Analog I/O

Ke Huang
Department of Electrical and Computer Engineering
San Diego State University

Announcement

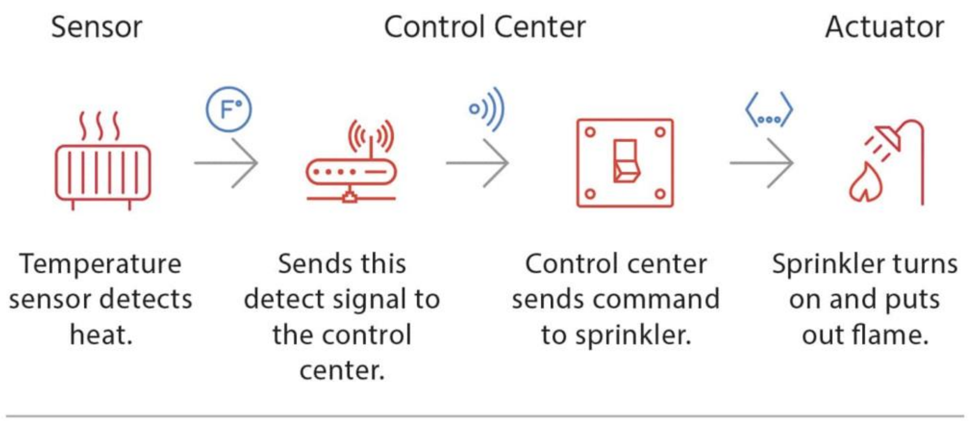
- The students who have a hold on their lab3 grades are required to resubmit a lab3 report with a corresponding late submission penalty, and with expectation that the similarity score will be less than 50%.
- For all future lab reports, any similarity score above 50% will result in a 0 in the corresponding lab grade and will trigger a plagiarism report to the university for further academic sanctions.

Outline

- Sampling
  - Nyquist theorem, aliasing
- Analog to Digital Conversion (ADC)
  - Quantization
  - Range, resolution, accuracy, error
  - ADC types
- Digital to Analog Conversion (DAC)
  - Examples, resistance networks
  - DAC types
  - DAC performance properties
- Noise and signal filtering
- Analog I/O on AVR

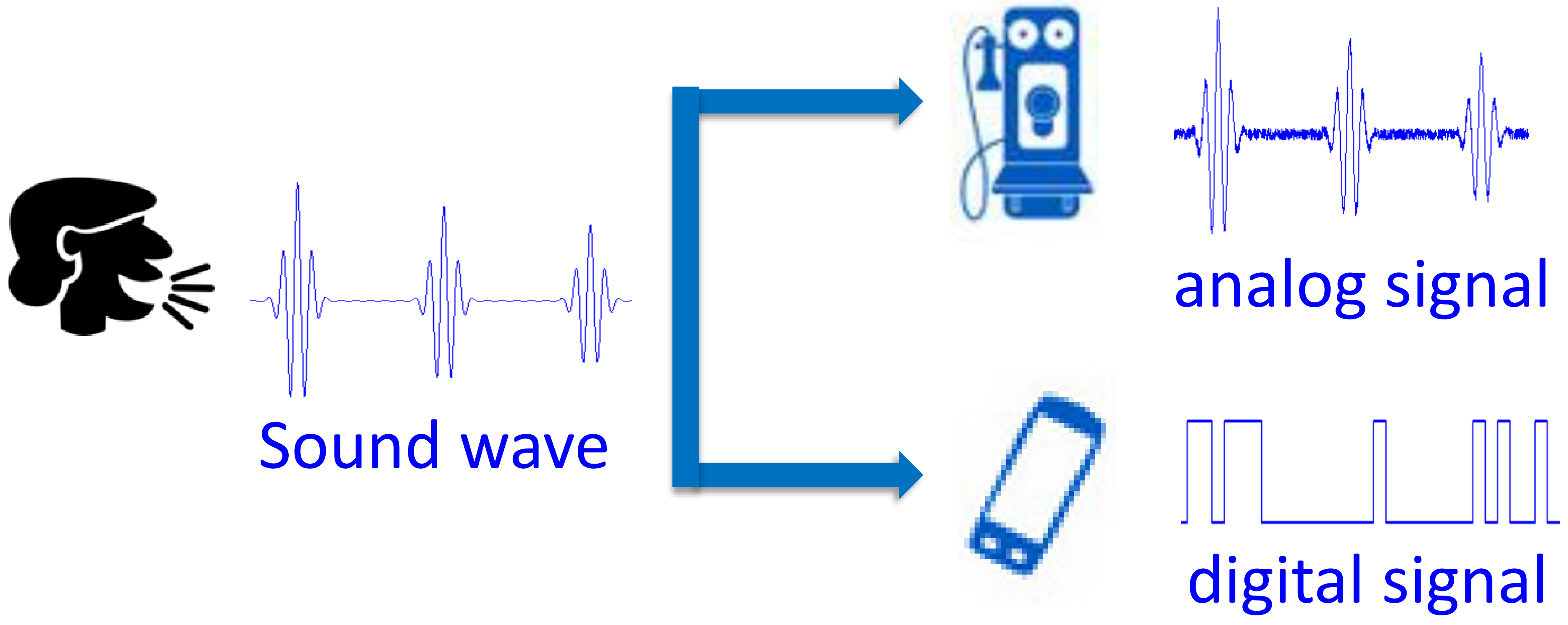
Sensors and Actuators

- Sensors
  - Capture physical stimulus
  - Convert it to electrical signal
- Actuators
  - Create physical stimulus
  - Given electrical signals
  - Examples:
    - Pneumatic systems, IR, thermal, motors, MEMS
- Need analog to digital and digital to analog converters
  - **Analog:** generate a voltage or current difference that must be measured and processed
    - e.g. ambient light
  - **Digital:** sensors directly generate a digital value
    - e.g. GPS


Sensor
Control Center
Actuator
Temperature sensor detects heat.
Sends this detect signal to the control center.
Control center sends command to sprinkler.
Sprinkler turns on and puts out flame.

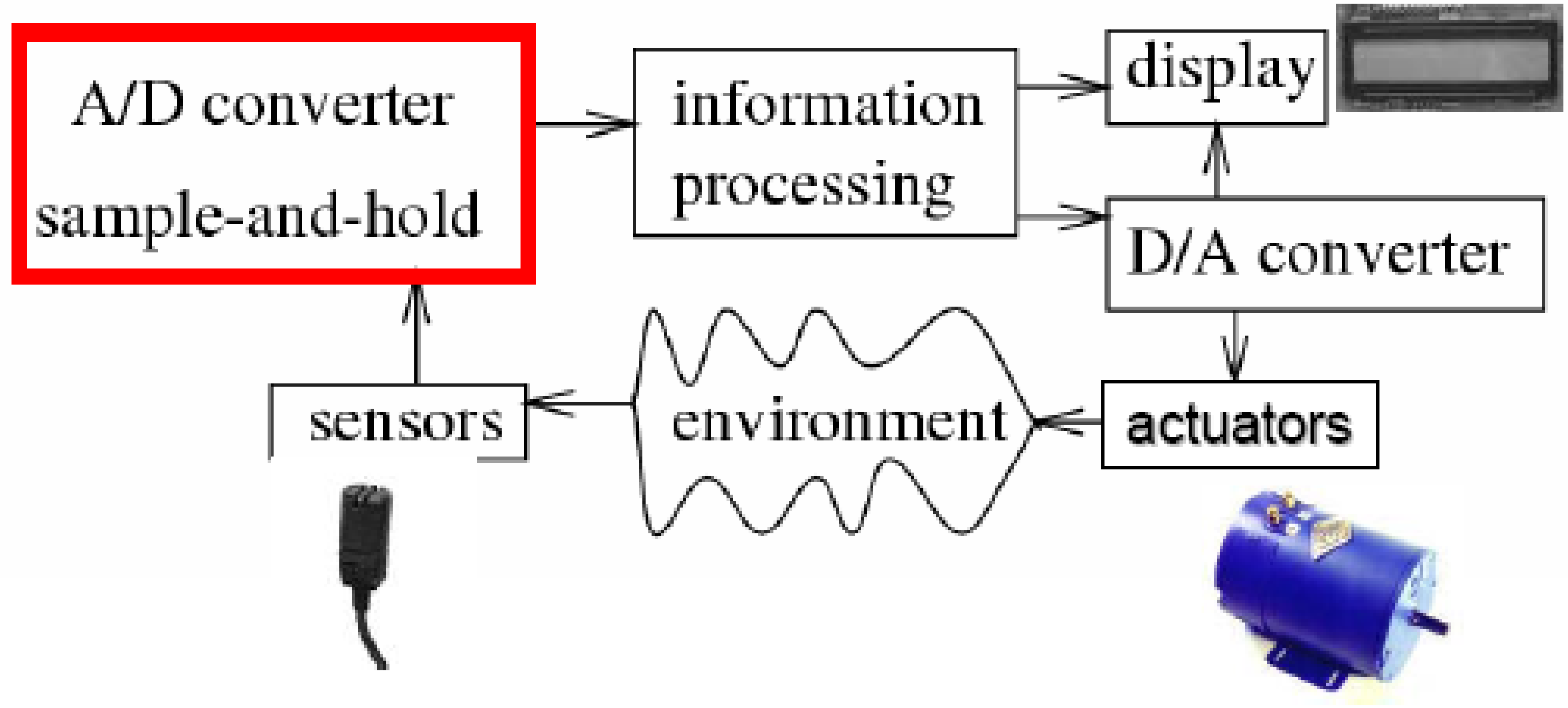
Sensor to **Actuator** Flow

Analog vs Digital Signal

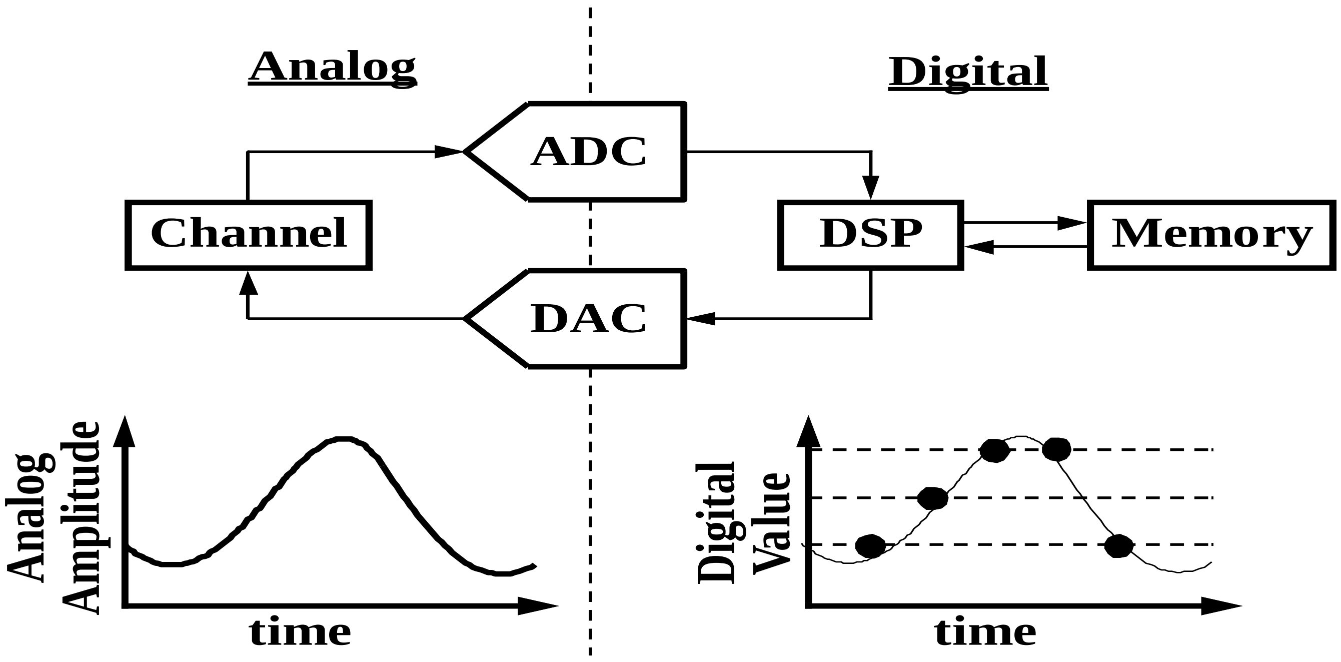


- Internal processing/programing in a microcontroller is always performed using digital signals, so in case we have analog input signals, we need to convert them to digital format.

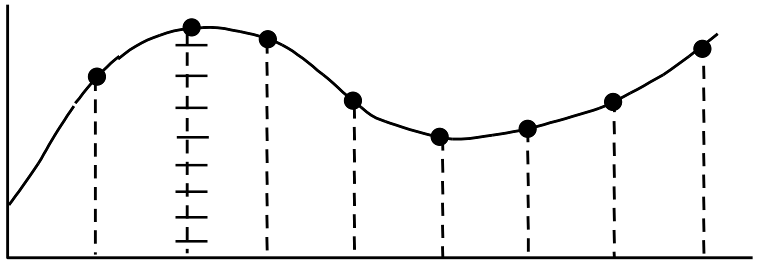
Embedded System Hardware Interfacing Sensors and Actuators



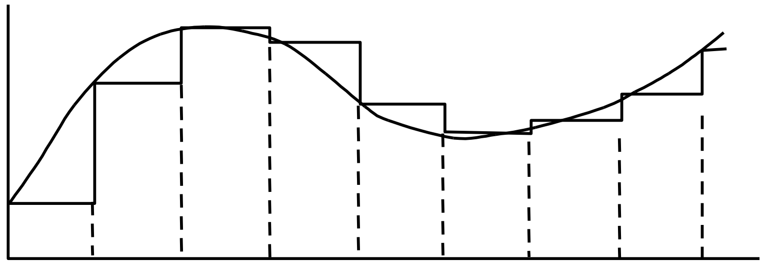
Real World Sampled Data Systems Consist of ADCs and DACs



ADC SAMPLED AND QUANTIZED WAVEFORM



DAC RECONSTRUCTED WAVEFORM

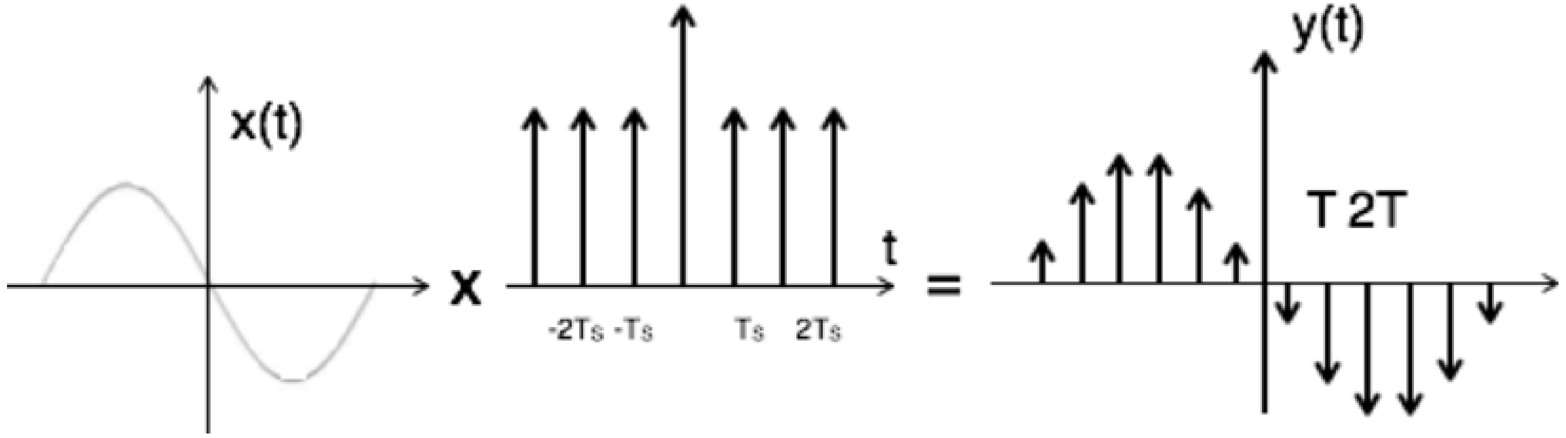


Signal Sampling

- Sampling converts a *continuous time* signal into a *discrete time* signal
  - It replicates spectrum of continuous-time signal at multiples of the sampling frequency
- Categories:
  - Impulse (ideal) sampling (infinitely narrow pulse)
  - Natural sampling (finite width pulse)
  - Sample and hold circuit

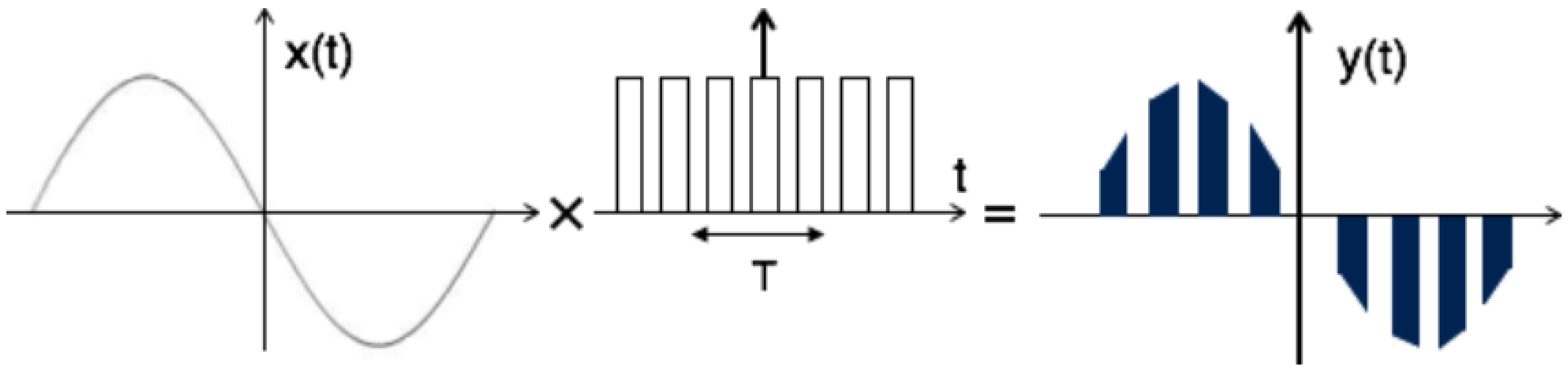
Impulse Sampling

- Impulse sampling can be performed by multiplying input signal x(t) with impulse train $\sum_{n=-\infty}^{\infty} \delta(t - nT)$ of period T. Here, the amplitude of impulse changes with respect to amplitude of input signal $x(t)$. The output of sampler is given by



Natural Sampling (Finite Width Pulse)

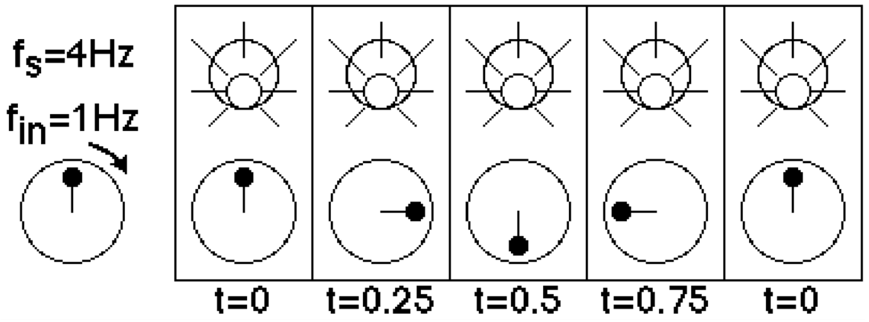
- Natural sampling is similar to impulse sampling, except the impulse train is replaced by pulse train of period T. i.e. you multiply input signal x(t) to pulse train $\sum_{n=-\infty}^{\infty} P(t-nT)$ as shown below

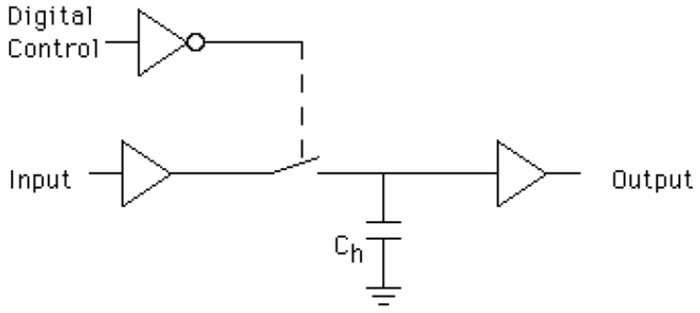


Sampled Signals

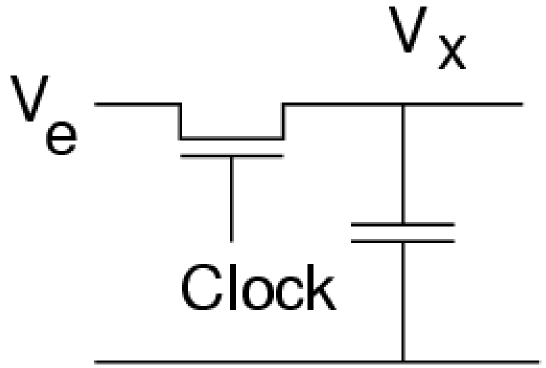
- Samples
- Ideally instantaneous
- Strobe light flashing
  - Strobe illuminates the spot on a rotating disk:

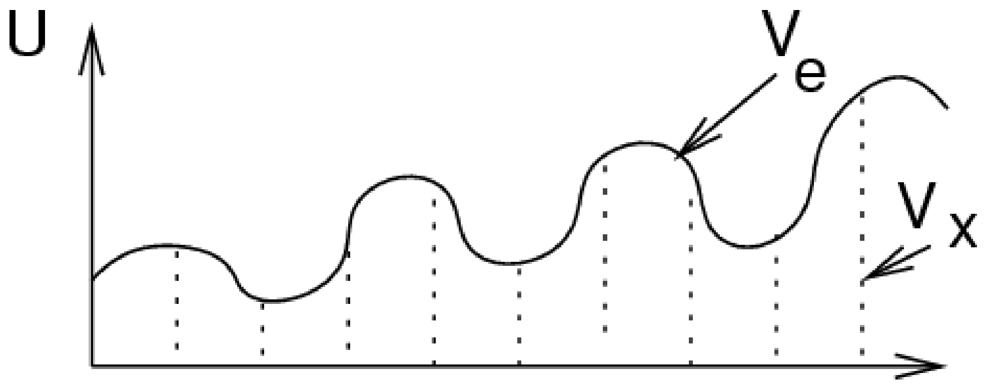
- Sampling yields
  - Discrete samples
  - A subset of reality
  - But still continuous in amplitude





Sample and Hold



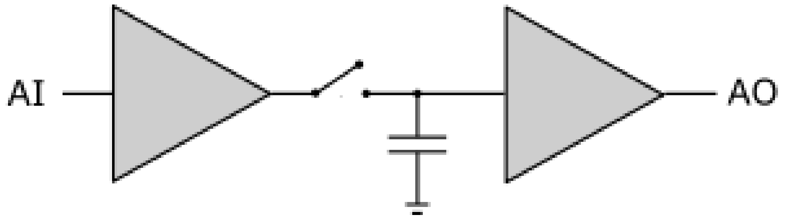


V_e is analog input signal

V_x is digital sampled signal

Sample-And-Hold Circuit

S/H



Analog Input (AI) is sampled when the switch is closed and its value is *held* on the capacitor where it becomes the Analog Output (AO)

