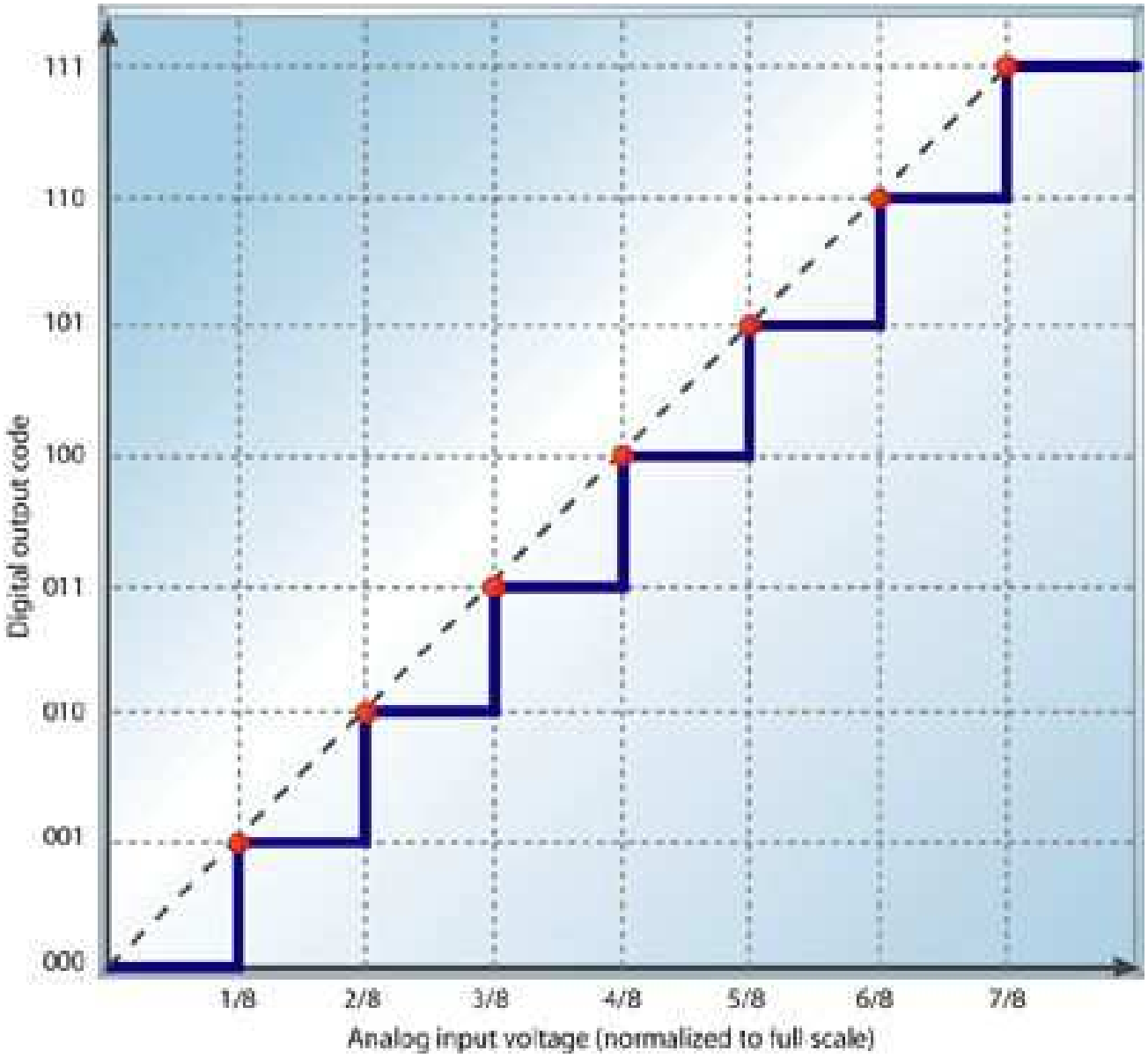


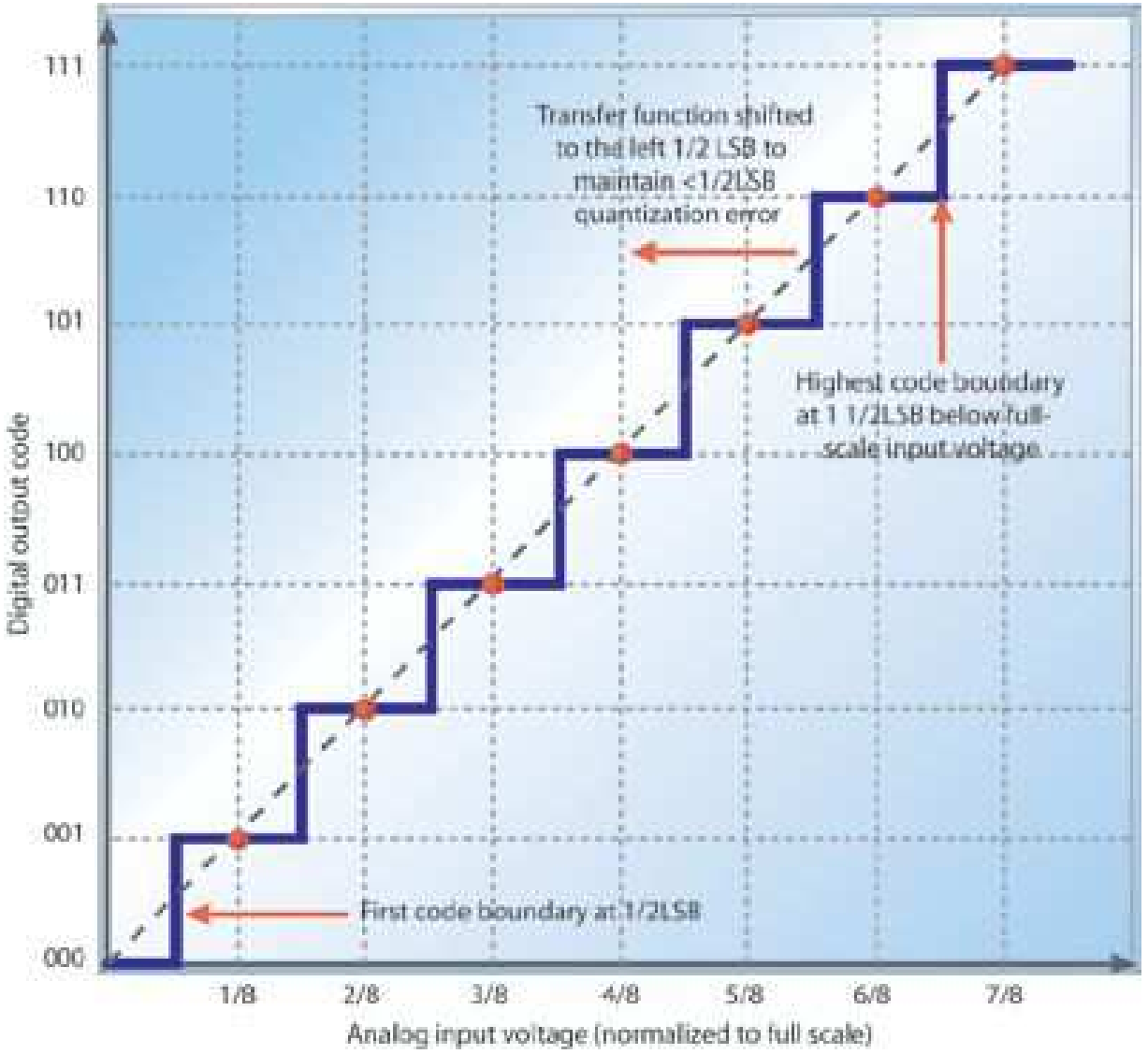
1 Sample → Multiple cycles

- Uses DAC for guessing.
- Faster: O(n)
- Goes from MSB to LSB.
- Not good for high-speed ADCs.

