Lecture #20

Analog Inputs

18-348 Embedded System Engineering

Philip Koopman

Wednesday, 30-March-2016

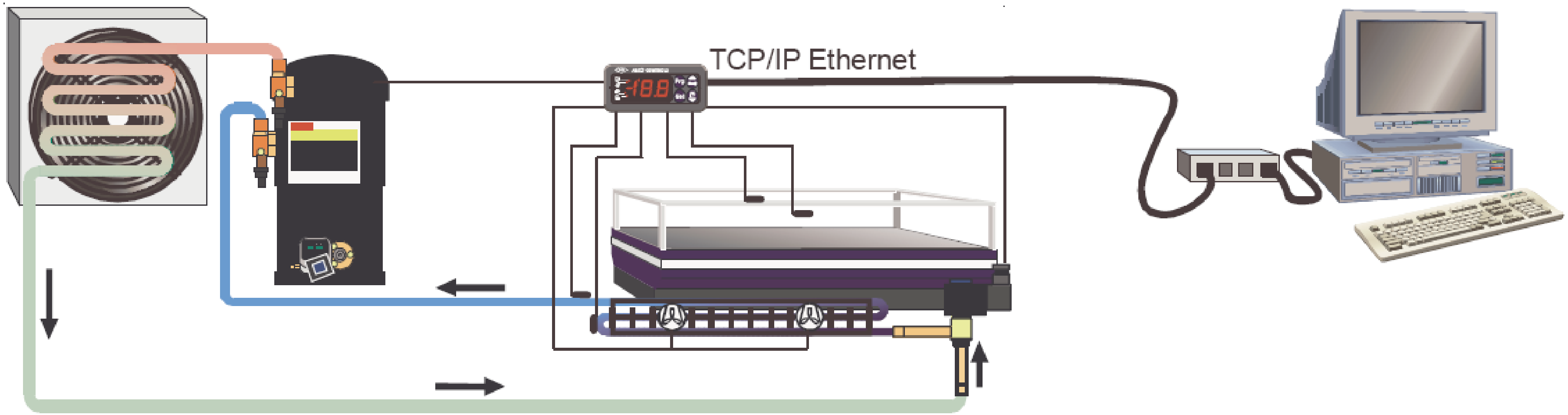
Electrical&Computer ENGINEERING

Carnegie Mellon

Commercial HVAC Is Networked For Monitoring

Electronic Controllers for Cold Rooms and Display Cases

The controllers of the popular EC-series are used for control of display cases, cold rooms, compressor packs and condensers. The controllers fulfill several control tasks simultaneously, as superheat and temperature control, defrost, compressor, condenser and condensing unit control.



[Emerson Climate Technology – Copeland]

Centralized Monitoring For Retail Cooling

- **How much does it cost to replace a freezer case full of spoiled seafood?**
  - How much energy can you save by detecting inefficient compressors?
  - Processes millions of automated alerts per year

[Emerson Climate Technology – Copeland]

Emerson ProAct Service Center, Atlanta GA

Where Are We Now?

- **Where we’ve been:**
  - Analog Output

- **Where we’re going today:**
  - Analog Input
  - Filters

- **Where we’re going next:**
  - Human I/O and misc I/O stuff
  - Gentle introduction to control
  - Embedded networks
  - …
  - Test #2; last project

Preview

- **Analog to Digital Converters ADC; A/D**
  - General components – analog mux, signal sample, ADC
  - Operating parameters

- **Types of ADCs**
  - Flash ADC
  - Successive Approximation ADC

- **Engineering consideration**
  - Triggering sample and determining when sample is done
  - Resolution, accuracy
  - Brief reminder of Nyquist Criterion

- **A *very* gentle introduction to filters**

Why Is Digital Data A Good Thing?