Sampling Methods Examples

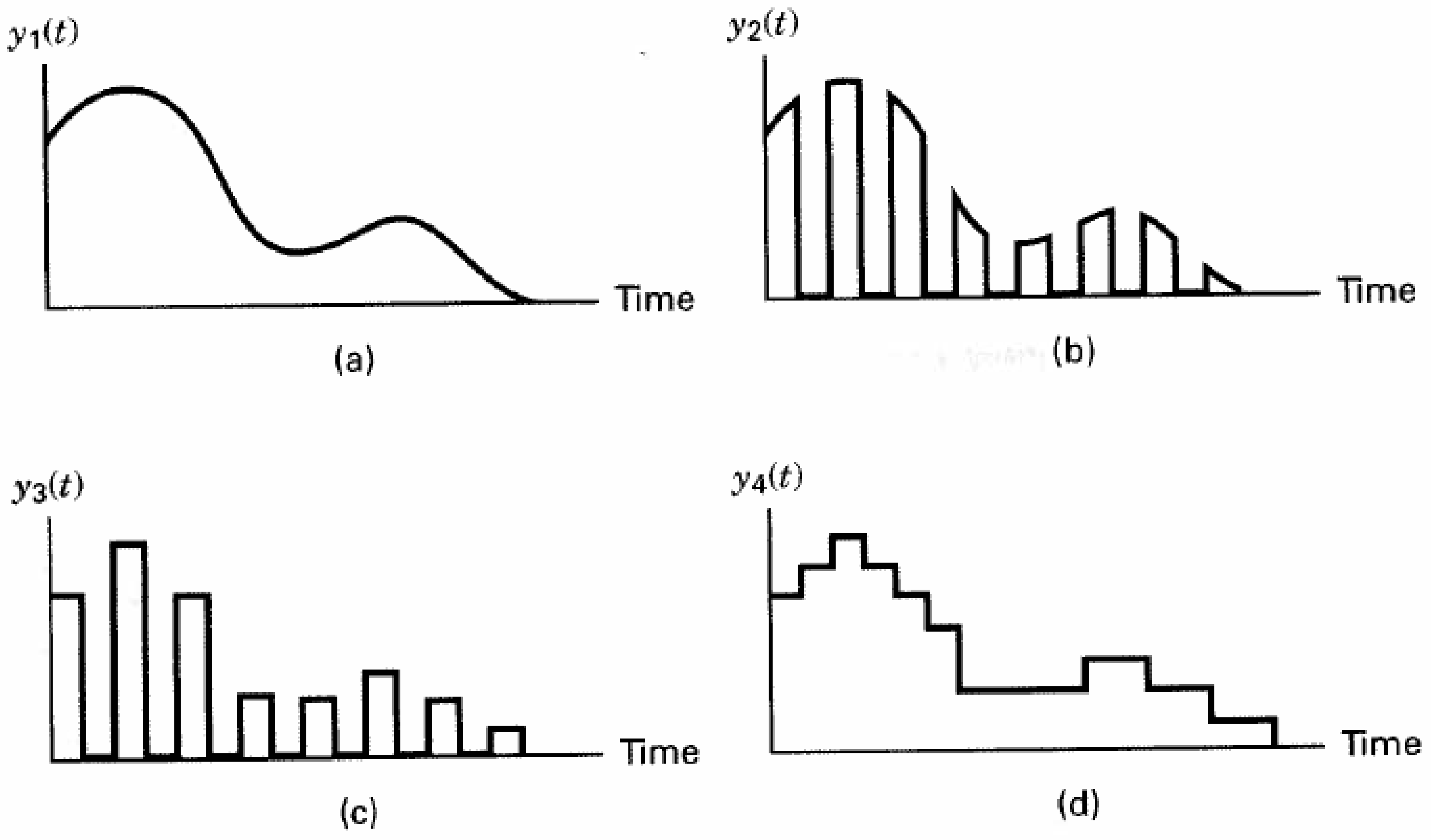
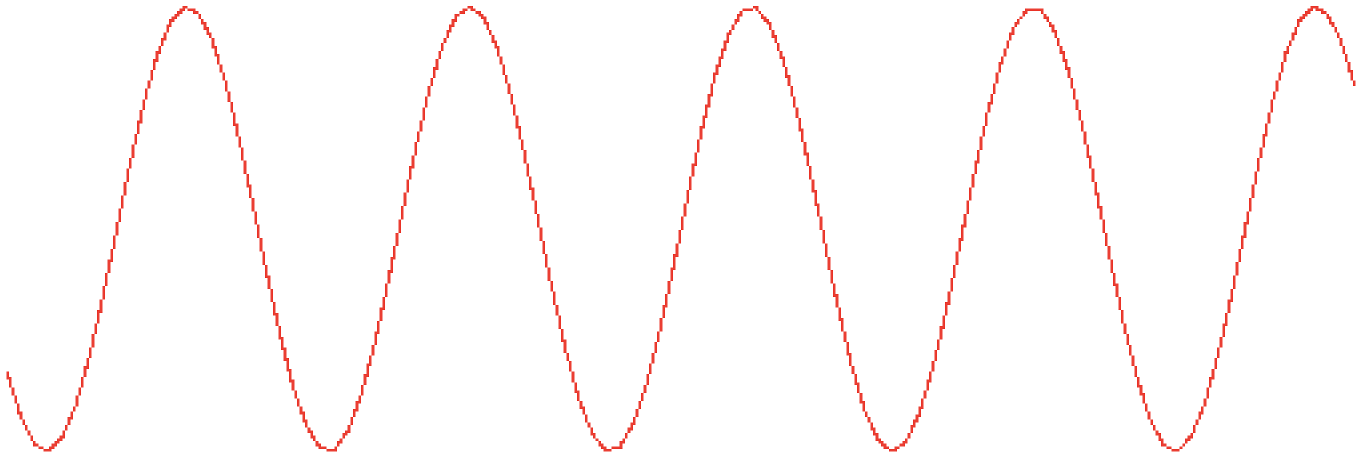


Figure 2.14 Amplitude and time coordinates of source data. (a) Original analog waveform. (b) Natural-sampled data. (c) Quantized samples. (d) Sample and hold.

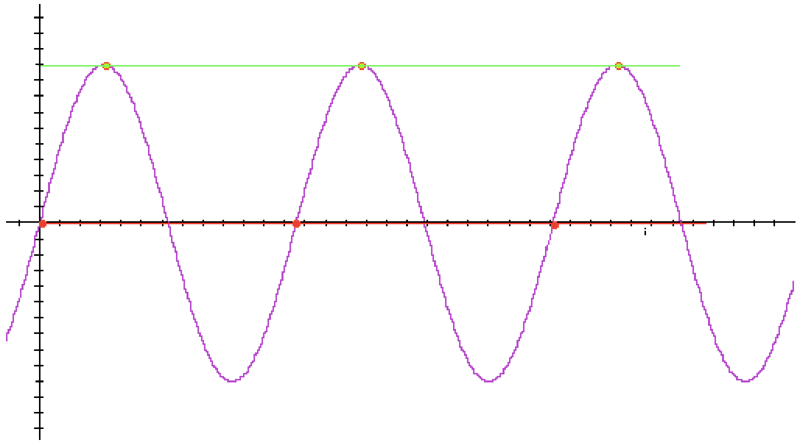
How fast should you sample? – Does it matter?

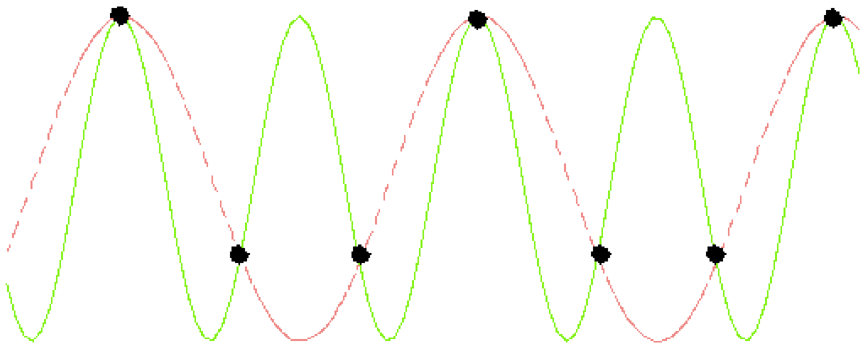
- Suppose that we want to sample a simple Sine wave (shown below)



How fast should you sample? – Does it matter?

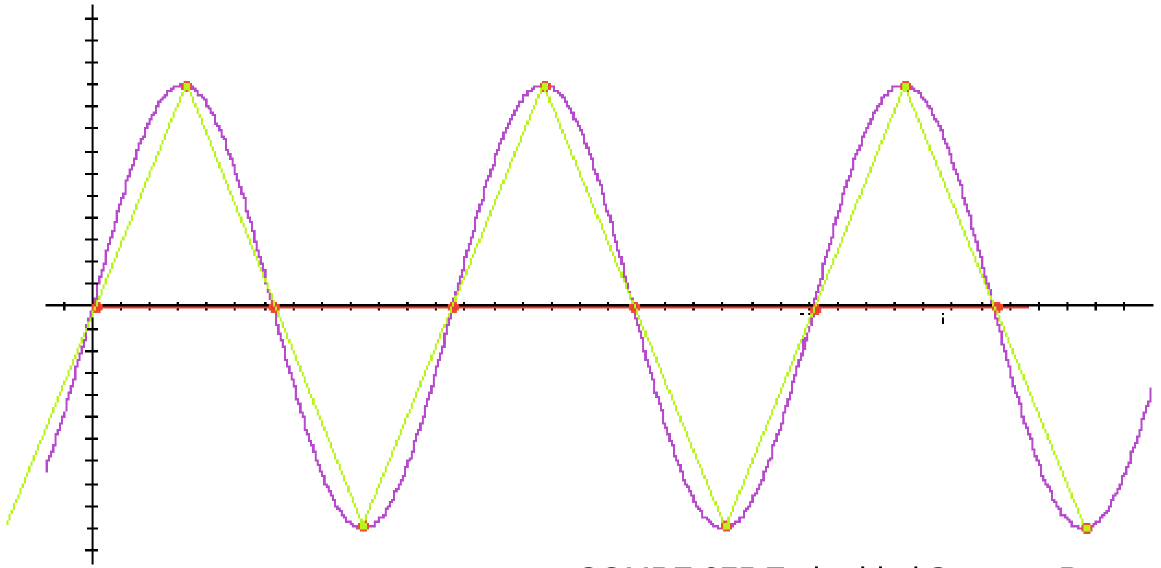
- If we sample once per cycle, we might think that it is constant
- If we sample at 1.5 times per cycle, we can think it's a lower frequency sine wave





How fast should you sample? – Does it matter?

- Now if we sample at twice the sample frequency, we start to make some progress
- An alternative way of viewing the waveform (re)generation is to think of straight lines joining up the peaks of the samples
  - In this case, we would see we get a saw-tooth wave that begins to start crudely approximating a sine wave



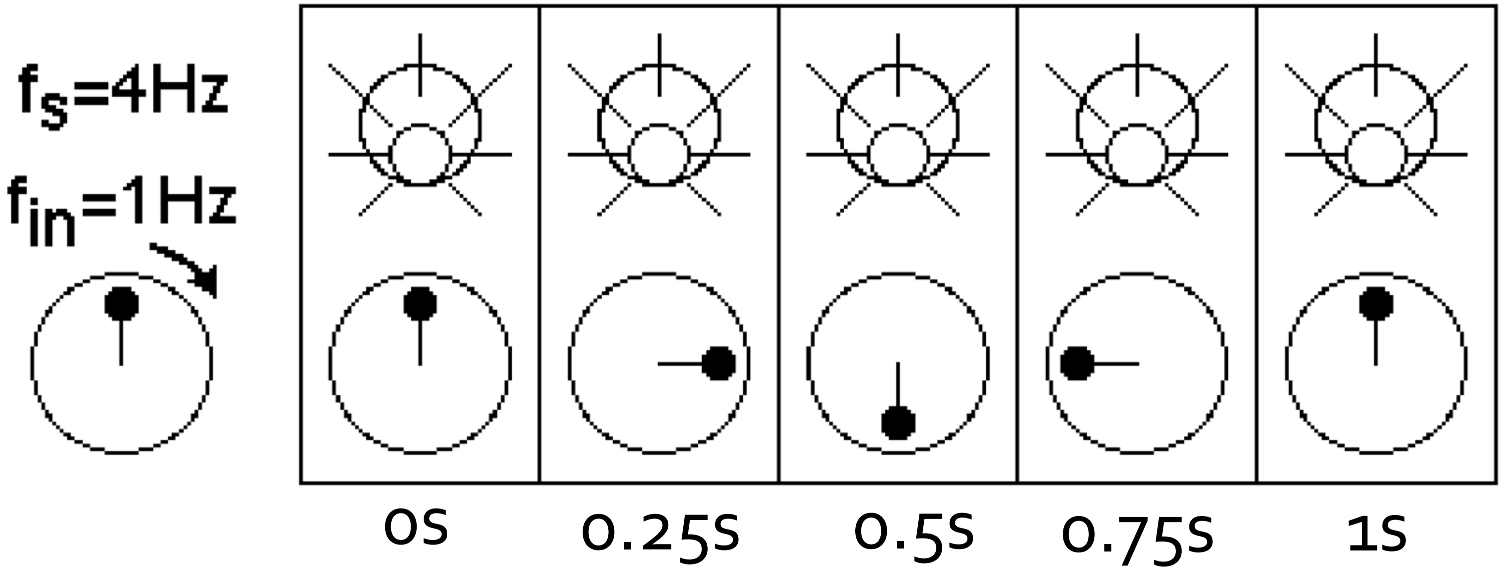
Nyquist rate -- For lossless digitization, the sampling rate should be at least twice the maximum frequency responses. Indeed many times more the better

Nyquist Theorem

- Sampling Frequency = f_s
- Input Frequency = f_{in}
- **Nyquist criterion:**
  - $f_s >= 2 * f_{in}$
- $f_s < 2 * f_{in}$ results in Aliasing
- "Nyquist Frequency" = $f_s/2$
- Aliasing is when $f_{apparent} < f_{in}$
  - Aliasing happens when the observed frequency is less than the actual one
  - $f_{apparent} = f_{in}$ when Nyquist criterion is satisfied
  - Generally oversampling is done → $f_s > 2 * f_{in}$

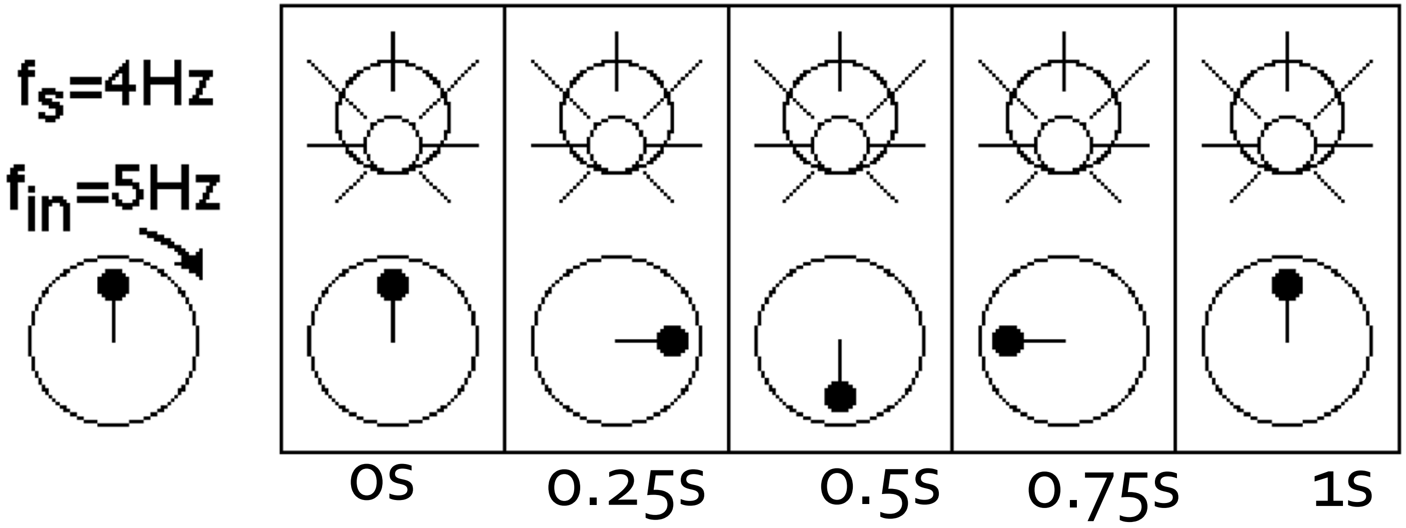
Sampling Example

- Disk rotates once/second, light flashes 4/sec
- Four samples per rotation:
  - $f_s > 2*f_{in}$, so it meets Nyquist Criteria => No Aliasing



Aliasing

- Aliasing occurs when $f_s < 2*f_{in}$



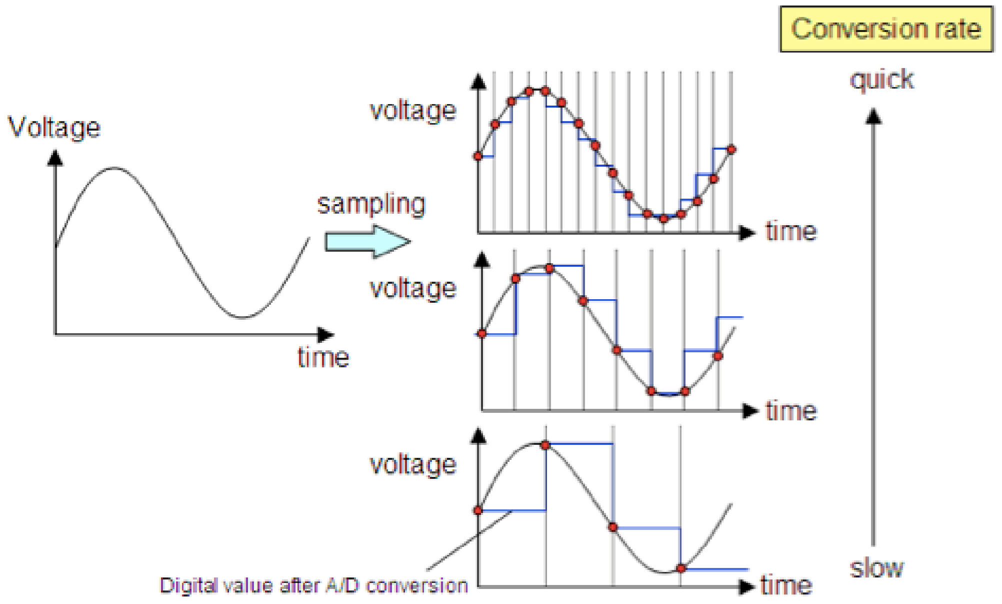
- Apparent frequency is 1Hz, not 5Hz

Outline

- Sampling
  - Nyquist theorem, aliasing
- **Analog to Digital Conversion (ADC)**
  - **Quantization**
  - **Range, resolution, accuracy, error**
  - **ADC types**
- Digital to Analog Conversion (DAC)
  - Examples, resistance networks
  - DAC types
  - DAC performance properties
- Noise and signal filtering
- Analog I/O on AVR

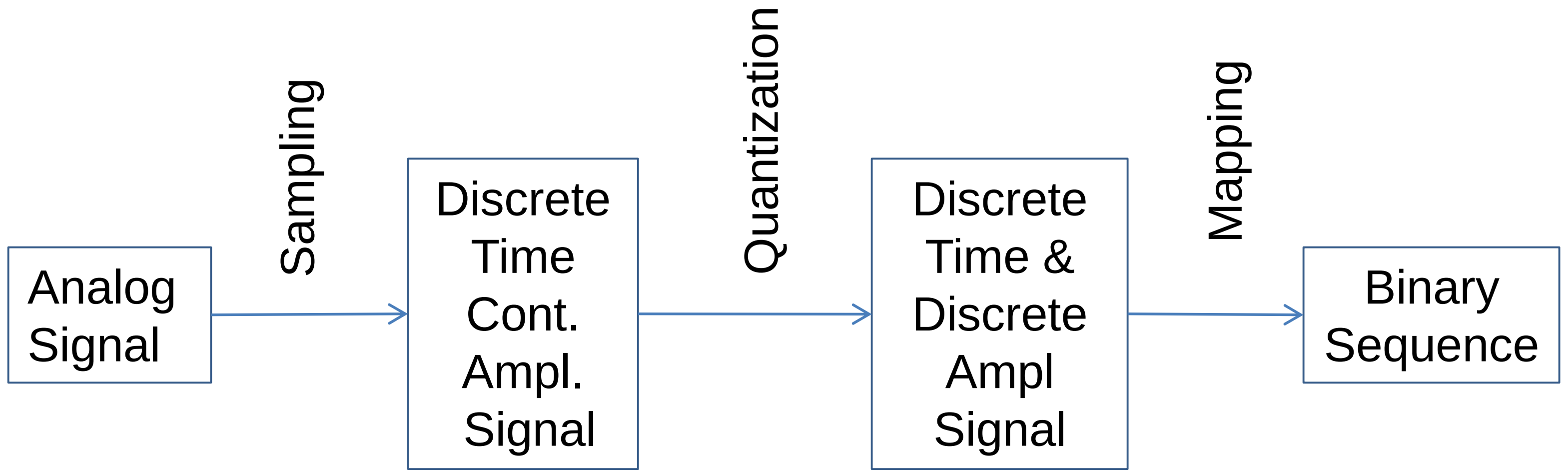
Analog to Digital Converters (ADC)

- Analog signals are continuous
  - A discrete version of the signal is created by “sampling”
  - ADC maps each sample onto a quantized range of voltages that can be represented by binary values



Quantization

- Quantization is done to make the signal amplitude discrete


Analog Signal
Sampling
Discrete Time Cont. Ampl. Signal
Quantization
Discrete Time & Discrete Ampl Signal
Mapping
Binary Sequence

Range, Resolution & Accuracy

- **Range**: Extremes that can be represented
  - E.g.: -32768 to +32767, -5.12V to +5.11V
- **Resolution**: Number of bits or the smallest difference that can be represented
  - E.g.: 1 mV, 1 LSB, or 0.1% of full scale
- **Accuracy**: How close is converted value relative to (deviates from) correct value
  - E.g.: +/-2 LSBs, 1/2 LSB, 1% of full scale

Range, Resolution & Accuracy

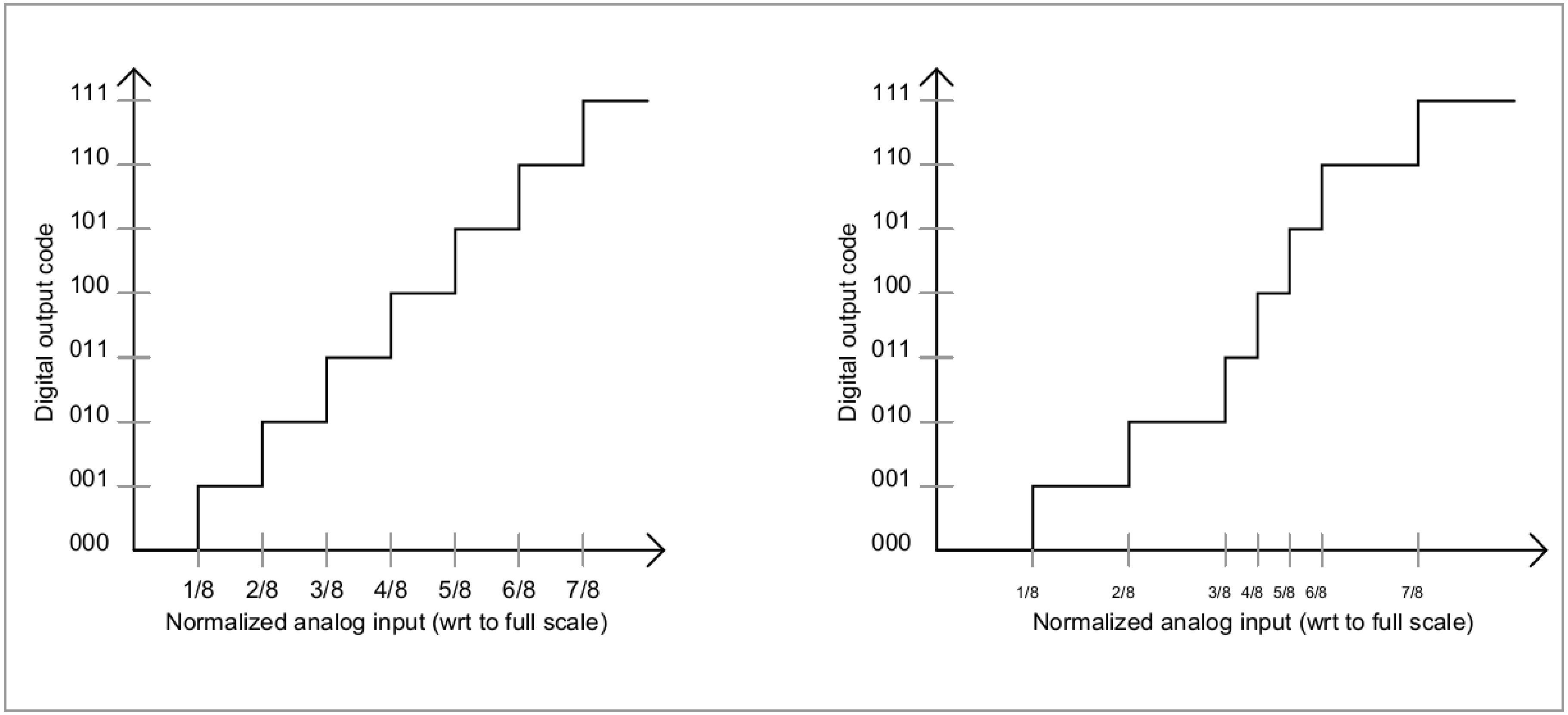
- An ADC partitions a conversion **range** (V_{pp}) into $(2^n - 1)$ **quantization steps (q)**. It is also referred to as a **least significant bit (LSB)**; therefore, an LSB is:

$$\text{LSB} = \frac{V_{pp}}{2^n - 1}$$

where n is the bit length. For a V_{pp} of 1V and an n of 10, 1 LSB is approximately 1mV.