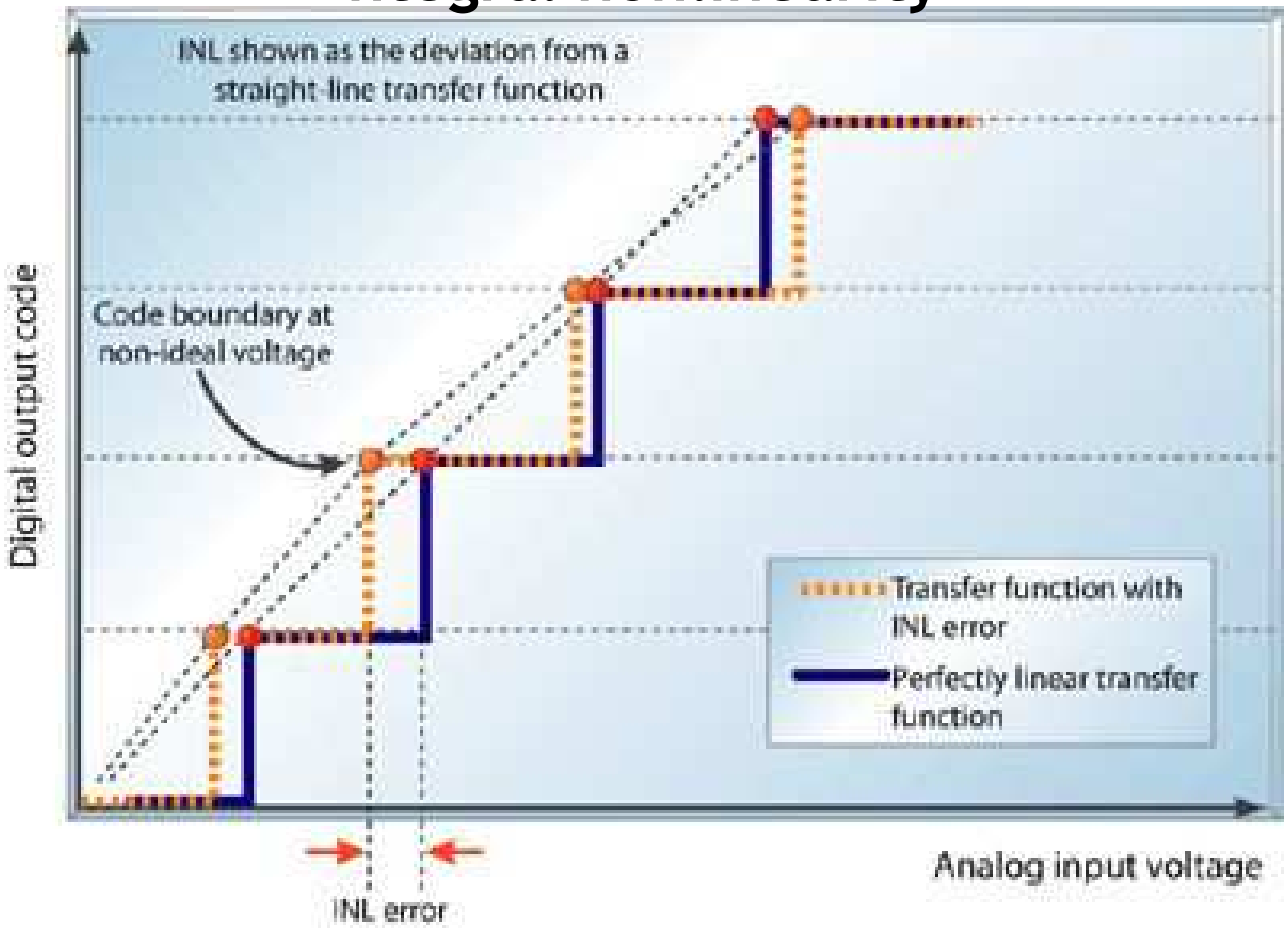
Errors and ADCs

- Figures and some text from:
  - Understanding analog to digital converter specifications. By Len Staller
  - http://www.embedded.com/showArticle.jhtml?articleID=60403334
- Key concept here is that the specification provides *worst case* values.


111
110
101
100
011
010
001
000
Digital output code
1/8
2/8
3/8
4/8
5/8
6/8
7/8
Analog input voltage (normalized to full scale)