- **Once encoded as a digital number, noise doesn’t matter**
  - The value is exact and can be encoded as a binary value
  - It won’t change over time or with noise – it is an exact value
  - You can have as many bits of precision as you want

- **It can be stored and transmitted**
  - Storing a digital value is easy – series of ones and zeros rather than analog value
  - Sending it down a network requires only ability to see a zero or one, not analog value

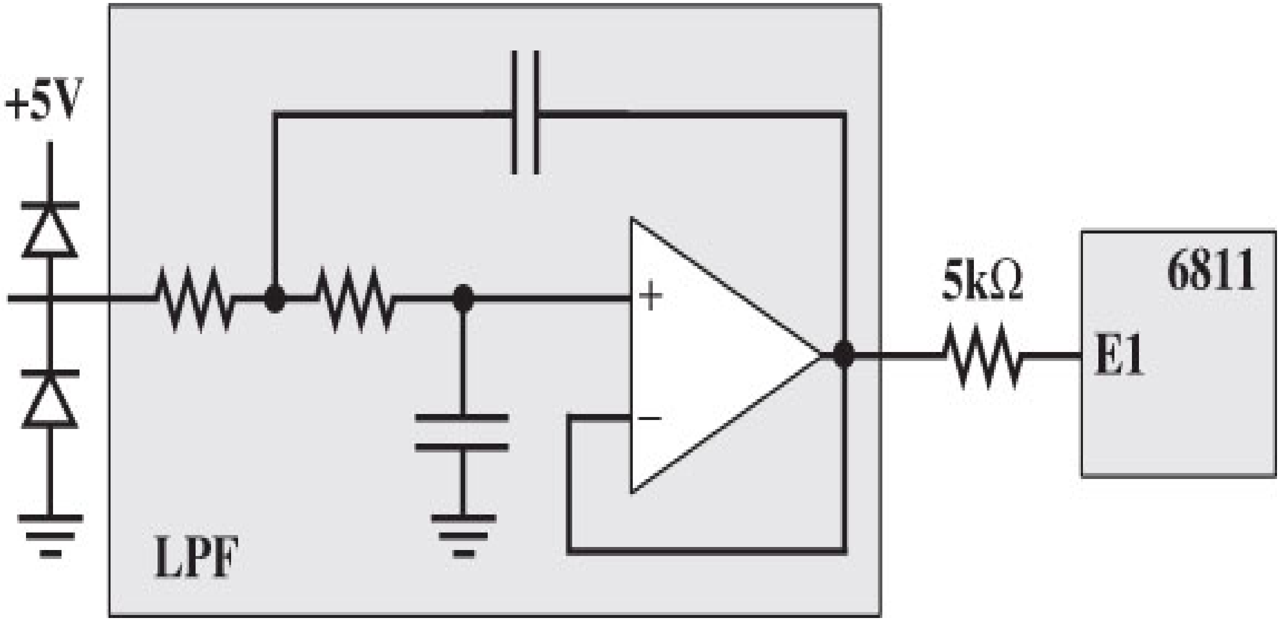
- **Error correction works**
  - Can use error detection to tell if value is corrupted (can’t really do that on analog values)
  - Can use error correction in some cases to compensate for errors

- **But, the world is analog, so…**
  - Need to get the data from analog to digital as accurately as possible

Analog Input Protection

- **Usually analog inputs must have compatible voltage with CPU**
  - E.g., analog input must be between 0V and +5V
  - If input is outside this voltage, it can damage CPU
  - Often every company has different approaches – but end goal is the same
- **Related considerations**
  - Protection against static electricity discharge
  - Protection against reverse power supply voltage (e.g. battery in backward)
  - Protection against power supply over-voltage (e.g., 24V jump start voltage)

Figure 11.70
One possible approach for protecting the analog input.