Quantization Types

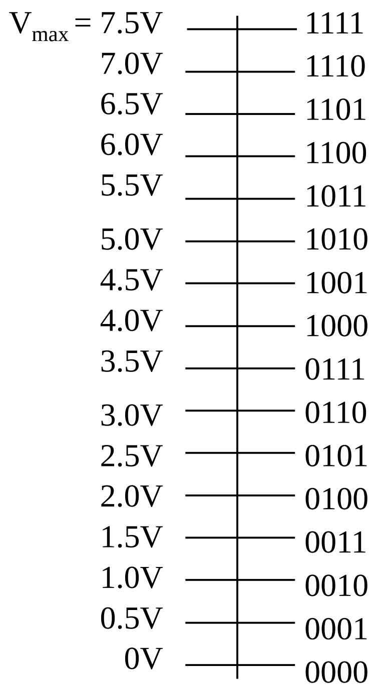
- Linear (uniform) vs. non-uniform



uniform quantization

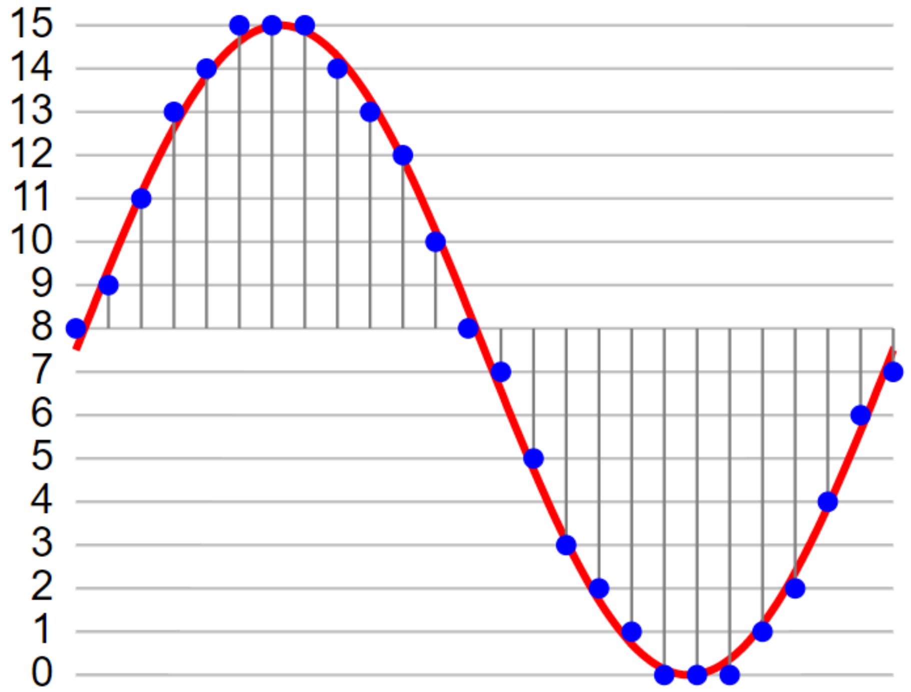
non-uniform quantization

Linear Pulse Code Modulation



V_{max} = 7.5V	1111
7.0V	1110
6.5V	1101
6.0V	1100
5.5V	1011
5.0V	1010
4.5V	1001
4.0V	1000
3.5V	0111
3.0V	0110
2.5V	0101
2.0V	0100
1.5V	0011
1.0V	0010
0.5V	0001
0V	0000

proportionality



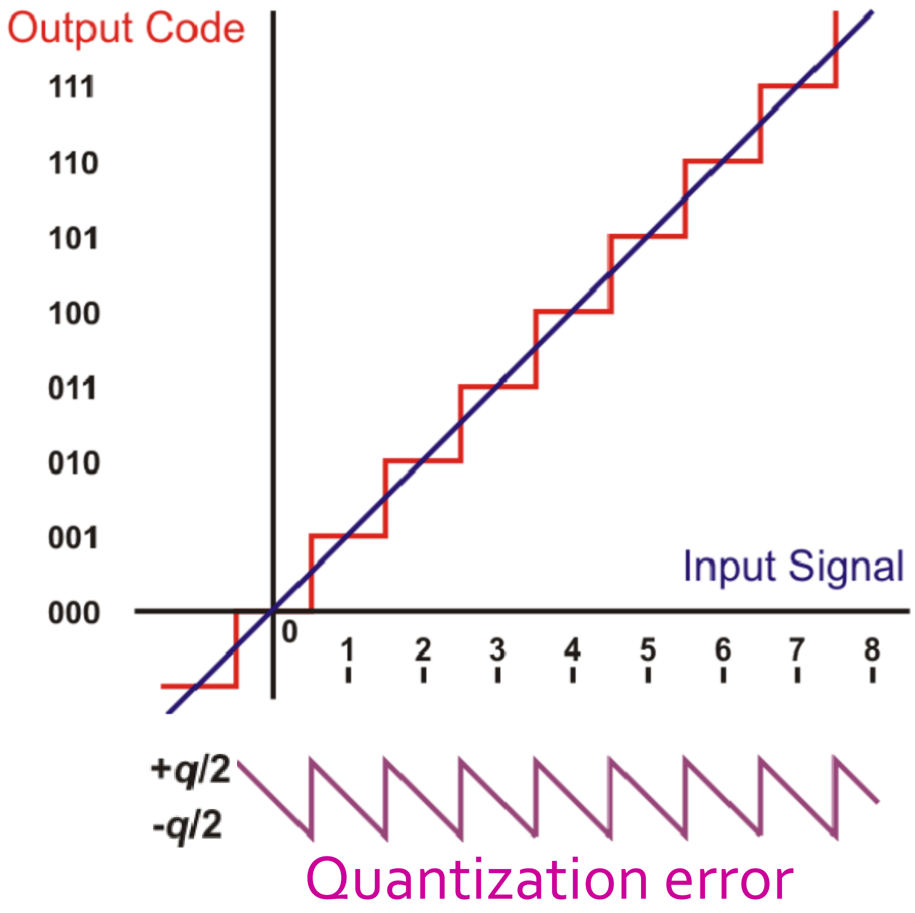
Pulse Code Modulation is a method used to digitally represent sampled analog signals

Linear PCM:

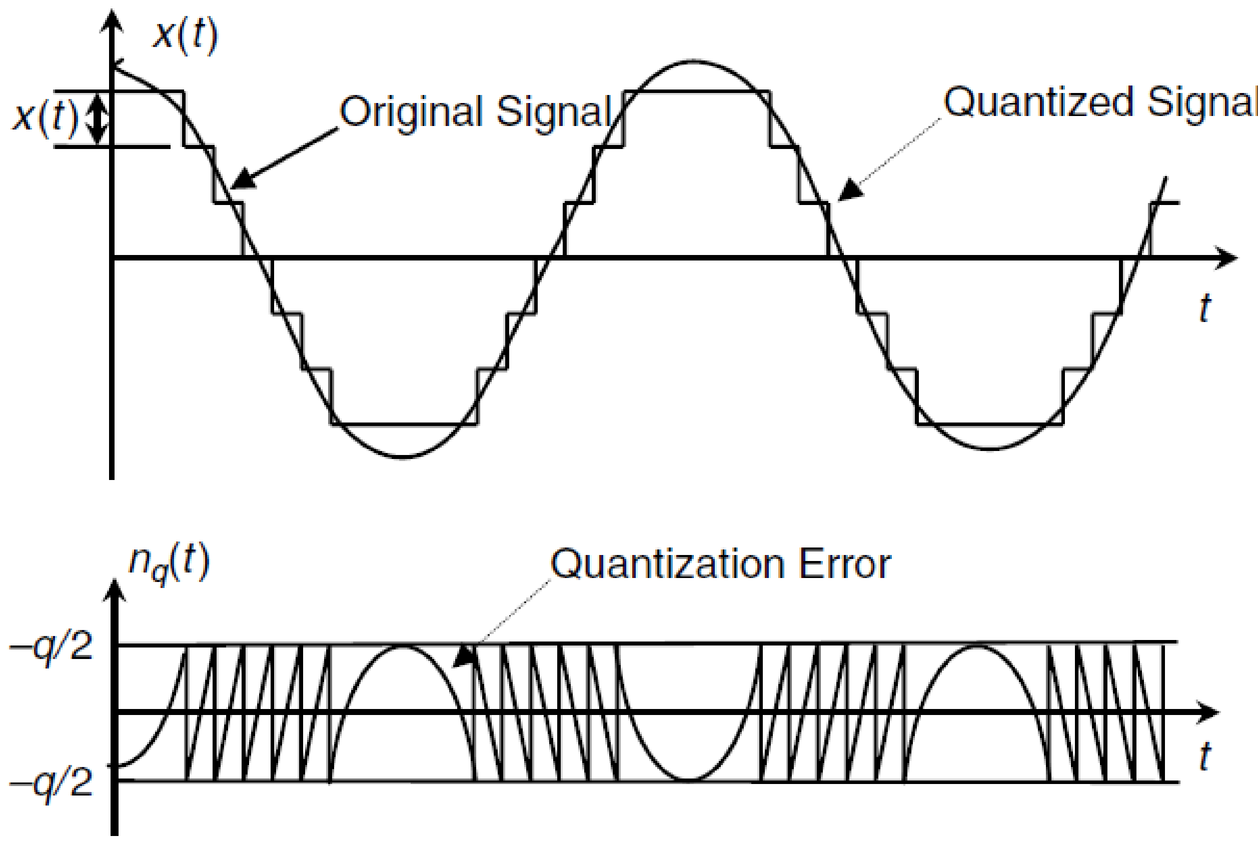
- Quantization levels are uniform (linear)
- Defined by a sampling rate & bit depth L (total # of values that can be represented)

Quantization Error (Linear)

- Error is zero at the predefined levels
- Error increases up to the next predefined level and becomes zero



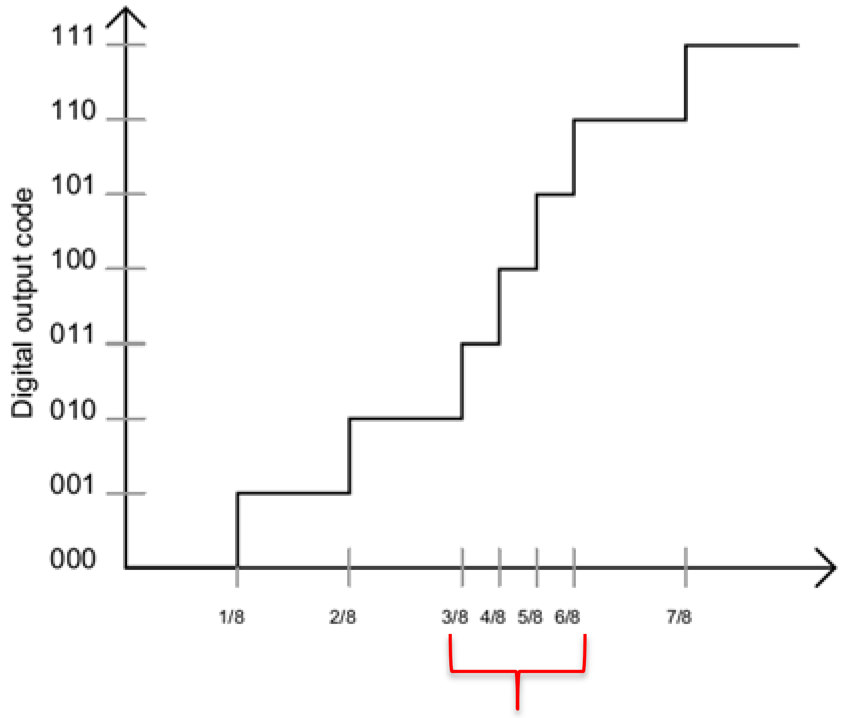
Quantization Error Example



Non-uniform Quantization

- Ways to implement non-uniform quantization:
  - Use an amplifier with non linear gain and apply resultant signal to uniform quantizer
    - This technique is commonly termed as companding
  - Adjust the ADC quantization levels directly
    - For example the threshold values can be varied in a flash converter by varying the resistor string
  - Have a uniform quantizer and use a look up table approach to generate non uniform quantized values
- One common application of this non-uniform quantizer can be found in speech communication
  - Audio and voice signals have higher-densities of smaller values

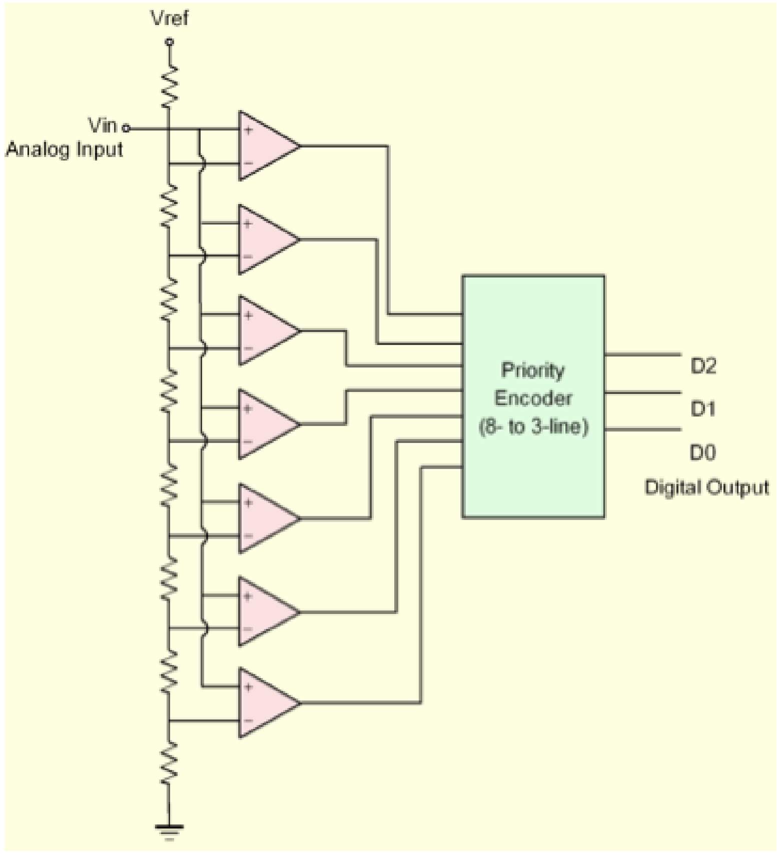
Non-uniform Quantization



if it is more probable to have values in this range, we will need to focus our sampling in this range

Flash ADC

- Parallel Design
  - A resistor divider network generates discrete voltage levels
  - Input voltage is compared against all the voltage levels at once
  - Priority Encoder considers the first "HIGH" input from the top as valid, and converts it to binary form
- Advantage: Fast
  - Conversion takes just one cycle
- Disadvantage: A lot of components needed
  - $2^n - 1$ comparators needed for n bit ADC

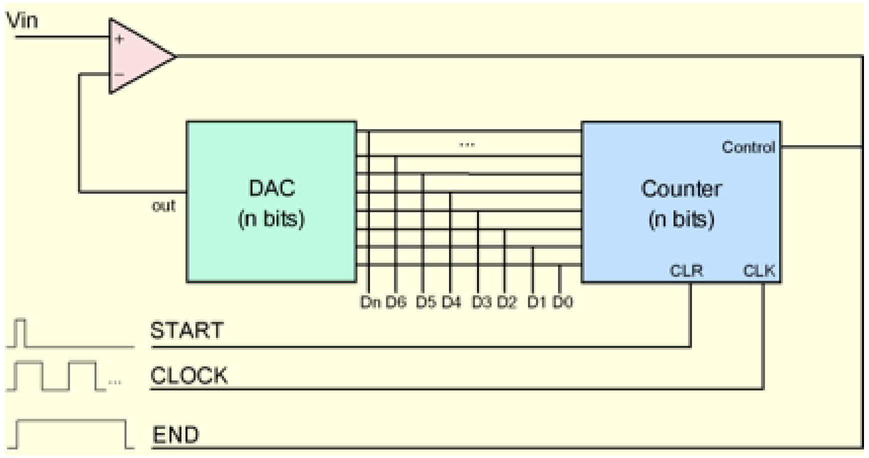


Picture Source: www.hardwaresecrets.com

Source: UConn, ECE3411

Ramp ADC

- Sequential design
- Advantage: Only a few components needed
- Disadvantage: Very slow
  - $2^n - 1$ cycles (in worst case) for n bit ADC conversion



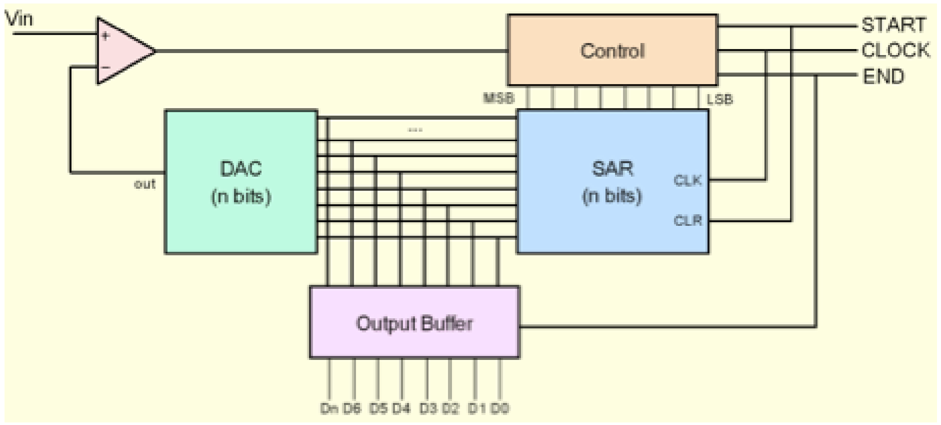
Picture Source: www.hardwaresecrets.com

- A Counter counts from 0 ⋯ 2^n
- A DAC generates discrete voltage levels corresponding to the digital values 0 ⋯ 2^n (i.e. a voltage Ramp)
- In each cycle, input voltage is compared against the current voltage level generated by DAC
- The comparator generates a "HIGH" value as soon as the ramp crosses the input value
- The corresponding counter value becomes the output

Source: UConn, ECE3411

Successive Approximation ADC

- Sequential design
- Most widely used ADC type
- Only a few components needed
- Conversion takes just *n* cycles



Picture Source: www.hardwaresecrets.com

- Closest digital value is approximated by “Binary Search”
- First, the MSB of SAR is set to 1, and the comparator decides whether the input voltage is higher or lower than DAC voltage
  - The bit value is adjusted accordingly
- The process is repeated for each bit from MSB down to LSB
- The final SAR value becomes the output

Source: UConn, ECE3411

Successive Approximation ADC

- Consider an analog input value of 0.425 V and a voltage reference (max value) of 1 V, and we use an 8-bit ADC:

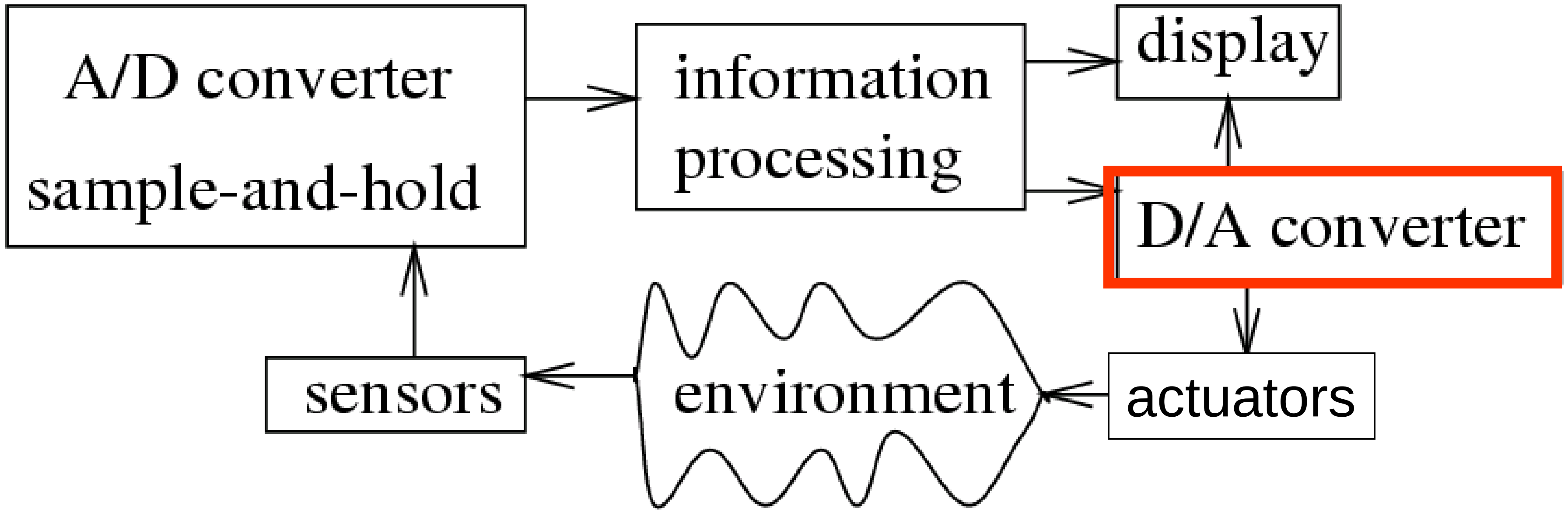
1. Set 1st bit of 8 bit output to 1 so output to DAC is 0.5
2. 0.5 is greater than 0.425 0 => the 1st bit should be 0
3. Set 2nd bit of 8 bit output to 1, so output to DAC is 0.25
4. 0.25 is less than 0.425 => the 2nd bit should be 1
5. Set 3rd bit of 8 bit output to 1, so output to DAC is 0.375
6. 0.375 is less than 0.425 => the 3rd bit should be 1
7. This process is repeated for all 8 bits until the output is determined to be: 01101100