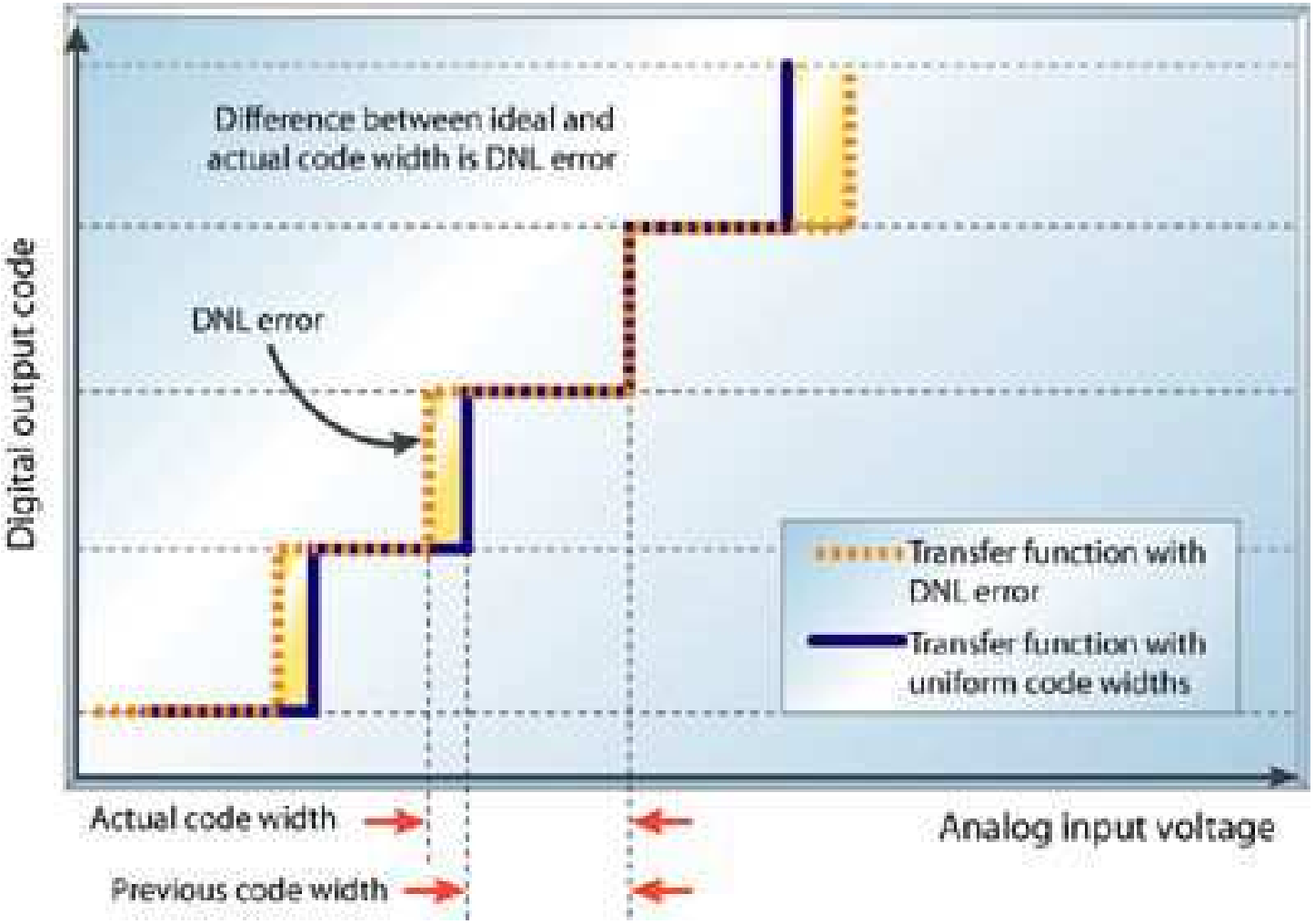

111
110
101
100
011
010
001
000
Digital output code
Transfer function shifted to the left 1/2 LSB to maintain <1/2LSB quantization error
Highest code boundary at 1 1/2LSB below full-scale input voltage
First code boundary at 1/2LSB
1/8
2/8
3/8
4/8
5/8
6/8
7/8
Analog input voltage (normalized to full scale)