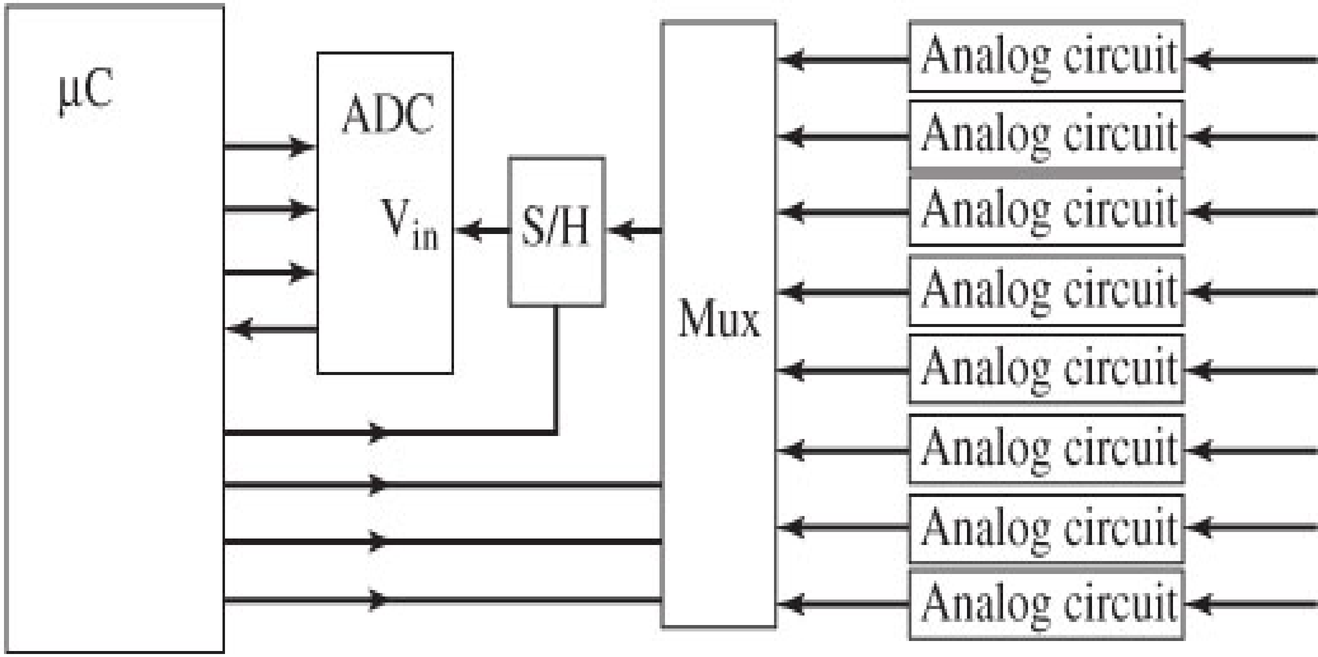


[Valvano]

Analog To Digital Conversion

- **Analog inputs permit CPU to sense external world**
  - Note that analog inputs are usually changing values (‘moving targets’)
- **Analog input components:**
  - ADC – Analog to Digital Converter
  - S/H – Sample and Hold
  - MUX – analog multiplex to share use of ADC among many inputs