Source: UConn, ECE3411

Outline

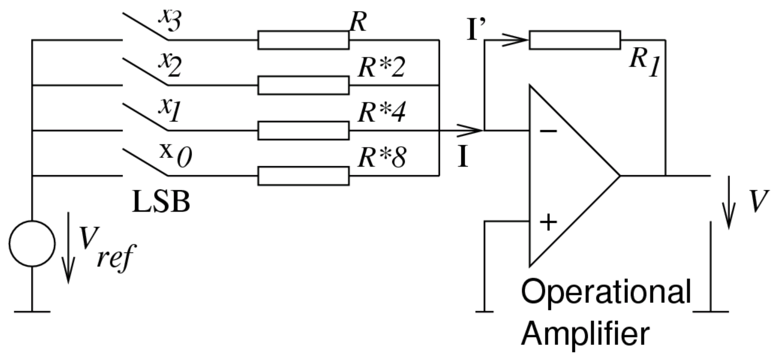
- Sampling
  - Nyquist theorem, aliasing
- Analog to Digital Conversion (ADC)
  - Quantization
  - Range, resolution, accuracy, error
  - ADC types
- **Digital to Analog Conversion (DAC)**
  - **Examples, resistance networks**
  - **DAC types**
  - **DAC performance properties**
- Noise and signal filtering
- Analog I/O on AVR

Embedded System Hardware Interfacing Sensors and Actuators

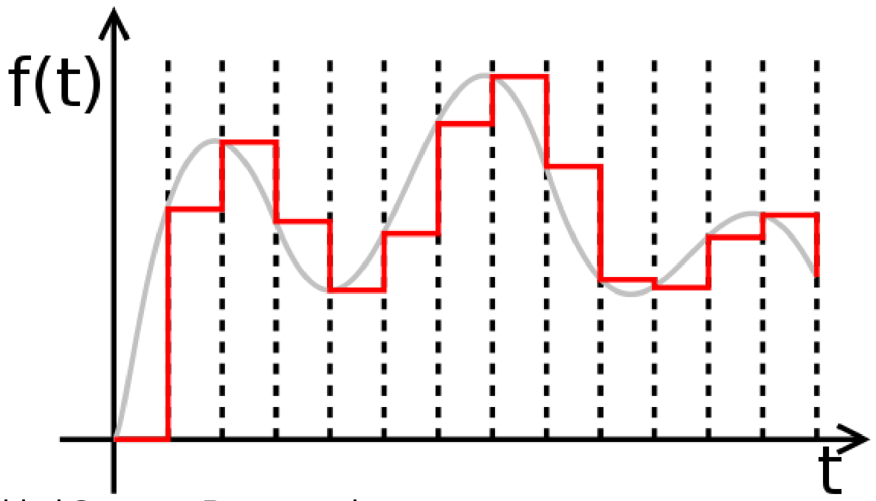


Digital to Analog Converters (DAC)

- Ideal sampling would allow us to reconstruct the signal perfectly with a sequence of impulses
  - But there is no ideal sampling, so…
- We use zero-order hold circuit to create an analog output
  - Further reconstruction: interpolate between samples with straight lines, zero-order hold circuit + a low pass reconstruction filter (to smooth out the steps)

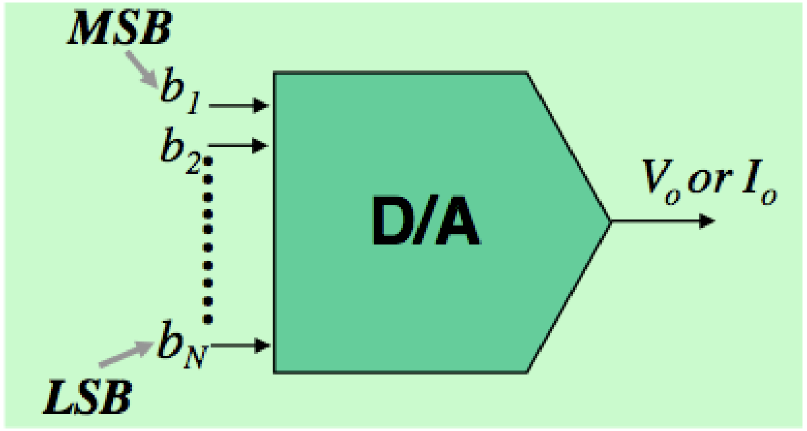


Binary-weighted DAC



Generic DAC Properties

- An ideal DAC:
  - Accepts digital inputs b_1-b_n
  - Produces either an analog output voltage or current
  - Assumption:
    - Uniform, binary digital encoding
    - Unipolar output ranging from 0 to V_{FS} (V_{max})



<u>Nomenclature:</u>

$N = \#\ of\ bits$

$V_{FS} = full\ scale\ output$

$\Delta = min.\ step\ size \rightarrow 1LSB$

$$\Delta = \frac{V_{FS}}{2^N}$$

$$or\ N = log_2 \frac{V_{FS}}{\Delta} \quad \rightarrow resolution$$

Generic DAC Properties

$$N = \#\ of\ bits$$

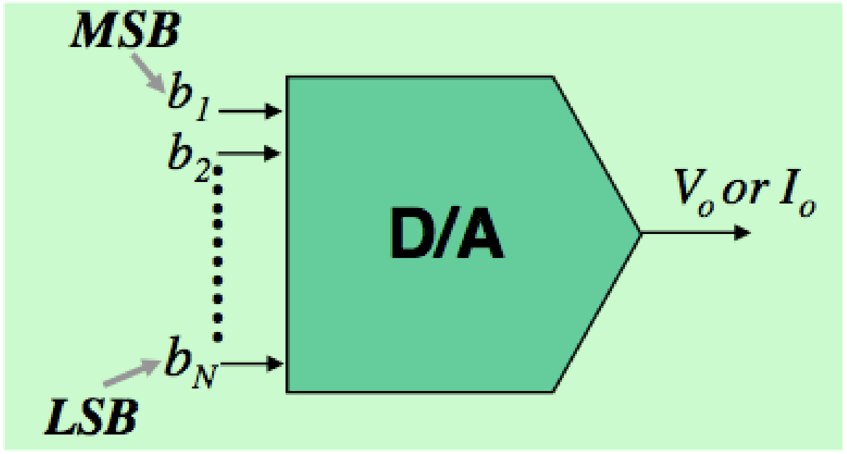
$$V_{FS} = full\ scale\ output$$

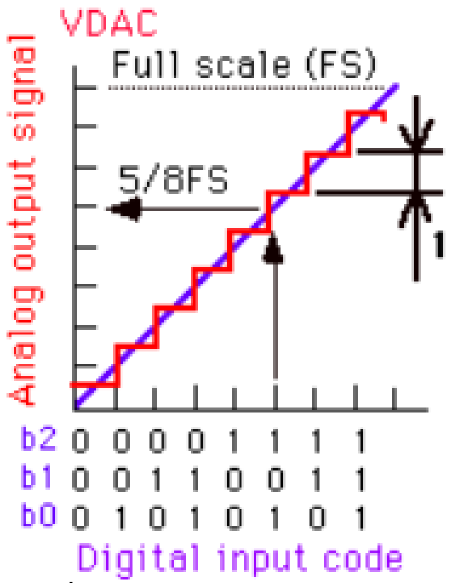
$$\Delta = min.\ step\ size \rightarrow 1LSB$$

$$\Delta = \frac{V_{FS}}{2^N}$$

$$V_0 = V_{FS} \sum_{i=1}^{N} \frac{bi}{2^i}$$

$$= \underbrace{\Delta \times \sum_{i=1}^{N} bi \times 2^{N-i}}_{\text{binary-weighted}} \quad , \quad bi = 0\ or\ 1$$





Source: EECS 247, UC Berkeley

Example: 3-bit DAC

Example: for $N=3$ and $V_{FS}=0.8V$

input code→ *101*

Find the output value V_0

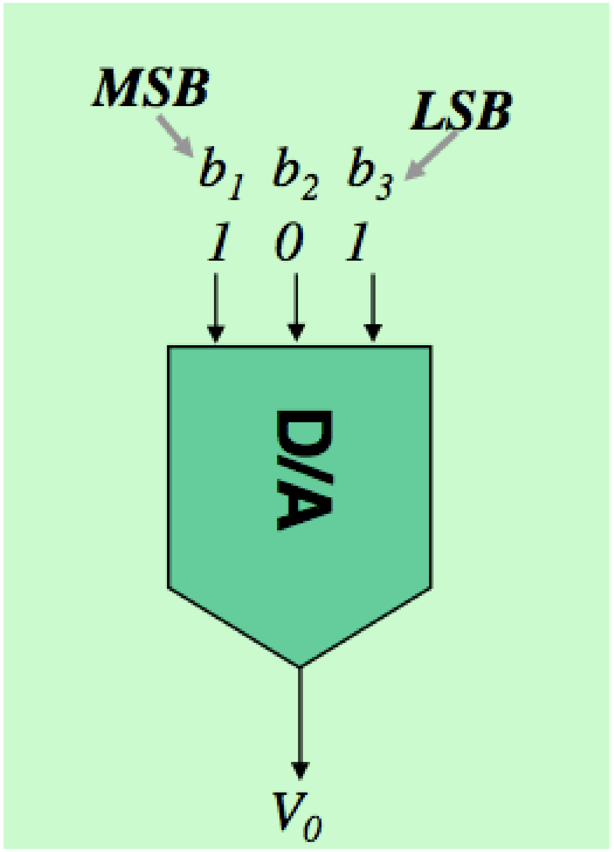
$$V_0=\Delta\left(b_1\times 2^2+b_2\times 2^1+b_3\times 2^0\right)$$

$$Then: \Delta=V_{FS}/2^3=0.1V$$

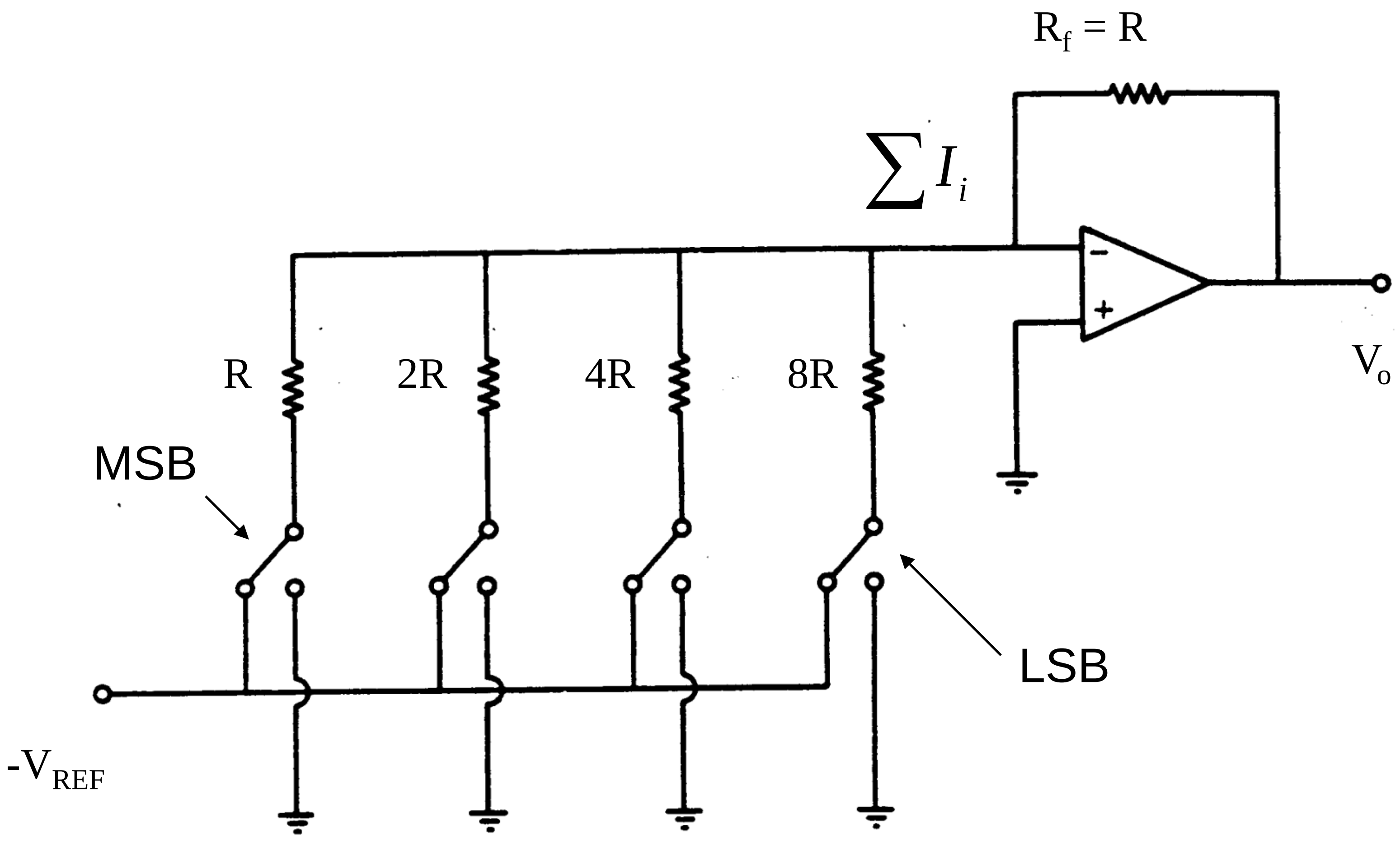
$$\rightarrow V_0=0.1V\left(1\times 2^2+0\times 2^1+1\times 2^0\right)=$$

$$\rightarrow V_0=0.5V$$

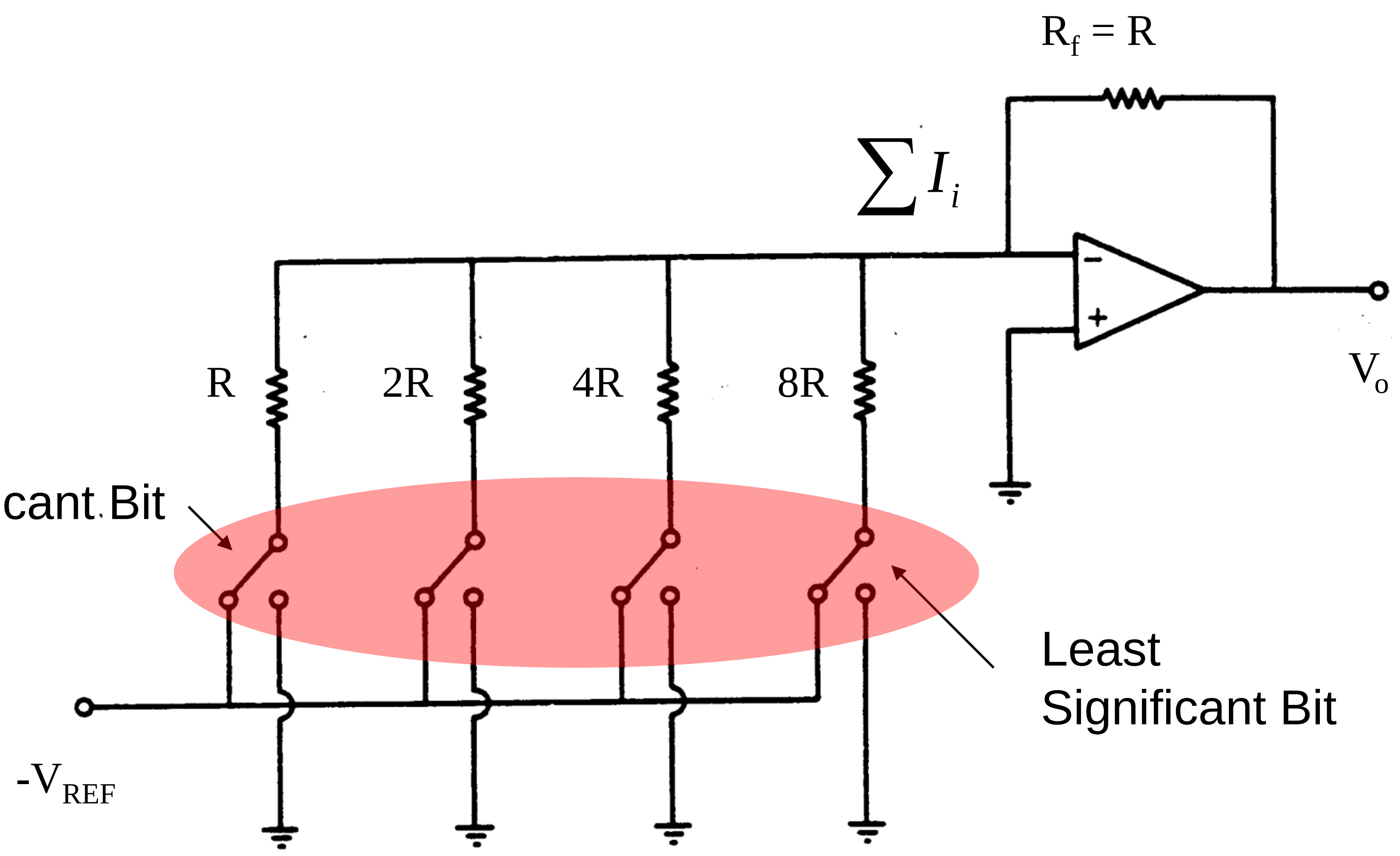
$$Note: MSB\rightarrow V_{FS}/2 \quad \& \quad LSB\rightarrow V_{FS}/2^N$$



Example: Binary Weighted Resistor DAC



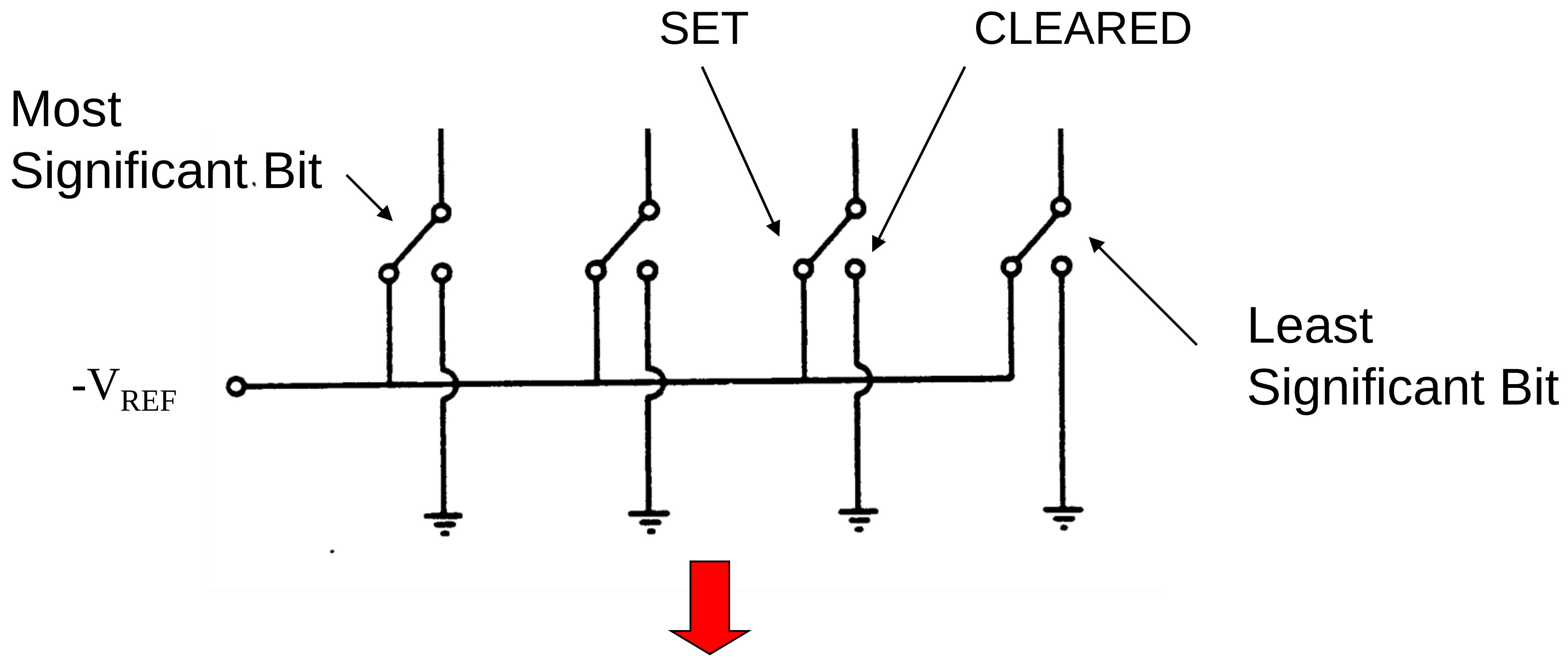
Binary Representation



Most Significant Bit

Least Significant Bit

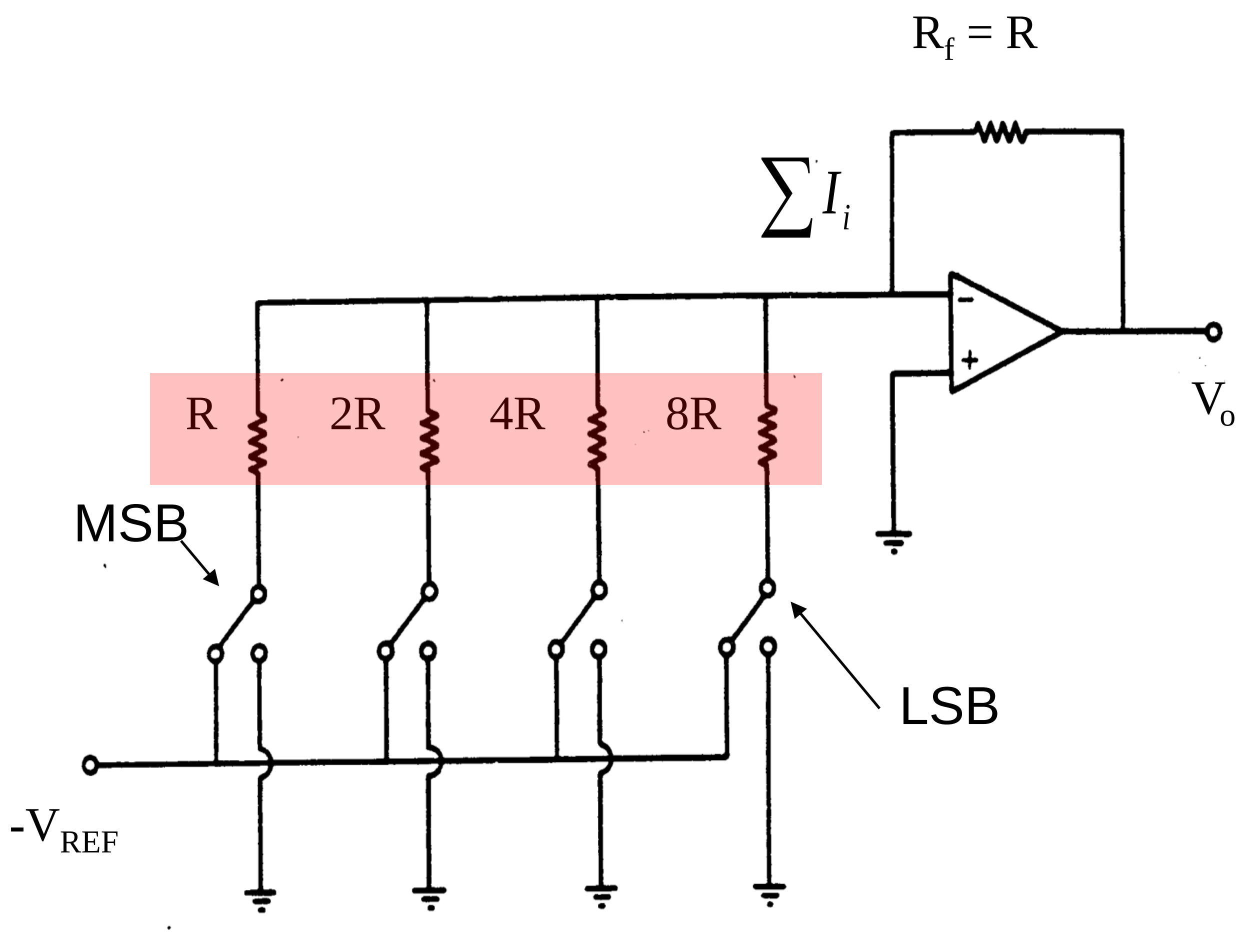
Binary Representation



$(\ 1 \quad 1 \quad 1 \qquad 1\ )_2 = (\ 15\ )_{10}$

Binary Weighted Resistor

- "Weighted Resistors" based on bit
- Reduces current by a factor of 2 for each bit