Integral nonlinearity



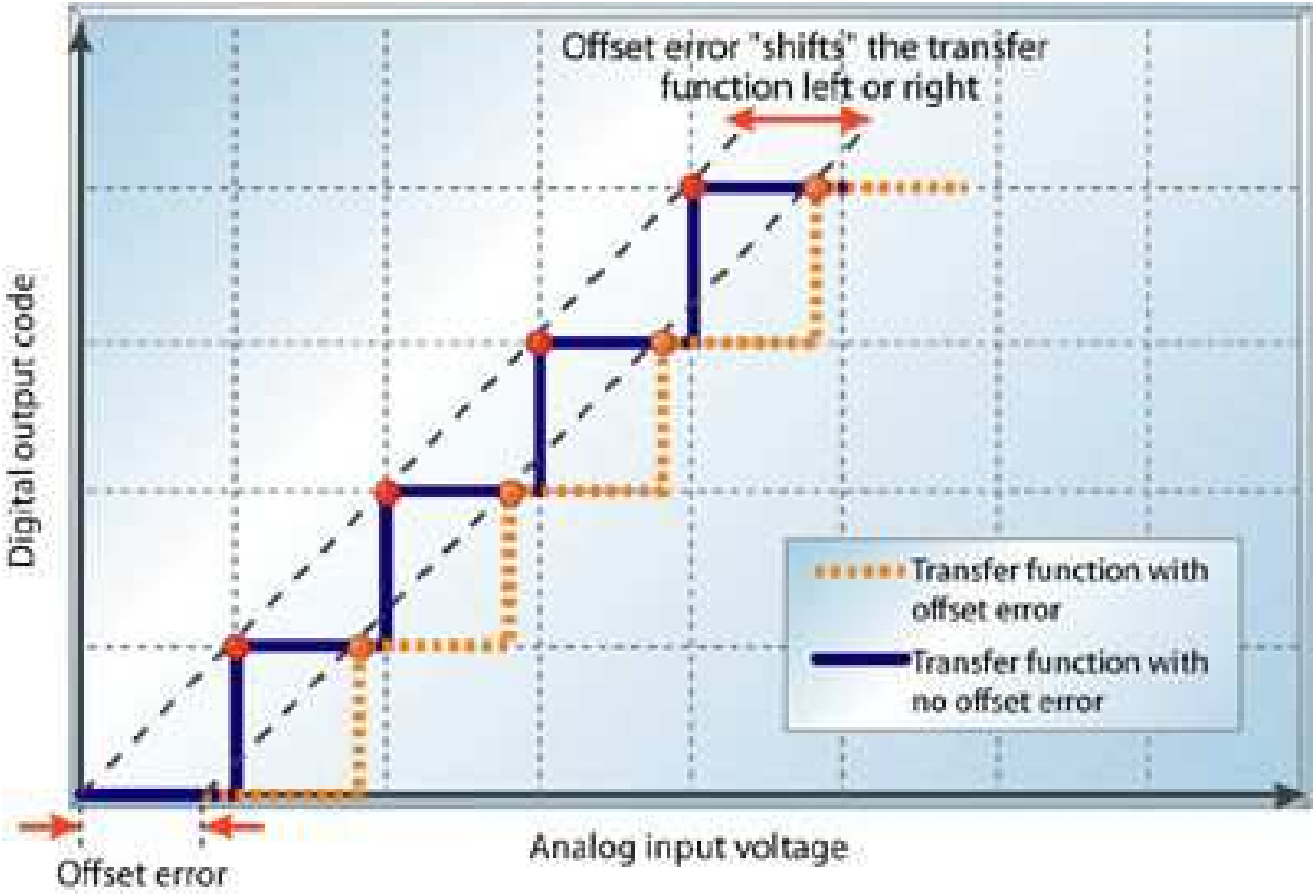
The *integral nonlinearity (INL) is the deviation of an ADC's transfer function from a* ***straight line***. This line is often a best-fit line among the points in the plot but can also be a line that connects the highest and lowest data points, or endpoints. INL is determined by measuring the voltage at which all code transitions occur and comparing them to the ideal. The difference between the ideal voltage levels at which code transitions occur and the actual voltage is the INL error, expressed in LSBs. INL error at any given point in an ADC's transfer function is the accumulation of all DNL errors of all previous (or lower) ADC codes, hence it's called *integral* nonlinearity.