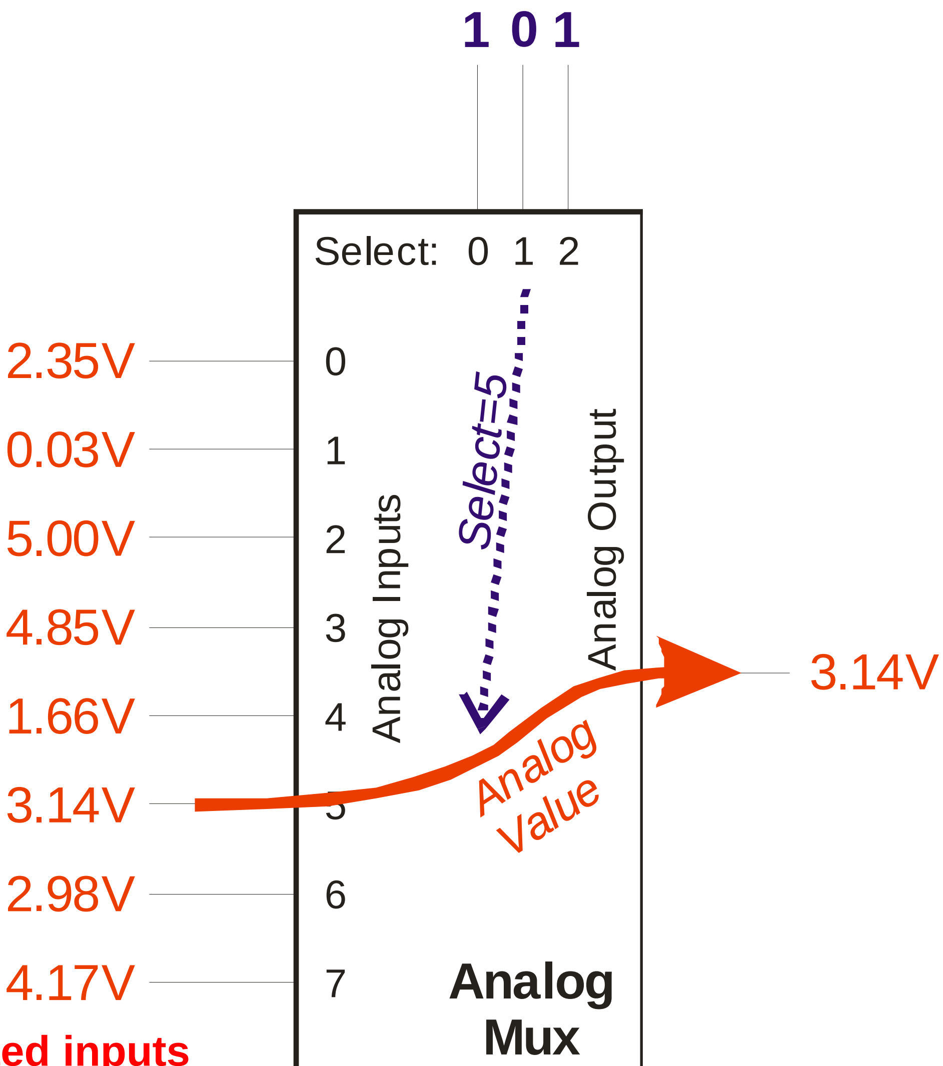
Figure 11.64
Together, the analog circuit, multiplexer, S/H, and ADC convert external signals to digital form.



[Valvano]

Analog Mux

- Same idea as digital mux, except passes analog values
- Important parameter: number of channels (often 8 to 12 channels)