Binary Weighted Resistor - Analysis

- Result:

$$\sum I = V_{REF}\left(\frac{B_3}{R}+\frac{B_2}{2R}+\frac{B_1}{4R}+\frac{B_0}{8R}\right)$$

$$V_{OUT} = I \cdot R_f (= R) = V_{REF}\left(B_3+\frac{B_2}{2}+\frac{B_1}{4}+\frac{B_0}{8}\right)$$

  - B_i = Value of Bit i

Binary Weighted Resistor - Analysis

- More generally:

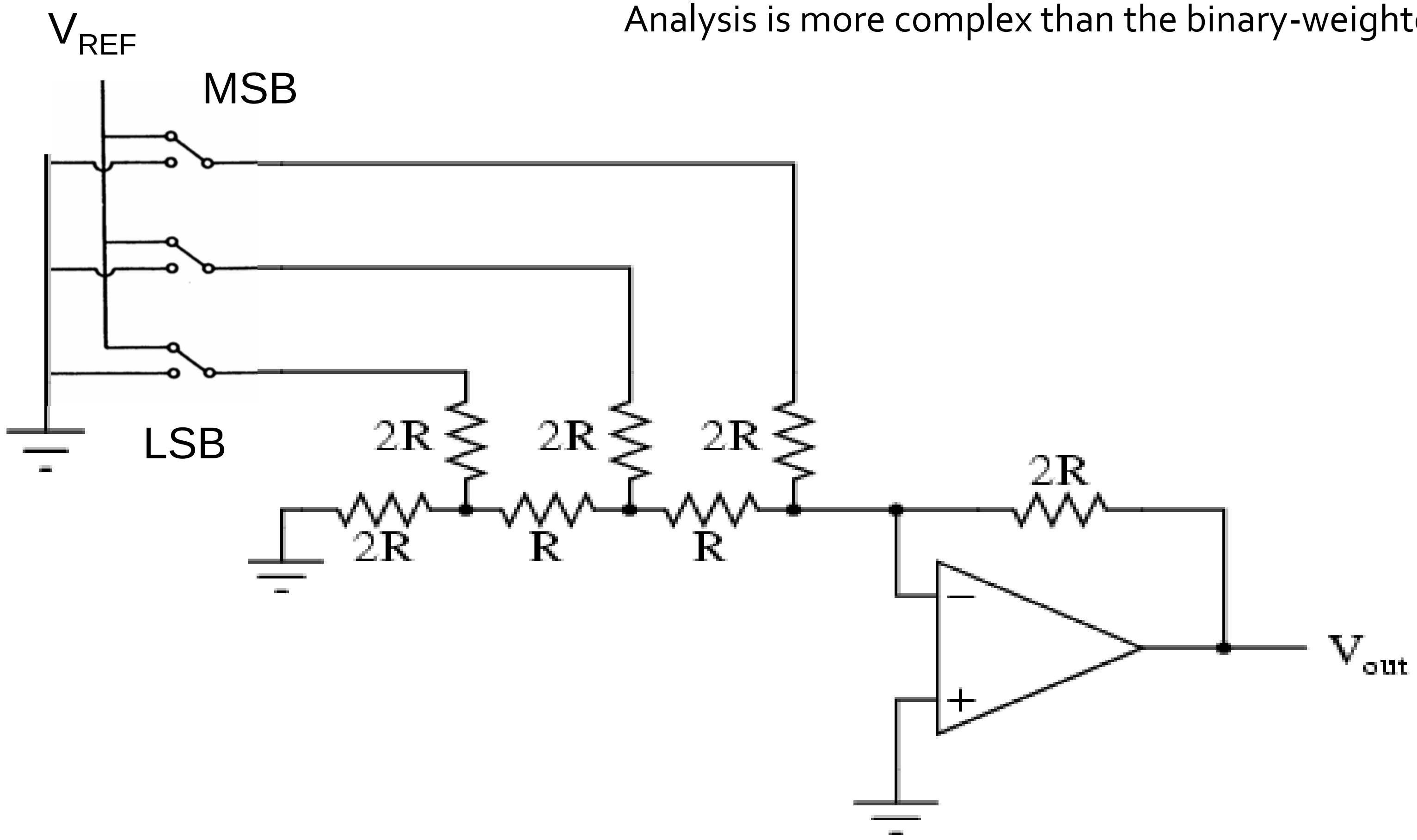
$$V_{OUT} = V_{REF} \sum \frac{B_i}{2^{n-i-1}}$$

$$= V_{REF} \cdot \text{Digital Value} \cdot \text{Resolution}$$

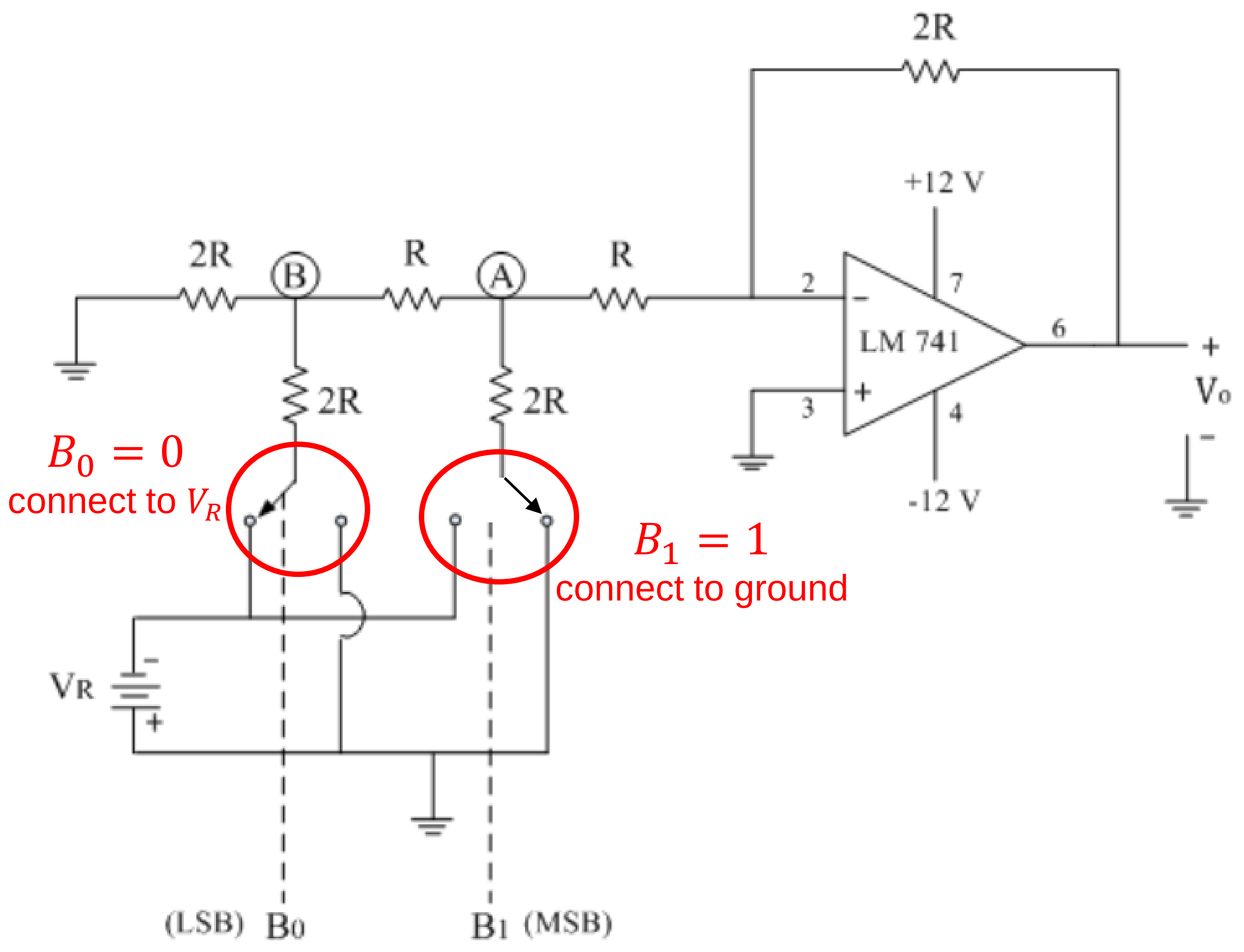
  - B_i = Value of Bit i
  - n = Number of Bits

Another Example: R-2R Ladder

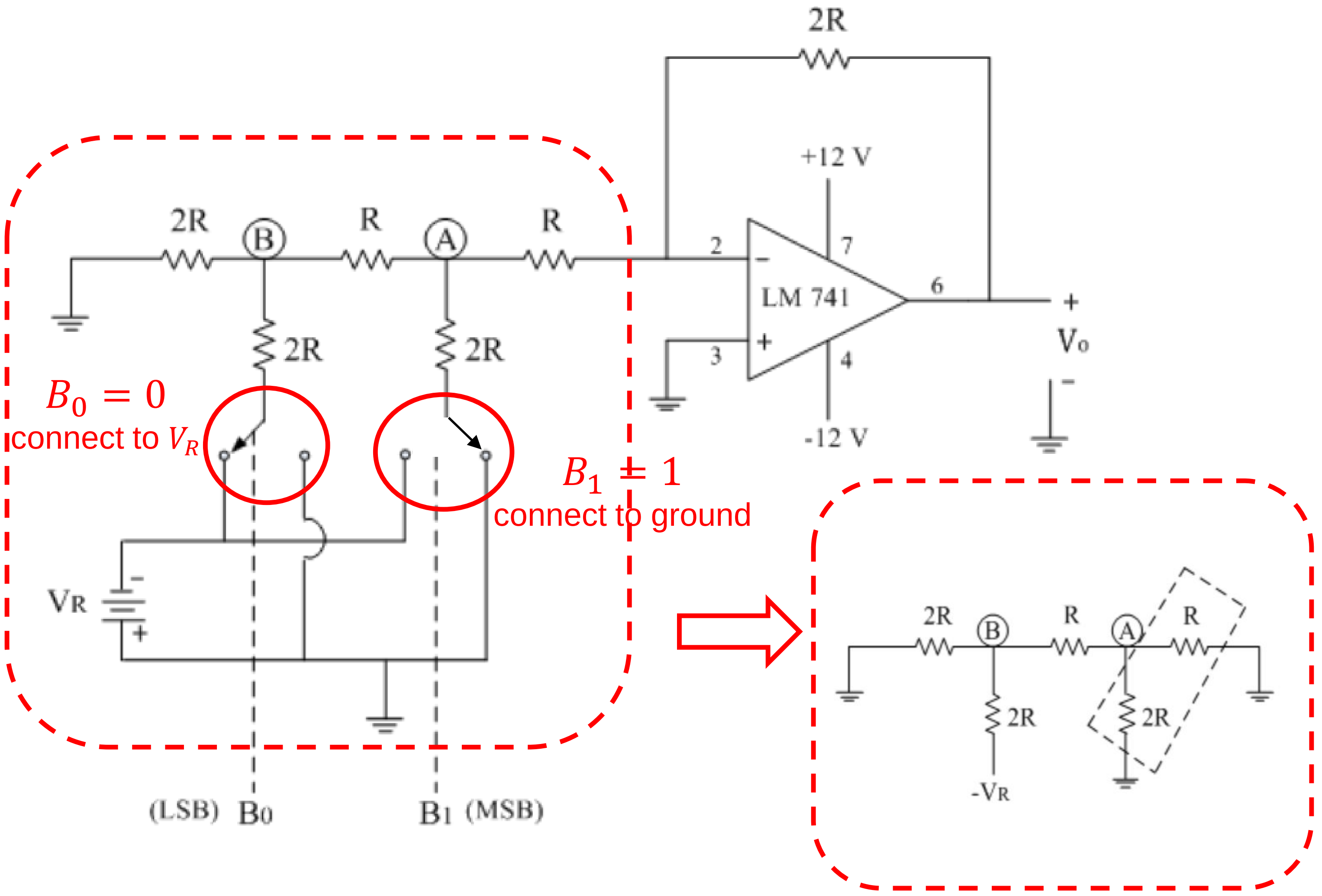
Analysis is more complex than the binary-weighted version



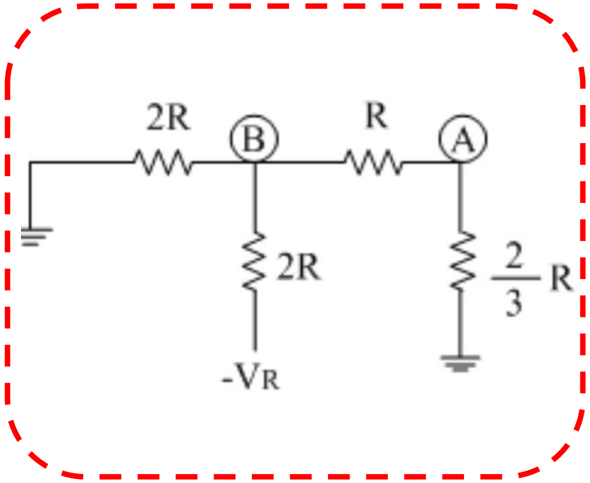
Numeric Example of R-2R Ladder



Numeric Example of R-2R Ladder



Numeric Example of R-2R Ladder



- Kirchhoff's Current Law applied at node A

$$\frac{V_A}{\frac{2}{3}R} + \frac{V_A - V_B}{R} = 0$$

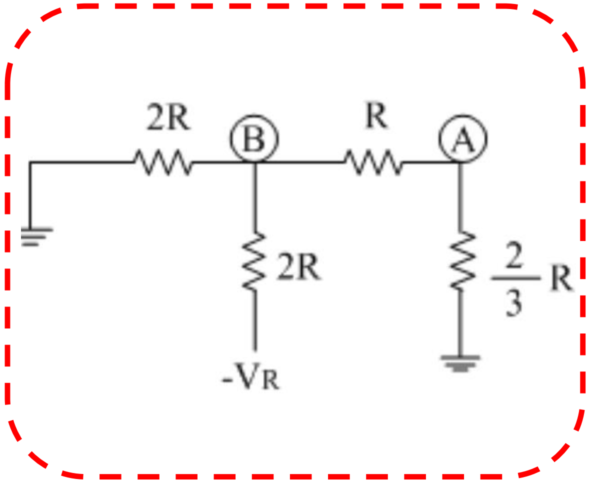
$$\therefore \frac{3V_A}{2R} + \frac{V_A - V_B}{R} = 0$$

$$\therefore \frac{3V_A + 2V_A - 2V_B}{2R} = 0$$

$$\therefore 5V_A = 2V_B$$

$$\therefore V_B = \frac{5V_A}{2}$$

Numeric Example of R-2R Ladder



- Kirchhoff's Current Law applied at node B

$$\frac{V_B}{2R} + \frac{V_B-(-V_R)}{2R} + \frac{V_B-V_A}{R} = 0$$

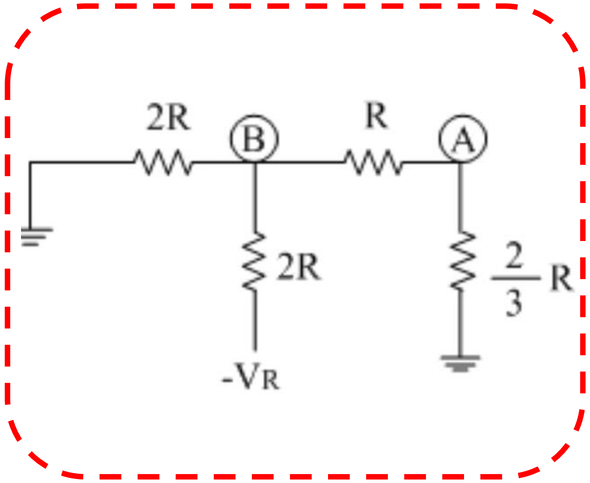
$$\therefore \frac{V_B+V_B+V_R+2V_B-2V_A}{2R} = 0$$

$$\therefore \frac{4V_B+V_R-2V_A}{2R} = 0$$

$$\therefore 4V_B+V_R-2V_A=0$$

$$\therefore V_A=2V_B+V_R/2$$

Numeric Example of R-2R Ladder



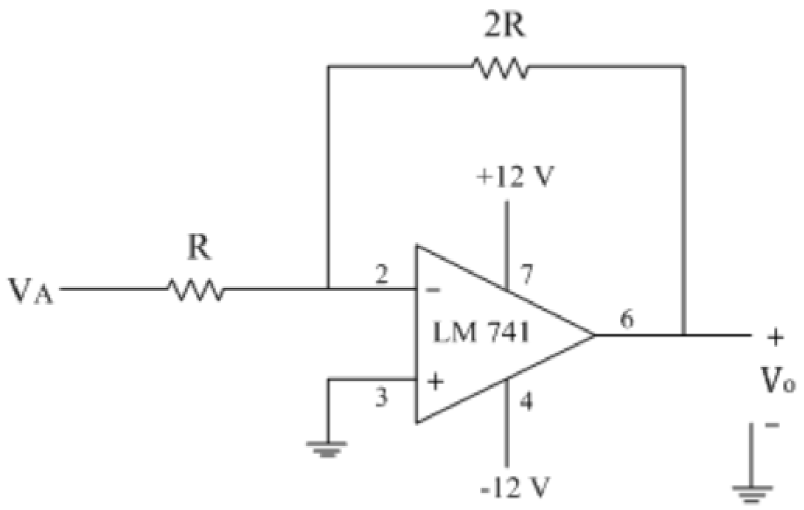
- Combining the previous 2 equations yields

$$\therefore V_A = 2\frac{5}{2}V_A + \frac{V_R}{2}$$

$$\therefore V_A = 5V_A + \frac{V_R}{4}$$

$$\therefore V_A = -\frac{V_R}{8}$$

Numeric Example of R-2R Ladder



- Equivalent circuit viewed from V_A

$$V_o = -2V_A = \frac{V_R}{4}$$

Pros & Cons

	Binary Weighted	R-2R
Pros	Easily understood	Only 2 resistor values Easier implementation Easier to manufacture Faster response time
Cons	Limited to ~ 8 bits Large # of resistors Susceptible to noise Expensive Greater Error	More complex analysis

Coming next week

Other DAC Types

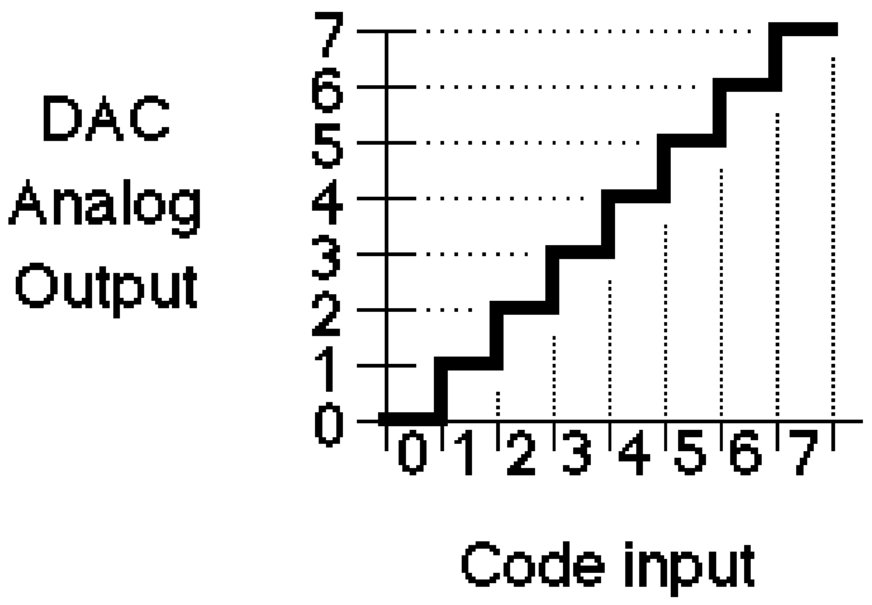
- Resistor networks
- Current sources
- Voltage and current output
- Other types
  - Capacitor networks
  - Pulse width modulation
  - Delta-sigma

D/A Converter Types

- Unipolar (Unsigned)
- Bipolar (Signed)
  - Offset binary
  - Signed magnitude
  - Two’s complement

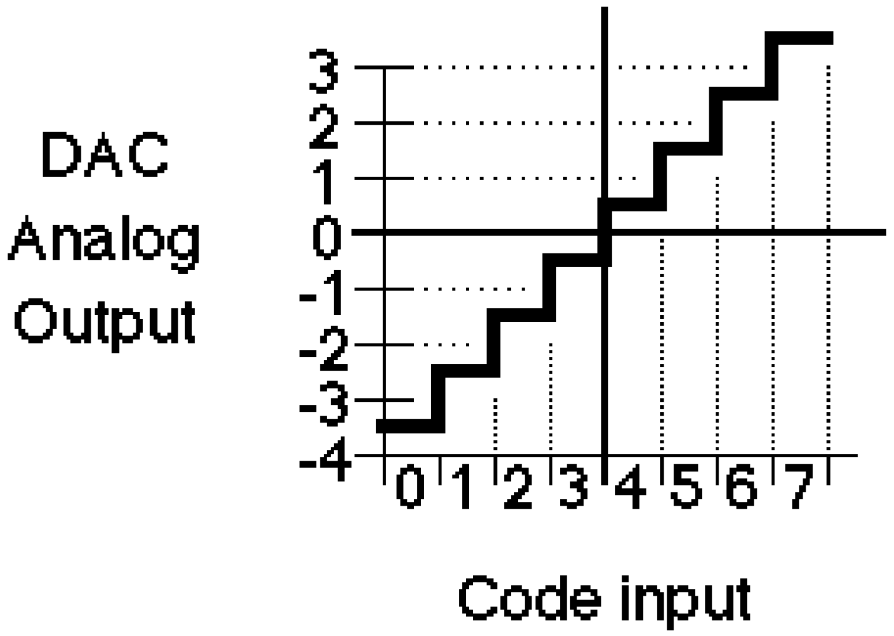
Unipolar DAC

- Unsigned Binary
  - 3-Bit DAC shown
  - Ideal (No Errors)
- DAC Out =
  - Reference * n/2^m
    - Input=n
    - m bit DAC
  - Range: 0 to 2^m-1
  - Resolution: 1/ 2^m