Differential nonlinearity

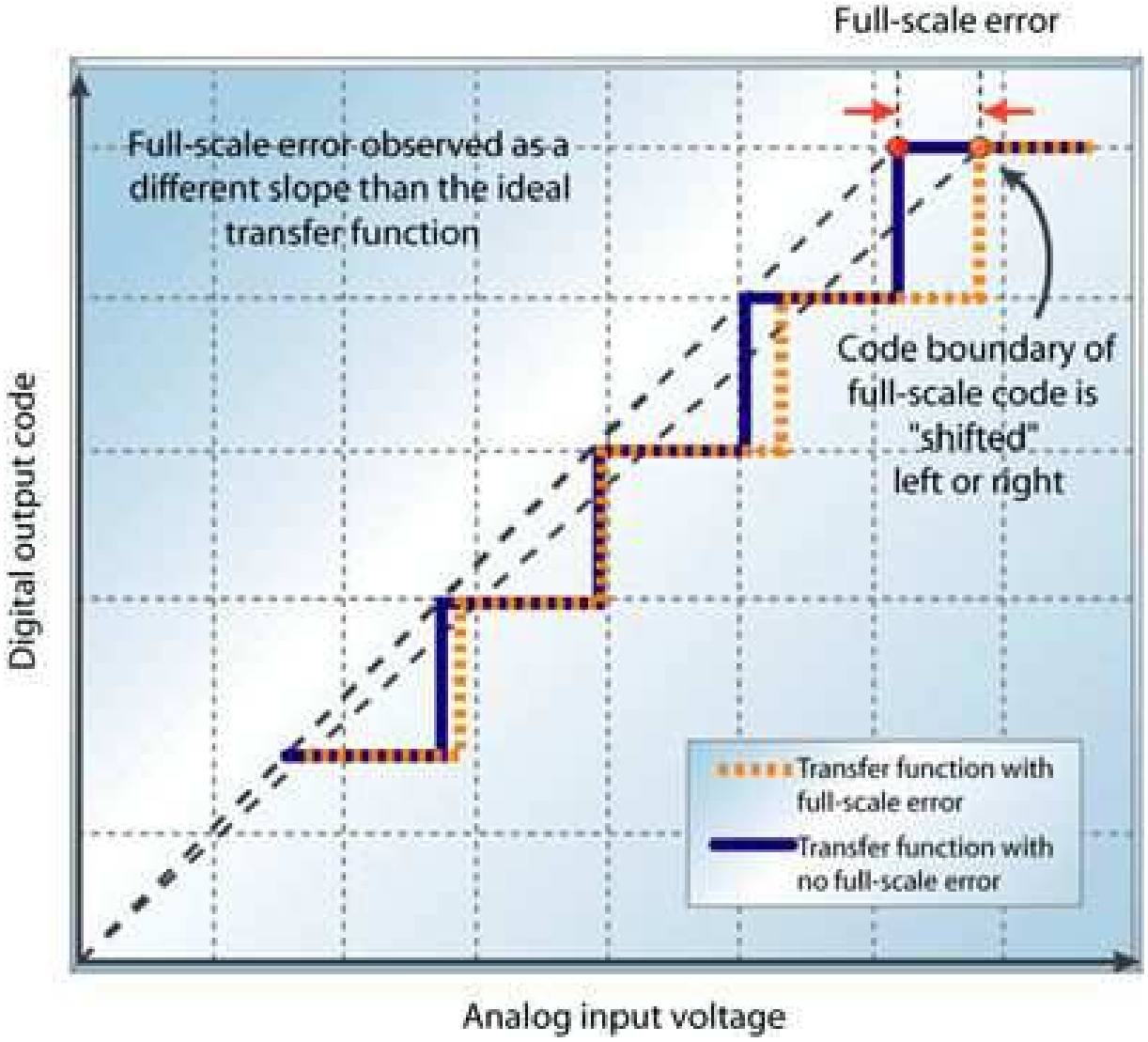


DNL is the worst cases variation of actual step size vs. ideal step size.

It's a promise it won't be worse than X.


Offset error "shifts" the transfer function left or right
Digital output code
Transfer function with offset error
Transfer function with no offset error
Analog input voltage
Offset error

Full-scale error is also sometimes called “gain error”



Full-scale error is the difference between the ideal code transition to the highest output code and the actual transition to the output code when the offset error is zero.

Errors

- Errors in a specification are bad.
  - So if you have an INL of ±0.25 LSB, you "know" that the device will never have more than 0.25 LSB error from its ideal value.
  - That of course assumes you are operating within the specification.
    - Temperature, input voltage, input current available, etc.
- Integral nonlinearity and differential nonlinearity are important.
  - Should know what full-scale error is.
  - Can compensate in software.
    - Where is best place to compensate?