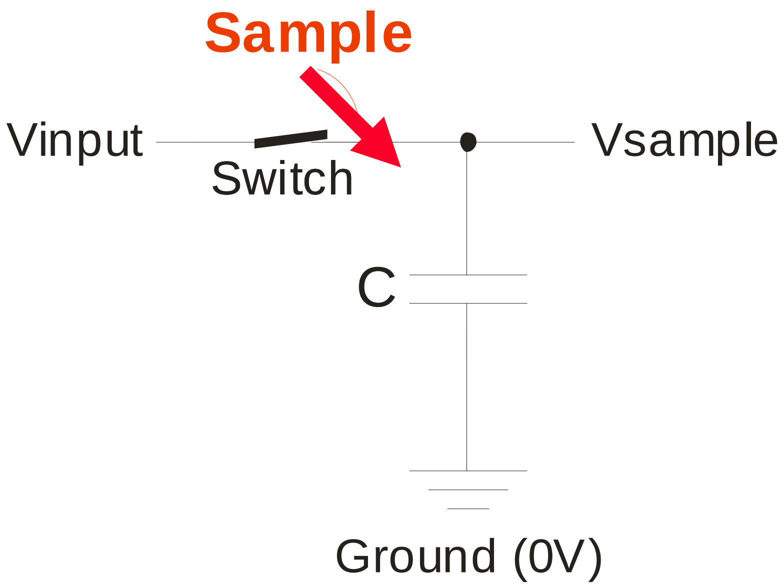


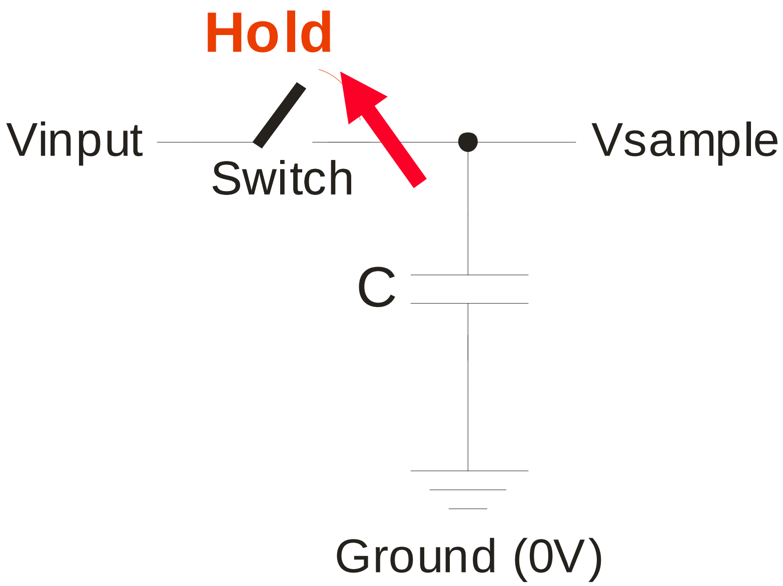
Various unrelated inputs
Mux picks one input to sample

Sample & Hold

- **Need to keep a constant analog value for measurement even if input changes**
- **Approach: charge an internal capacitor to match input voltage and hold value**
  - S/H stage 1: input amplifier – precharge capacitor close to input voltage
  - S/H stage 2: connect capacitor to input to exactly match voltage
  - S/H stage 3: hold charge, isolating it from input after the sample is complete…

    … and then run ADC process





ADC Specifications

(examples are from course processor)

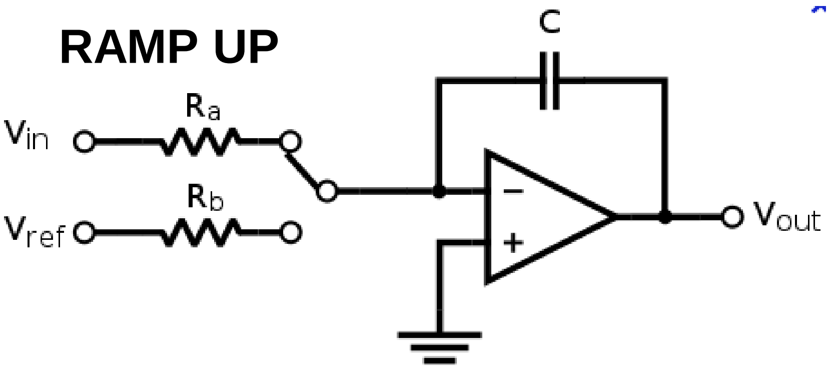
- **Resolution – how many bits of input value?**
  - Often 8 to 12 bits of resolution
  - For k bits of resolution, each quantum of input value is ~$V_{range}/2^k$

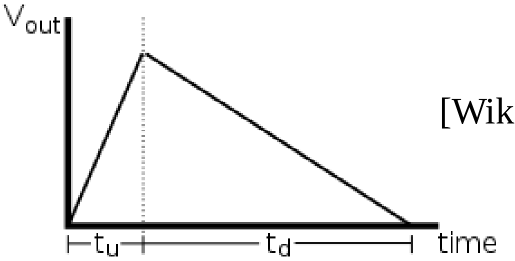
- **Conversion time**
  - Time from sample available (S/H complete) to answer
    - Might vary depending on value!
  - E.g., “7 µsec, 10-bit single conversion time”

- **Conversion approach**
  - Multiple approaches to conversion, each with cost/performance tradeoffs
  - “Flash” – fast but expensive
  - “Successive Approximation” – (SAR) – slower, but cheaper
  - Other methods as well…