Offset Binary DAC

- Unipolar DAC Shifted Negative ~1/2 Range
- Example: 3-bit shown
  - Same as unipolar, but
  - Shifted down 3.5 V
  - Range: -3.5 to +3.5
  - Resolution: 1
  - No true zero output

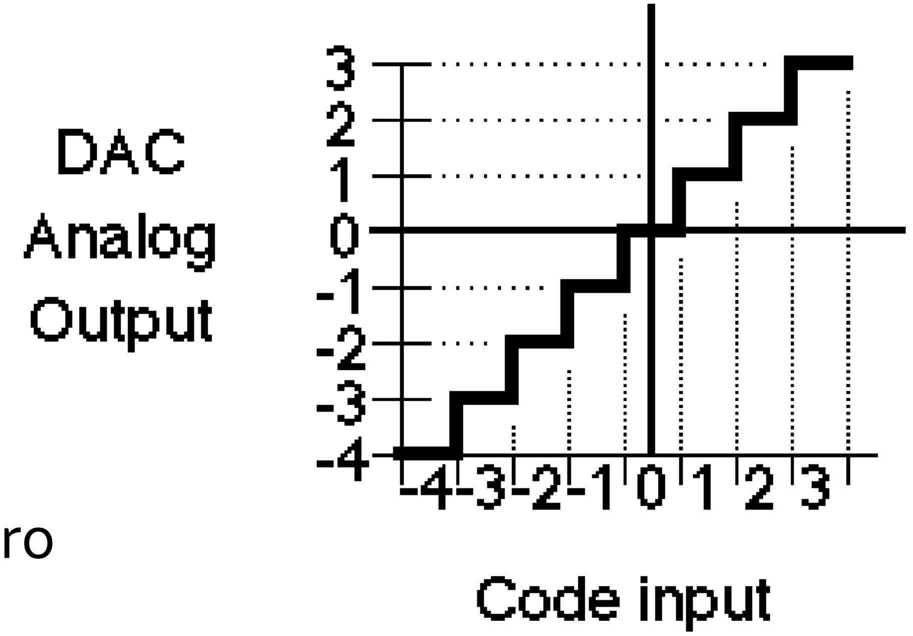


Signed Magnitude DAC

- Unipolar DAC plus analog circuitry to:
  - Multiply DAC Output by +1 or -1 depending on the sign bit input
  - More complex than offset binary
  - Equivalent to m+1 bit signed DAC
  - Symmetrical about zero: +0 and -0
- Range: $-Vref*(2^m-1)/2^m$ to $+Vref*(2^m-1)/2^m$
- Resolution: $Vref/2^m$

Two's Complement DAC

- Code MSB = -2^{m-1}
- Example: 3-bit
  - Range: -4 to +3
  - Resolution: 1
  - Reference = 4
- True zero output
- Asymmetrical output about zero
- Simplest software



DAC Performance Specifications

- Resolution
- Reference Voltages
- Settling Time
- Linearity
- Speed
- Errors

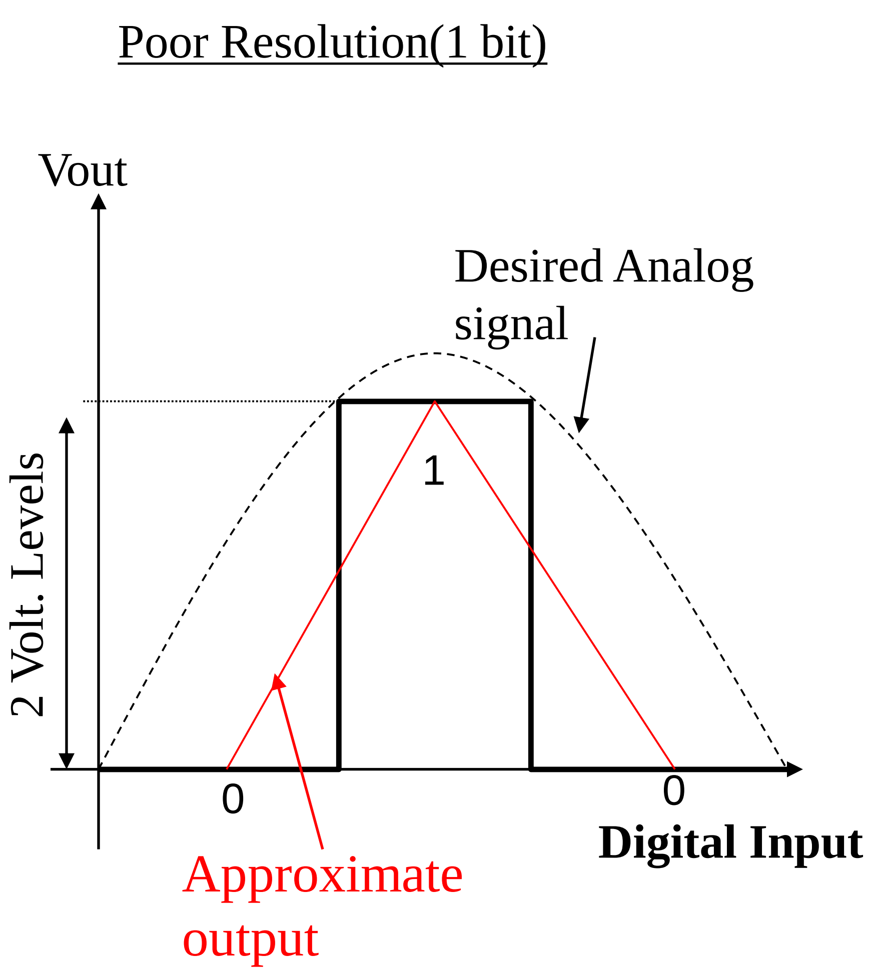
Resolution

- Resolution: is the amount of variance in output voltage for every change of the LSB in the digital input
- How closely can we approximate the desired output signal (Higher Res. = finer detail = smaller voltage divisions)
- A common DAC has a 8 - 12 bit Resolution

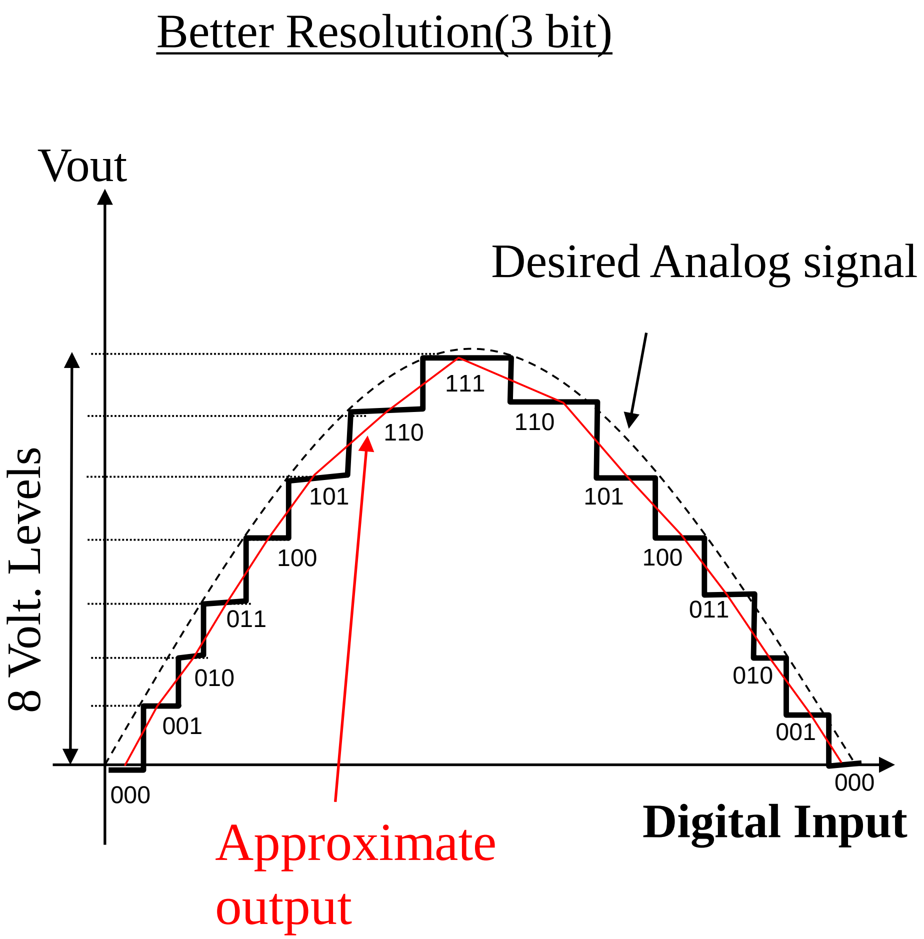
$$\text{Resolution} = V_{LSB} = \frac{V_{\text{Ref}}}{2^N}$$

N = Number of bits

Resolution



Poor Resolution(1 bit)
Vout
Desired Analog signal
1
0
0
2 Volt. Levels
Digital Input
Approximate output

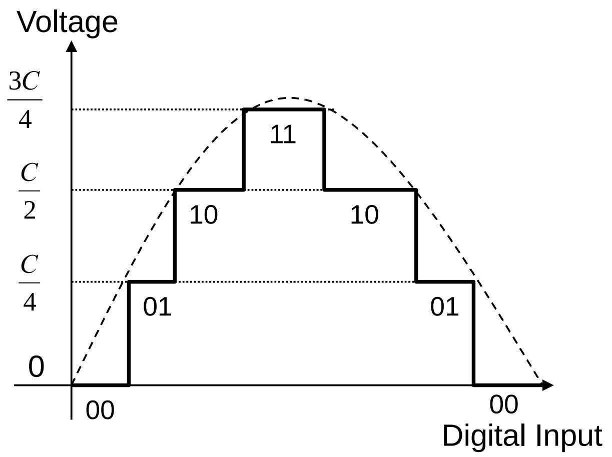


Better Resolution(3 bit)
Vout
Desired Analog signal
111
110
110
101
101
100
100
011
011
010
010
001
001
000
000
8 Volt. Levels
Digital Input
Approximate output

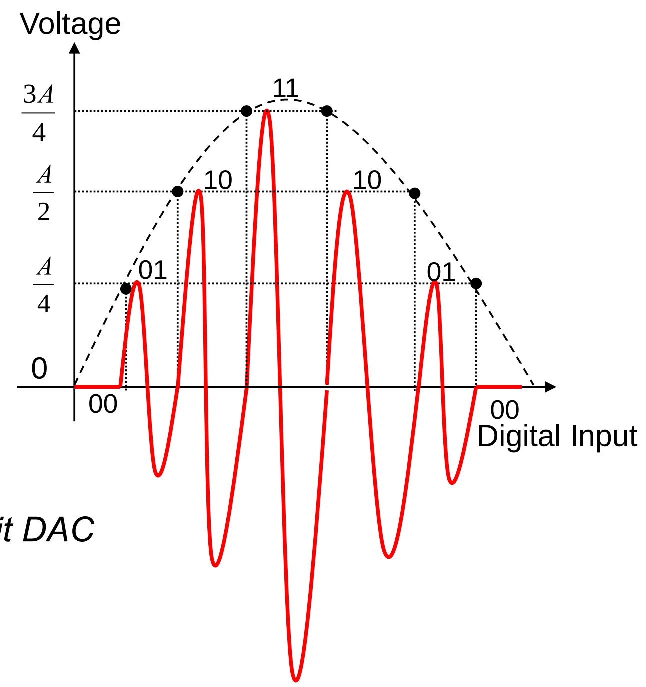
Reference Voltage

- Reference Voltage: A specified voltage used to determine how each digital input will be assigned to each voltage division
- Types:
  - Non-multiplier: internal, fixed, and defined by manufacturer
  - Multiplier: external, variable, user specified

Non-Multiplier: (V_{ref} = C)



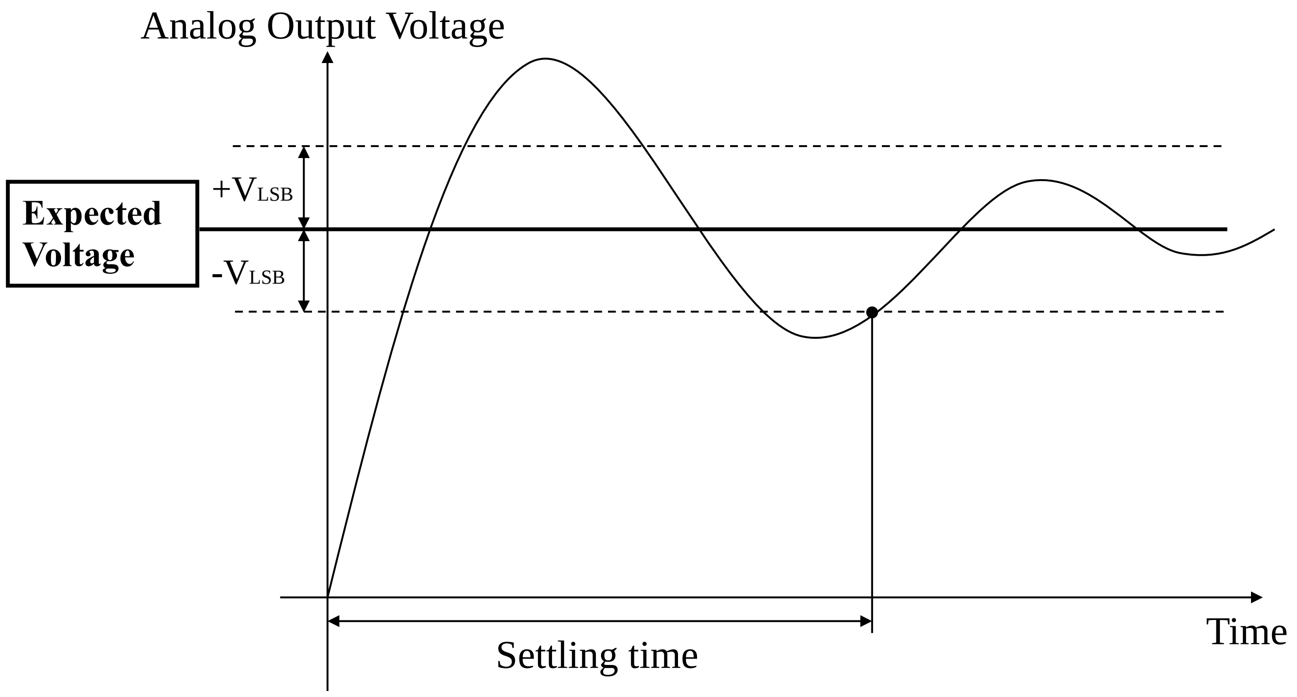
Multiplier: (V_{ref} = Asin(wt))



Assume 2 bit DAC

Settling Time

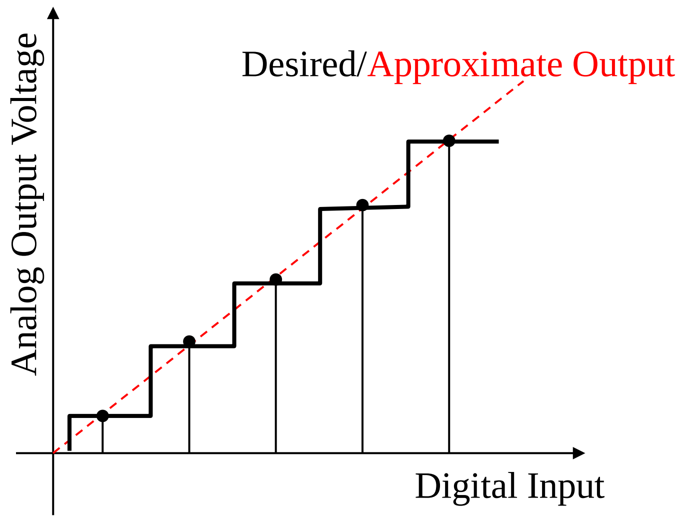
- Settling Time: The time required for the input signal voltage to settle to the expected output voltage(within +/- V_{LSB}).
- Any change in the input state will not be reflected in the output state immediately. There is a time lag, between the two events



Linearity

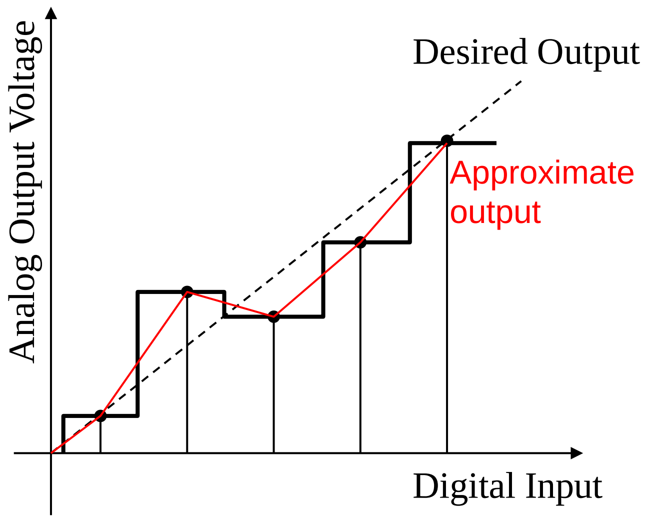
- Linearity: is the difference between the desired analog output and the actual output over the full range of expected values
- Ideally, a DAC should produce a linear relationship between a digital input and the analog output, this is not always the case

Linearity(Ideal Case)



Perfect Agreement

NON-Linearity(Real World)



Miss-alignment

Speed

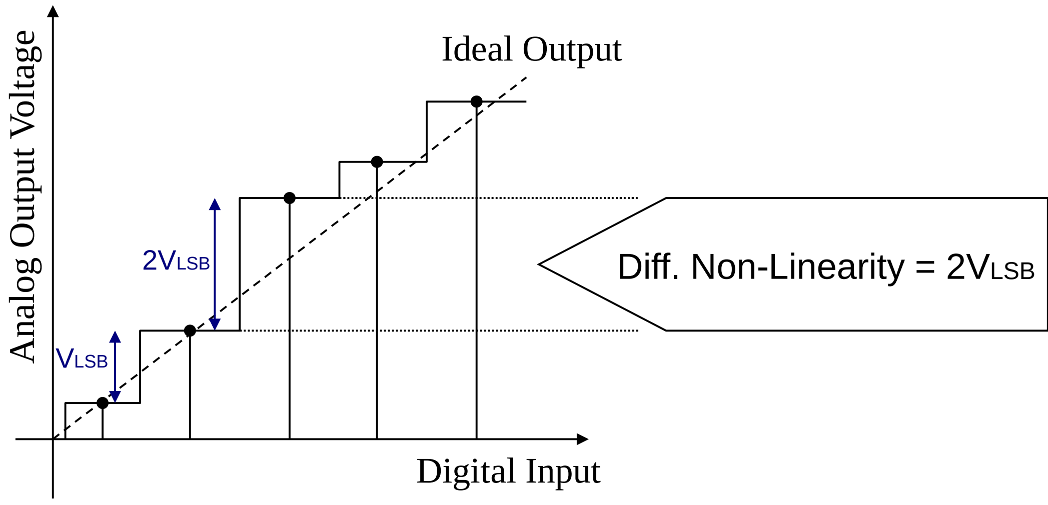
- Speed: Rate of conversion of a single digital input to its analog equivalent
- Conversion rate
  - Depends on clock speed of input signal
  - Depends on settling time of converter

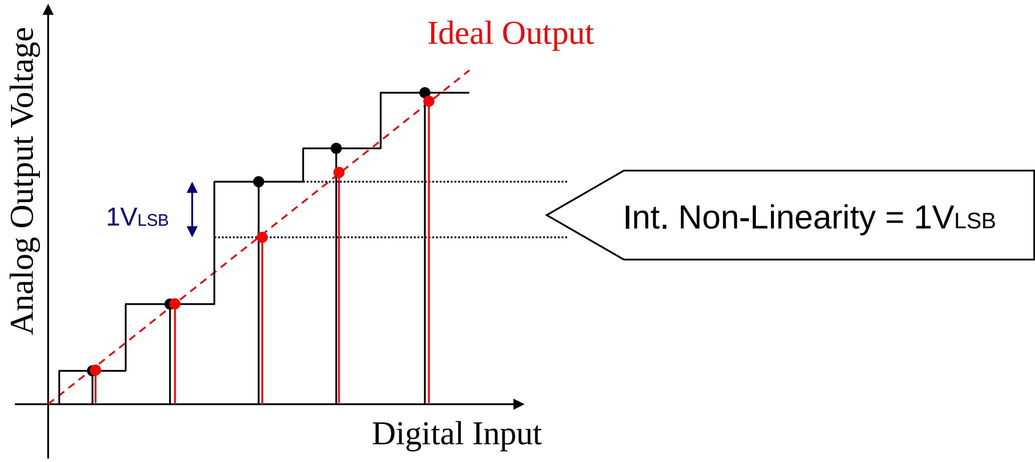
Errors

- Non-linearity
  - Differential
  - Integral
- Gain
- Offset
- Non-monotonicity

Non-linearity

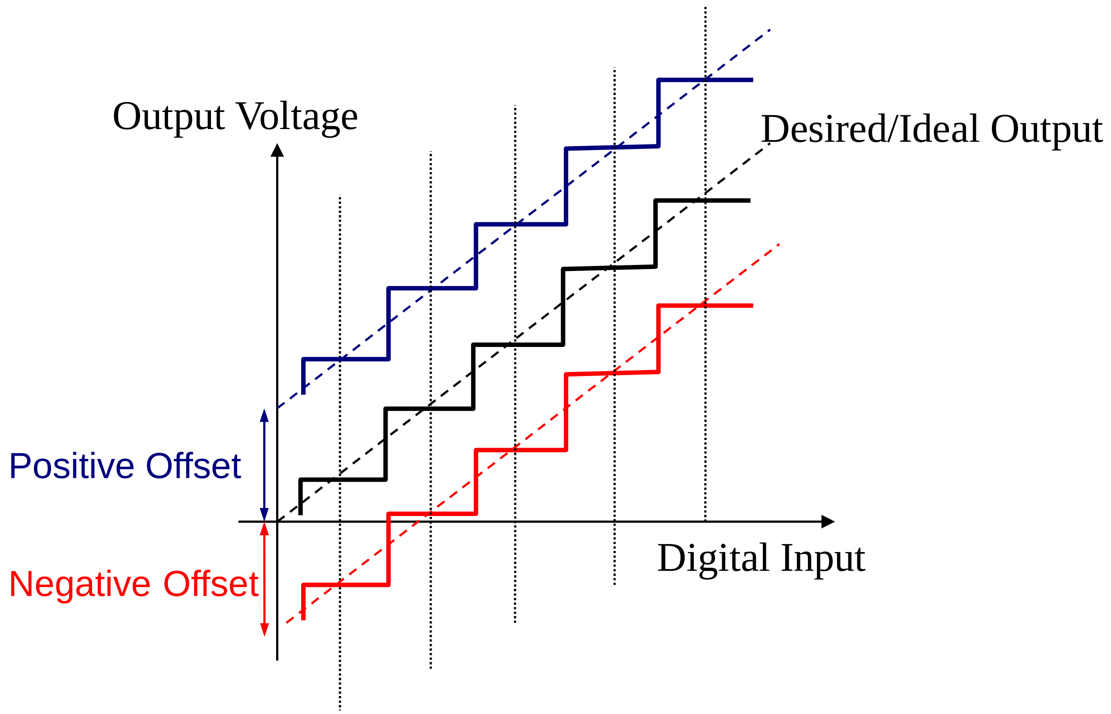
- <u>Differential Non-Linearity:</u> Difference in voltage step size from the previous DAC output (Ideally All DLNs = 1 V_{LSB})
- <u>Integral Non-Linearity:</u> Deviation of the actual DAC output from the ideal (Ideally all INLs = 0)





Offset

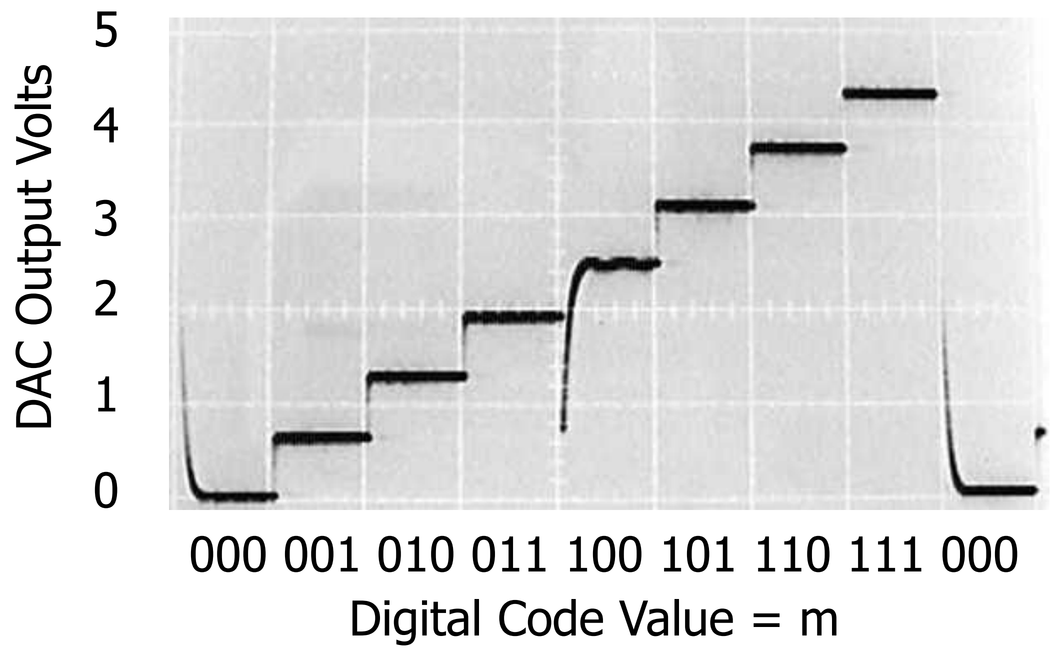
- <u>Offset Error:</u> A constant voltage difference between the ideal DAC output and the actual
  - The voltage axis intercept of the DAC output curve is different than the ideal



DAC Output Glitch

- Bits don’t change at exactly the same time
- Ex: transition from m=3 to m=4
  - Results in

    011-000-100



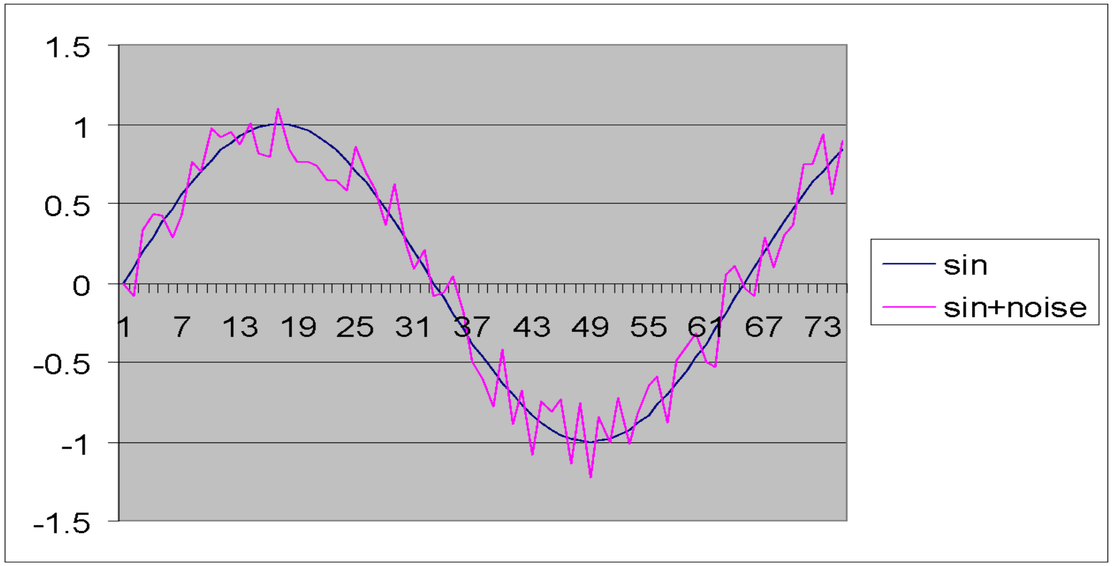
DAC output exhibits glitch on MSB 0->1 transition between codes 011 and 100

Outline

- Sampling
  - Nyquist theorem, aliasing
- Analog to Digital Conversion (ADC)
  - Quantization
  - Range, resolution, accuracy, error
  - ADC types
- Digital to Analog Conversion (DAC)
  - Examples, resistance networks
  - DAC types
  - DAC performance properties
- **Noise and signal filtering**
- Analog I/O on AVR

Signal vs. Noise?

- Signal = What we want
- Noise = Everything else!
- Filtering:
  - Estimating the noise free signal
  - Estimate = (Noisy signal) – (Estimate of noise)



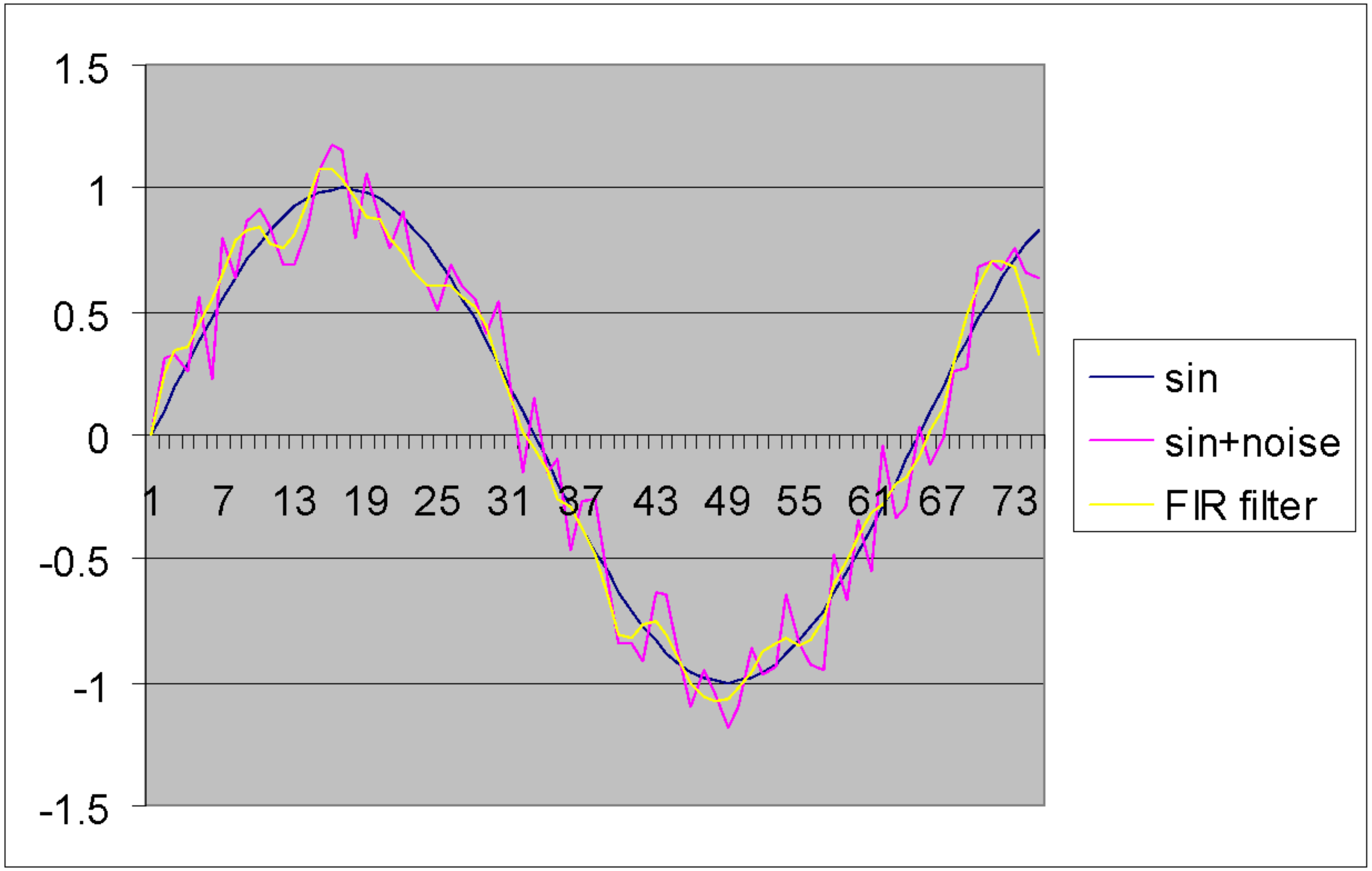
Filtering

- Depends on assumptions about:
  - Signal characteristics
  - Noise characteristics
- Common assumptions:
  - Noise has an average value of 0
  - Signal has smaller bandwidth than noise
  - Noise is additive
- Common solution:
  - Average multiple measurements

Filter – Weighted Moving Average

- Y_i = (Output) is the average of
  - Current (X_i) and past **inputs** (X_{i-1}, X_{i-2}, X_{i-3}, …)
- Example, weighted average of inputs:
  - $Y_i = C_0 * X_i + C_1 * X_{i-1}$
- Equally weighted:
  - $Y_i = (X_i + X_{i-1}) /2$
  - $= 0.5 * X_i + 0.5 * X_{i-1}$
- So in this case, coefficient $C_0 = C_1 = 0.5$

Example: Moving Average Filter



Filter Terminology

- FIR – Finite Impulse Response
  - $Y_i = C_0X_i + C_1X_{i-1} + \dots$
  - Non-recursive
- IIR – Infinite Impulse Response
  - $Y_i = C_0X_i + \dots + C_1Y_{i-1} + \dots$
  - Recursive

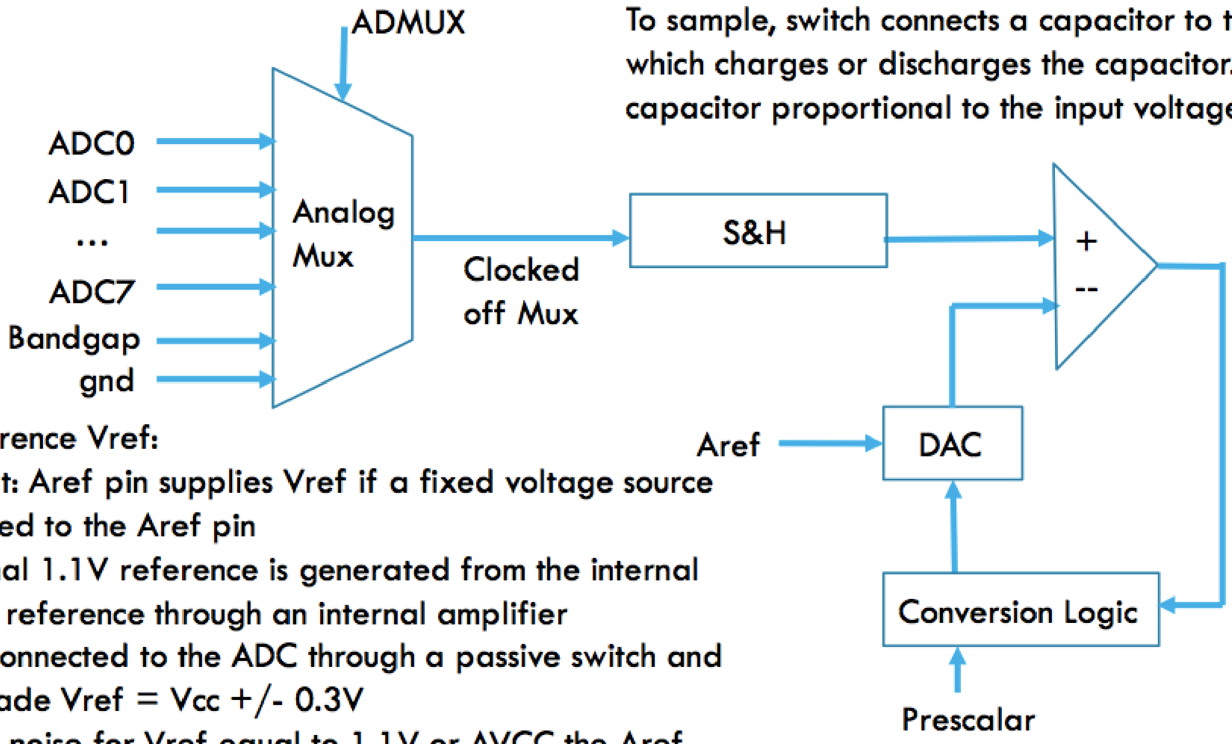
Outline

- Sampling
  - Nyquist theorem, aliasing
- Analog to Digital Conversion (ADC)
  - Quantization
  - Range, resolution, accuracy, error
  - ADC types
- Digital to Analog Conversion (DAC)
  - Examples, resistance networks
  - DAC types
  - DAC performance properties
- Noise and signal filtering
- **Analog I/O on AVR**

ADC on AVR (starting at page 237 of your manual)

- 10-bit Resolution
- 0.5LSB Integral Non-linearity
- ±2LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 6 Multiplexed Single Ended Input Channels
- 2 Additional Multiplexed Single Ended Input Channels(TQFP and QFN/MLF Package only)
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Readout
- 0 - VCC ADC Input Voltage Range
- Selectable 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

ADC Block Diagram



To sample, switch connects a capacitor to the output of a buffer amplifier, which charges or discharges the capacitor. This makes voltage across the capacitor proportional to the input voltage. To hold, the switch disconnects.

Conversion logic implements a successive approximation algorithm (a binary search; one bit per search):

- DAC takes as input the output of the conversion logic and converts it to an analog voltage where Aref sets the full range
- Analog comparator decides whether the DAC output or input voltage is the largest

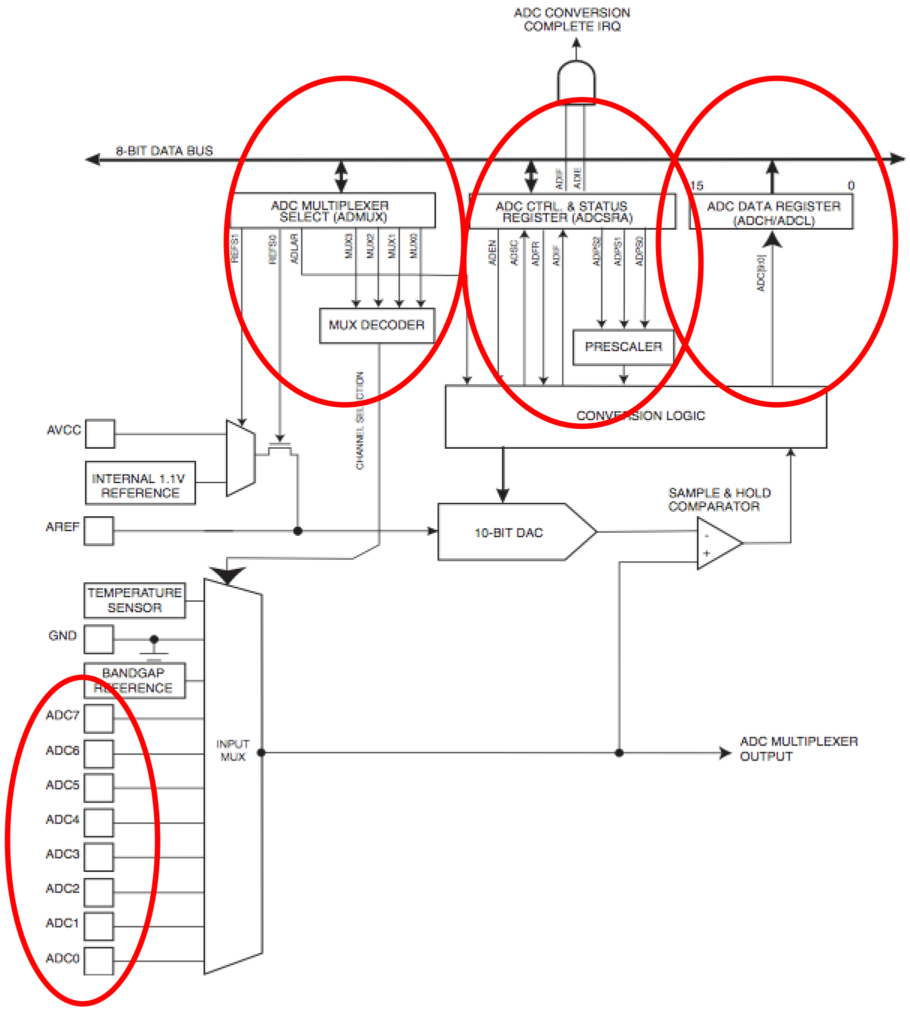
Voltage reference Vref:

- By default: Aref pin supplies Vref if a fixed voltage source is connected to the Aref pin
- The internal 1.1V reference is generated from the internal bandgap reference through an internal amplifier
- AVCC is connected to the ADC through a passive switch and can be made Vref = Vcc +/- 0.3V
- To reduce noise for Vref equal to 1.1V or AVCC the Aref pin can be externally decoupled by a capacitor to ground

Source: UConn, ECE3411

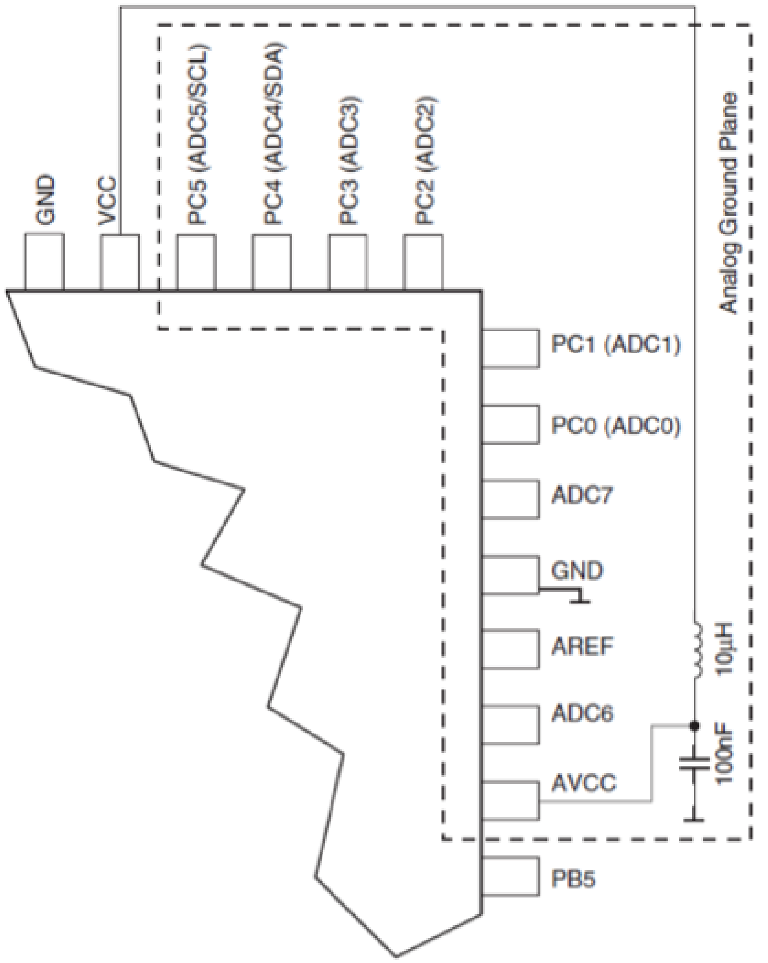
ADC Diagram

Figure 24-1. Analog to Digital Converter Block Schematic Operation,



Pin Assignments

Figure 23-9. ADC Power Connections

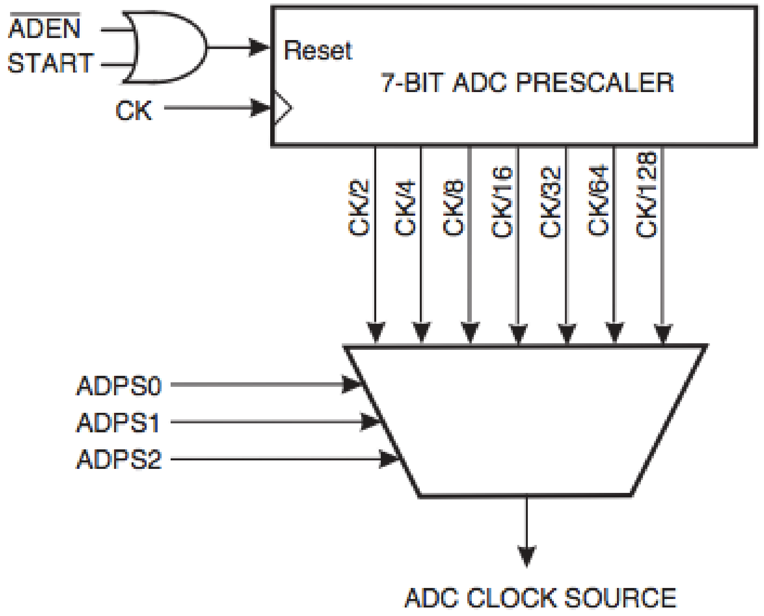


General Framework

- The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC.
- The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.
- The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.
- If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

ADC Pre-scaling

- By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution.
- If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher conversion rate.