Integrating ADC

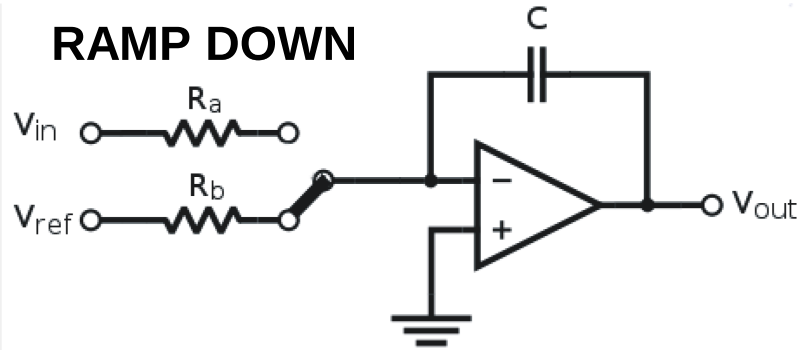
- **Two-part operation**
  1. Ramp up capacitor voltage from Vin input through Ra for fixed amount of time
  2. Bleed down capacitor to <u>negative</u> Vref voltage through Rb and measure time it takes for op amp to switch to its + input (ground value)





[Wikipedia]

- **Compare time to ramp up with time to ramp down**
  - Time ratio tells you the voltage
  - “Dual slope” so that capacitor variation is averaged out by using the ramping ratio



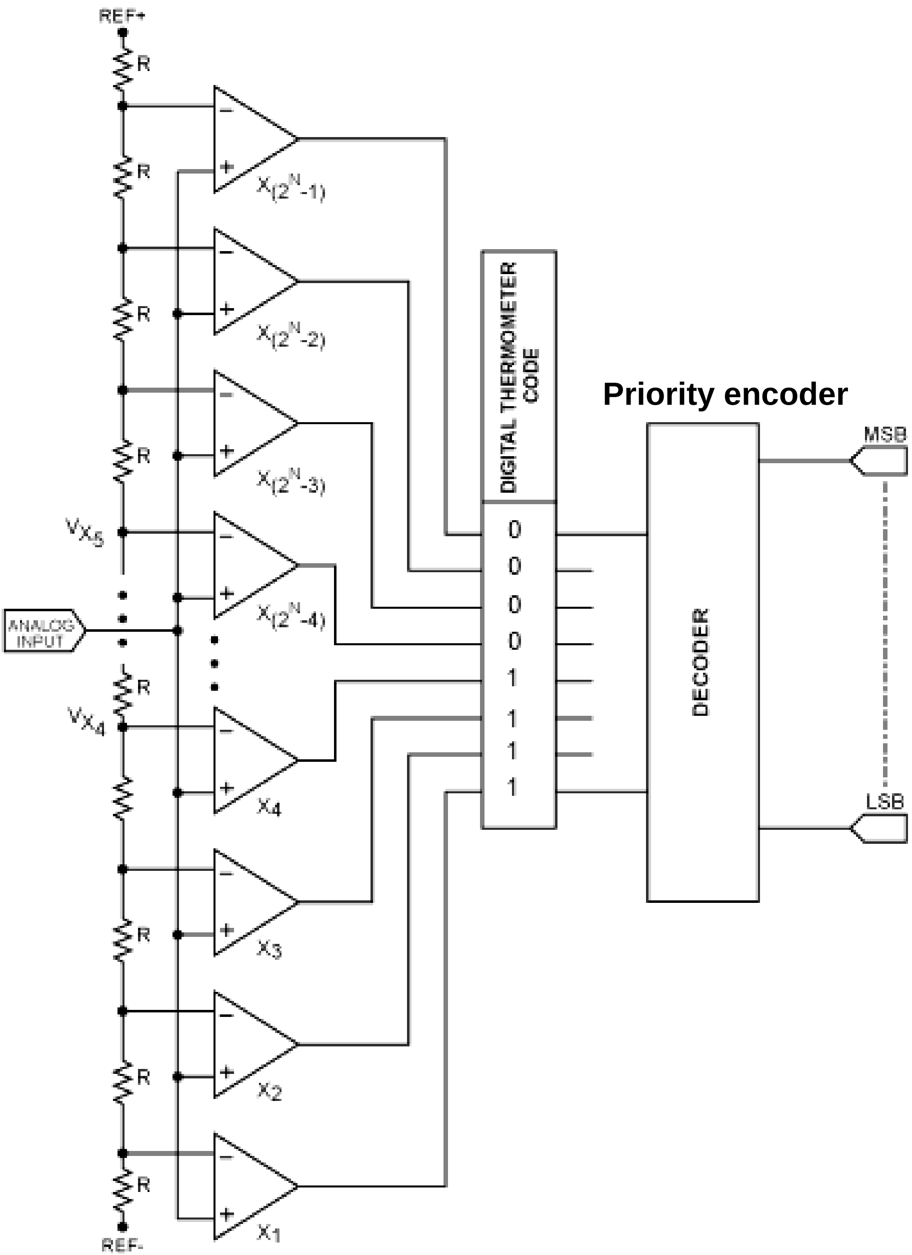
- **Cheap**
  - And slow!

Flash ADC

- **Produces value after one propagation delay through circuit**
  - Stack of resistors provides uniformly spaced voltage points from gnd to V_{ref}
  - All resistors same value R
  - Every op amp provides saturated comparison value:
    - “lower than me” = 0
    - “higher than me” = 1
  - Priority encoder gives position of highest “1” value
    - That’s the digital version of voltage

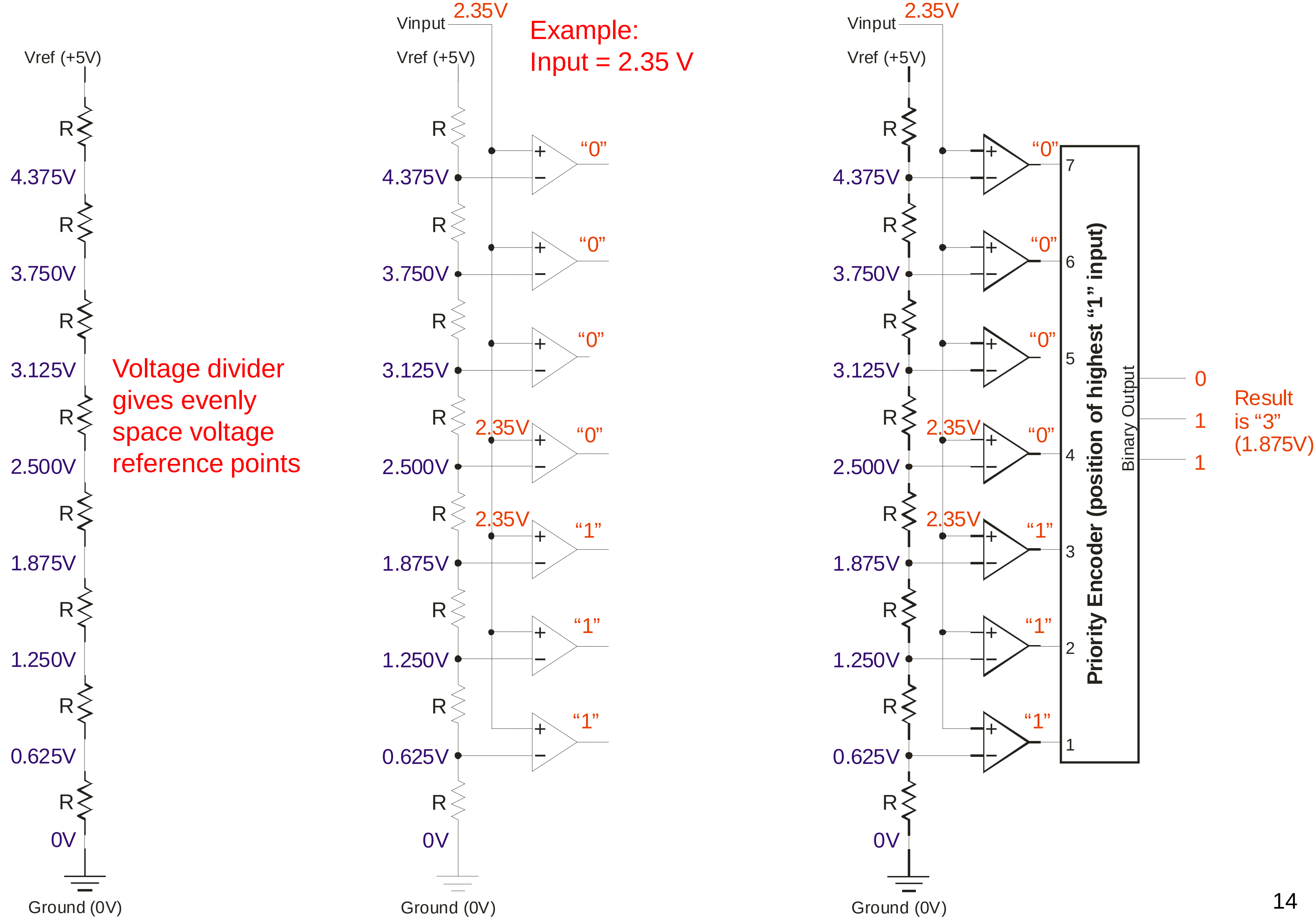
(optimizations possible, but this is the basic idea)