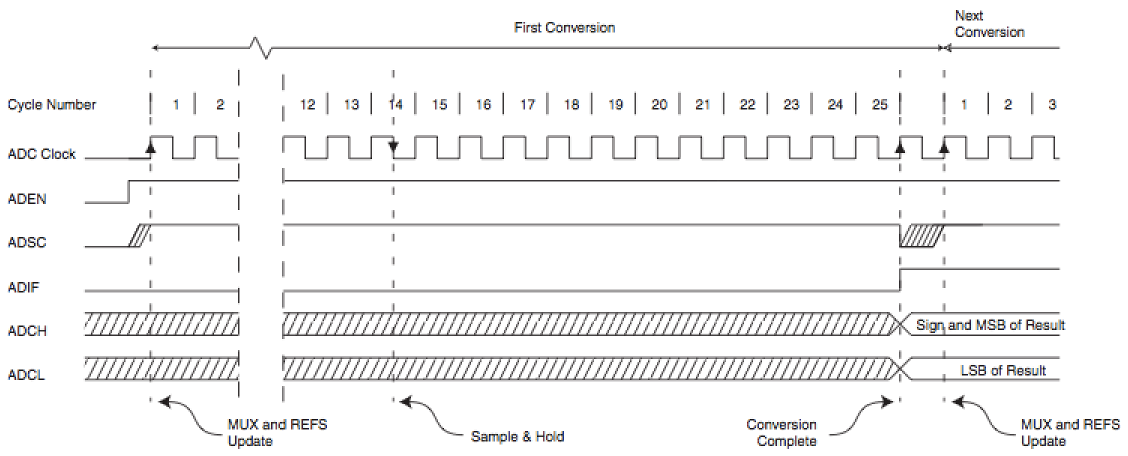
The faster you convert, you get a smaller number of accurate output bits (since the binary search cannot completely finish)

ADC Pre-scaling

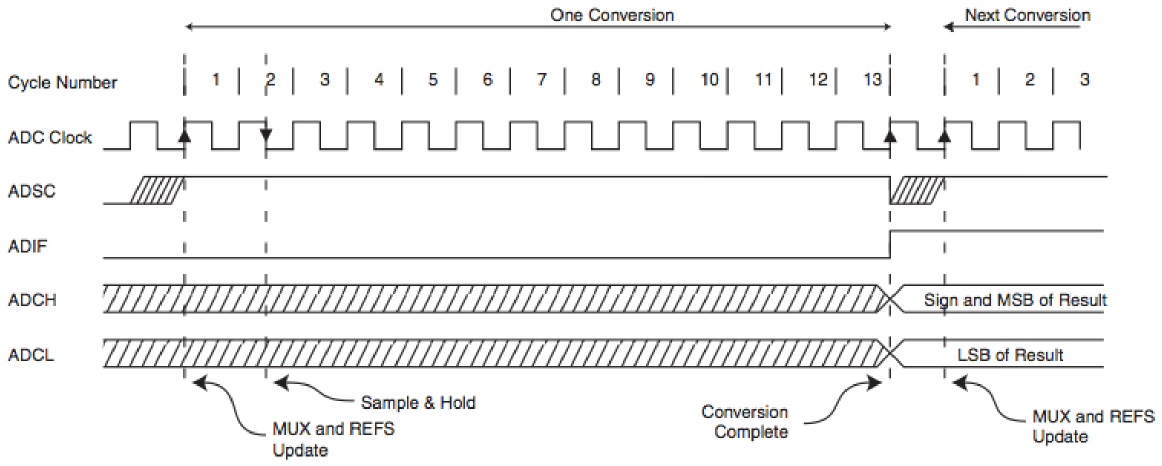
- The pre-scaling is set by the ADPS bits in ADCSRA.
- The pre-scaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The pre-scaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.
- A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.
- When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously.

ADC Conversion Timing Diagrams

ADC Timing Diagram, First Conversion (Single Conversion Mode)

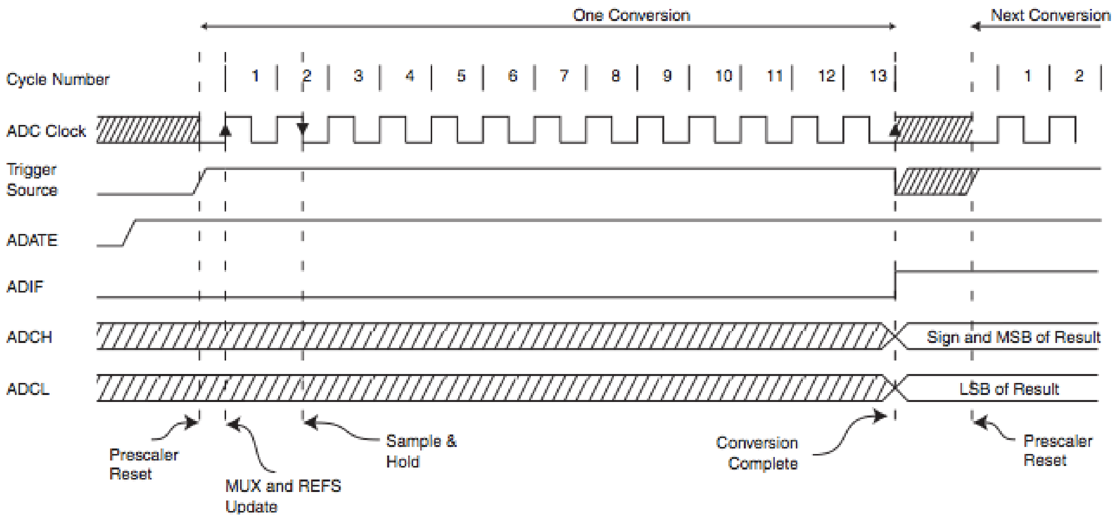


ADC Timing Diagram, Single Conversion



ADC Conversion Timing Diagrams

ADC Timing Diagram, Auto Triggered Conversion



ADC Timing Diagram, Free Running Conversion

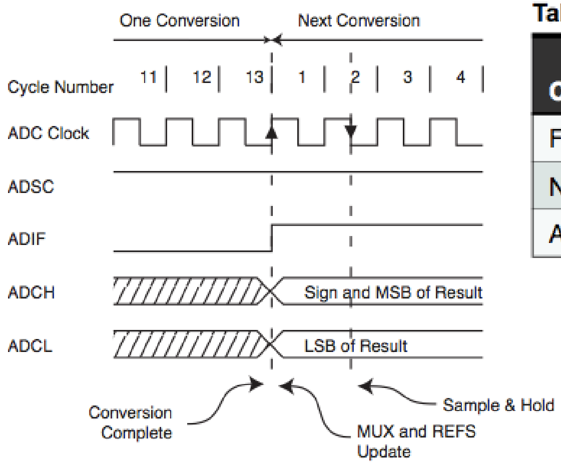


Table 24-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5

ADMUX Register

- Bit 7:6 – REFS[1:0]: Reference Selection Bits

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- Bit 5 – ADLAR: ADC Left Adjust Result
  - 1: Left-adjust, 0: Right-adjust
- Bits 3:0 – MUX[3:0]: Analog Channel Selection Bits

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Input Channel Selection

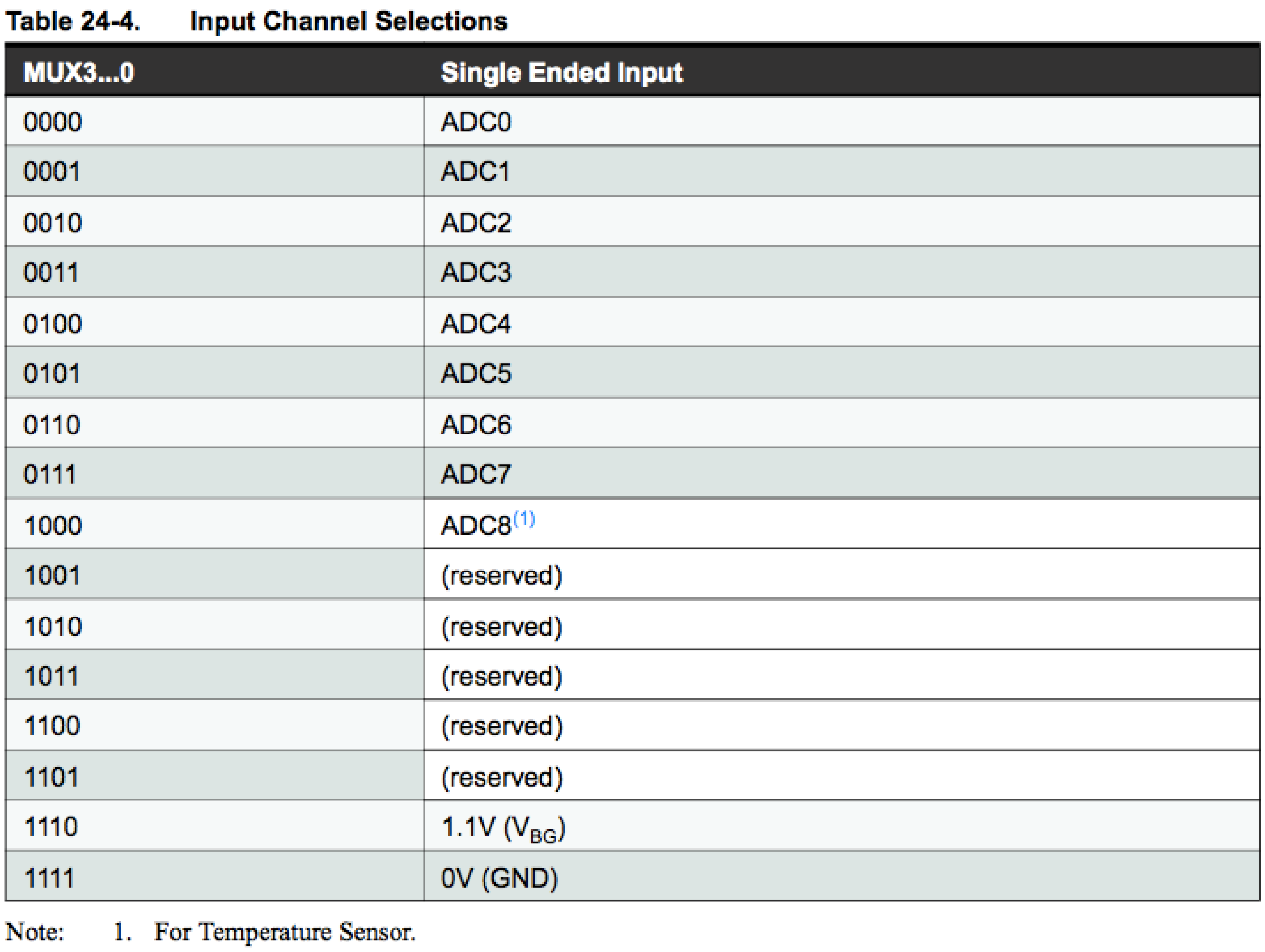


Table 24-4. Input Channel Selections

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Note: 1. For Temperature Sensor.

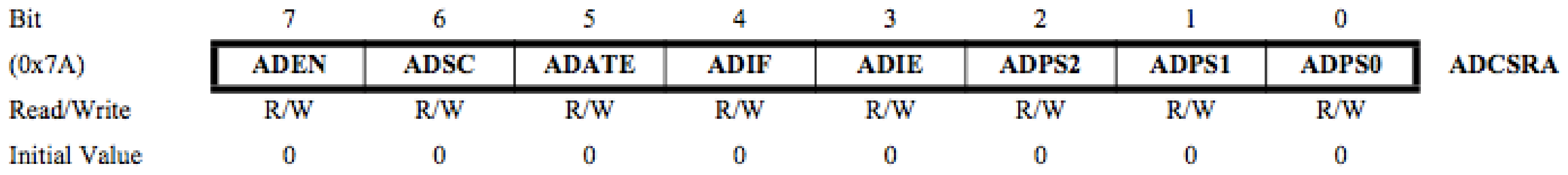
ADCSRA – ADC Control and Status Register A

- Bit 7 – ADEN: ADC Enable: Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off.
- Bit 6 – ADSC: ADC Start Conversion: In Single Conversion mode, write this bit to one to start each conversion. ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.
- Bit 5 – ADATE: ADC Auto Trigger Enable: When this bit is written to one, Auto Triggering of the ADC is enabled.

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADCSRA – ADC Control and Status Register A

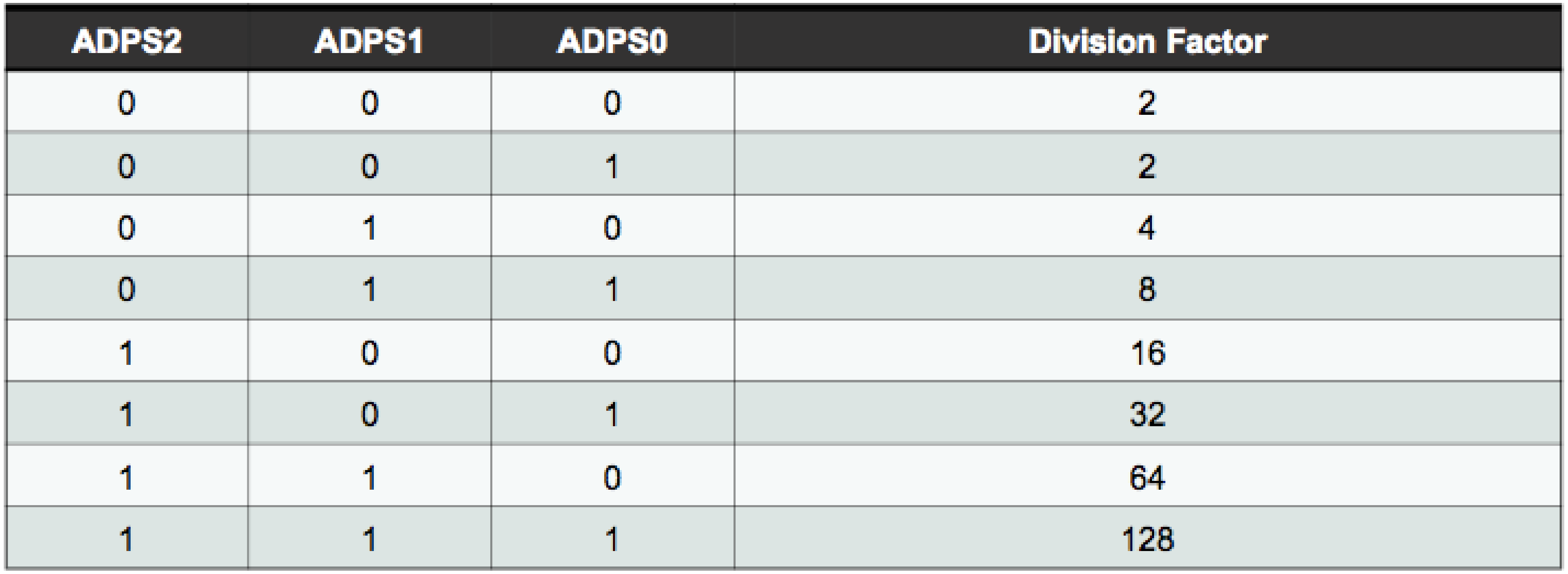
- Bit 4 – ADIF: ADC Interrupt Flag: This bit is set when an ADC conversion completes and the Data Registers are updated.
- Bit 3 – ADIE: ADC Interrupt Enable: When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.
- Bits 2:0 – ADPS[2:0]: ADC Pre-scaler Select Bits



Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADC Pre-scaler Selection

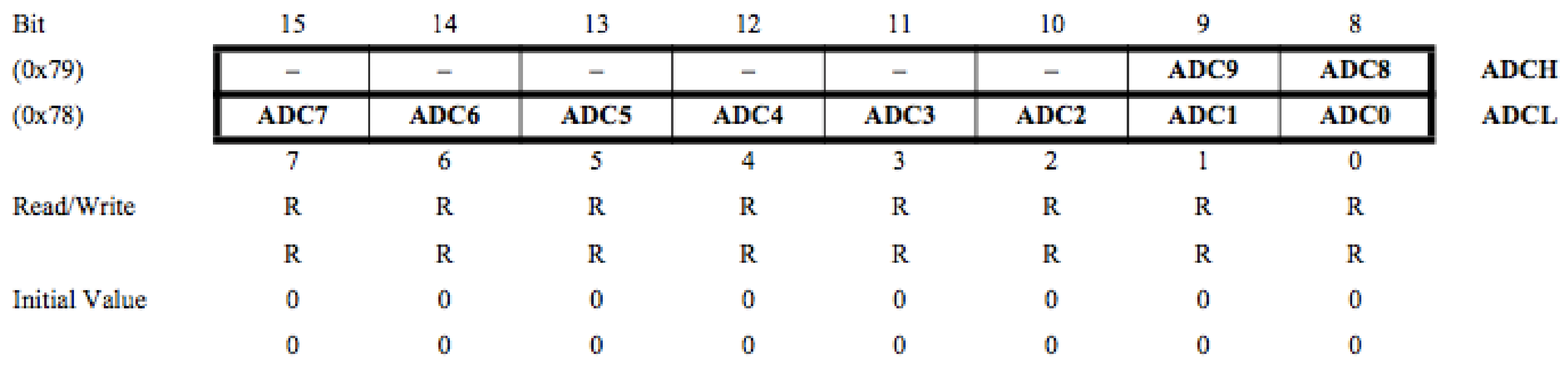
Table 24-5. ADC Prescaler Selections



ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

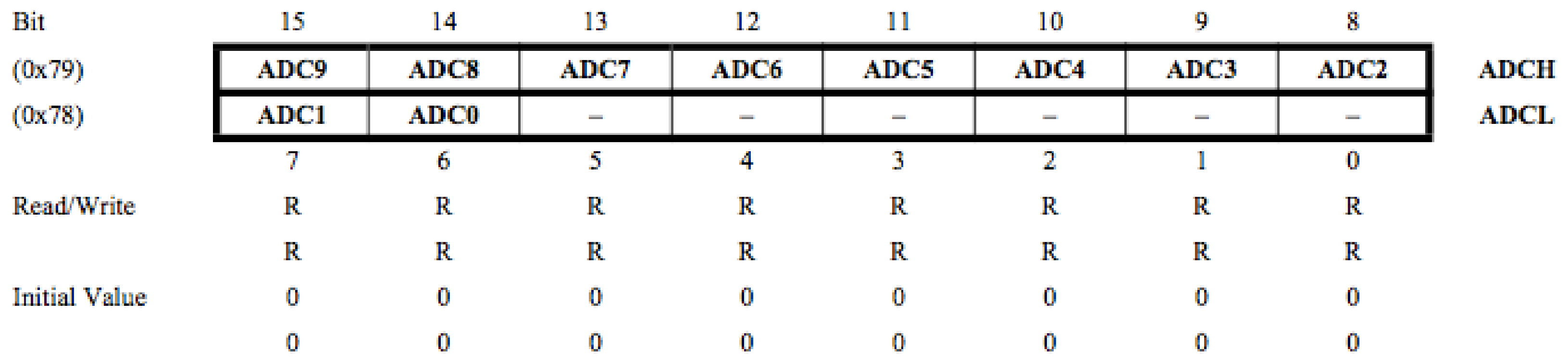
ADCL and ADCH – The ADC Data Register

24.9.3.1 ADLAR = 0



Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

24.9.3.2 ADLAR = 1



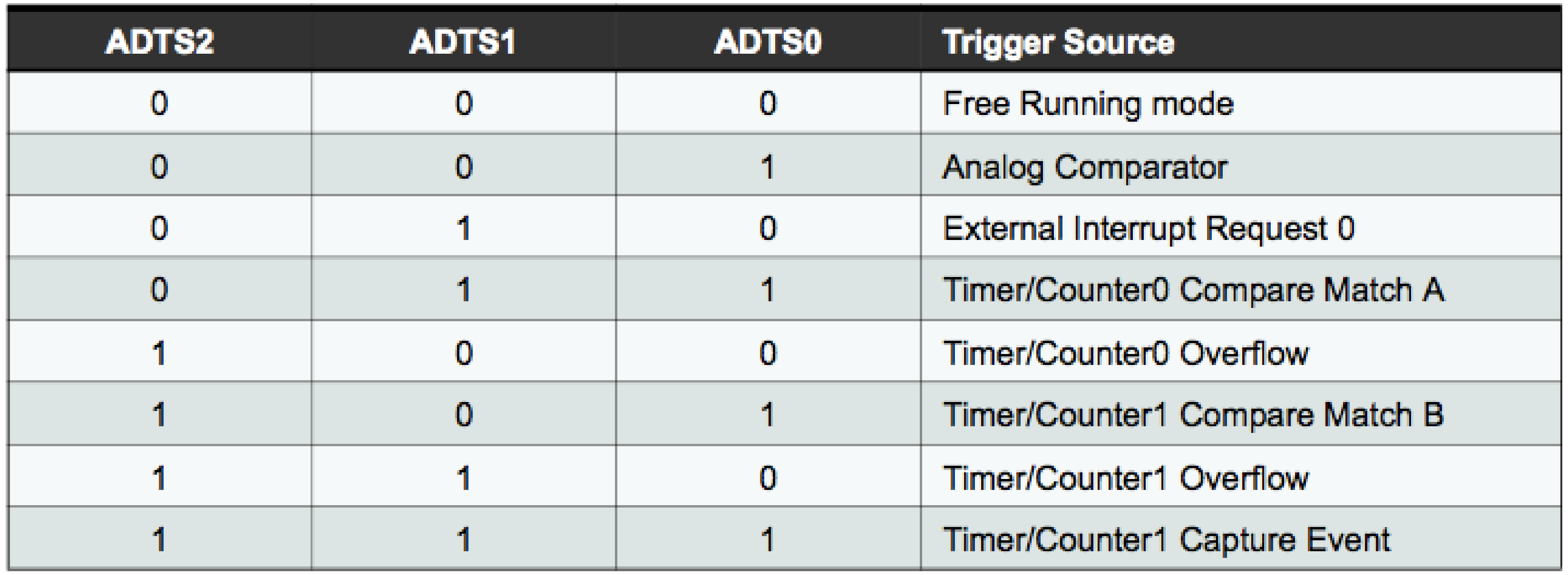
Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

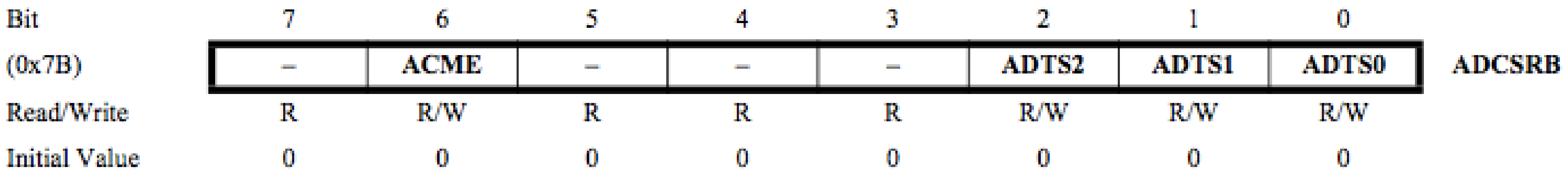
ADCSRB – ADC Control and Status Register B

- Bit 2:0 – ADTS[2:0]: ADC Auto Trigger Source

Table 24-6. ADC Auto Trigger Source Selections



ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event



Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Coding Example: Initialize ADC

```
void adc_init() {

// AREF = AVcc
ADMUX = (1<<REFS0);

// ADC Enable and pre-scaler of 128
// 16000000/128 = 125000
//104 us per conversion
ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

}
```

Coding Example: Reading ADC Value

```
uint16_t adc_read(uint8_t ch) {

// select the corresponding channel 0~7
// ANDing with ’7’ will always keep the value of ‘ch’ between 0 and 7
ch &= 0b00000111;// AND operation with 7
ADMUX = (ADMUX & 0xF8)|ch; // clears the bottom 3 bits before ORing

// start single conversion
// write ’1’ to ADSC
ADCSRA |= (1<<ADSC);

// wait for conversion to complete
// ADSC becomes ’0’ again
// till then, run loop continuously
while(ADCSRA & (1<<ADSC));

// ADC is predefined to hold the 10-bit conversion value
return (ADC);

}
```