- **Cost: 2^b resistors + 2^b-1 op amps**
  - \$9 to \$70+ depending on specs
- **Time: almost constant time**



http://www.maxim-ic.com/appnotes.cfm/appnote_number/810/

How A Flash ADC Works



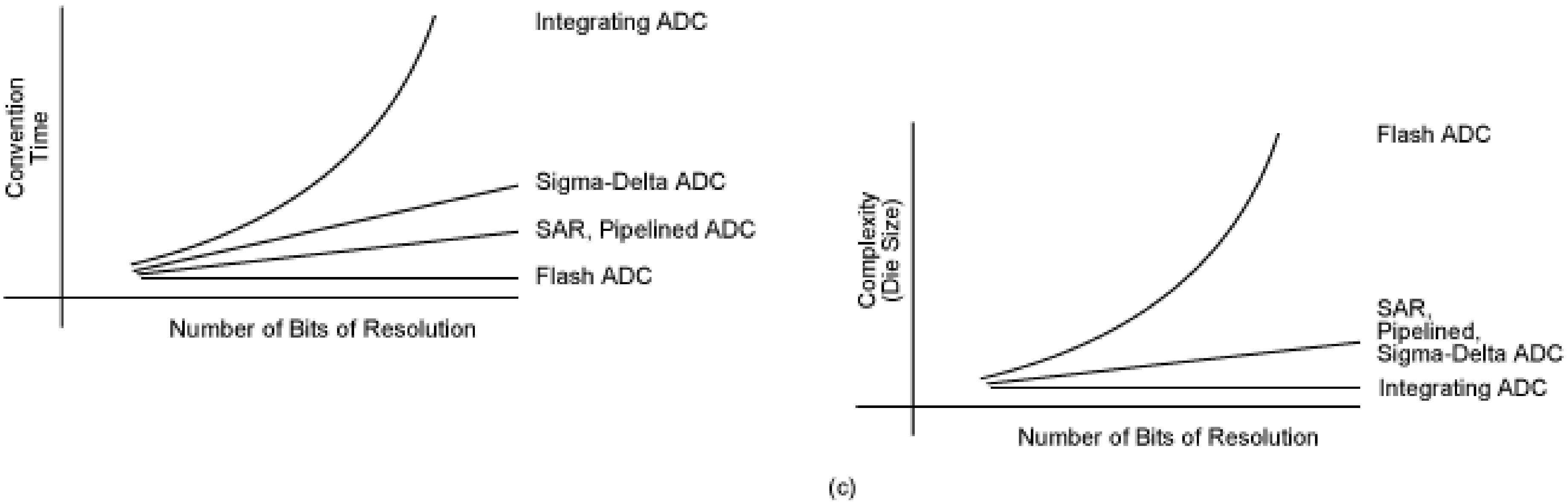
Voltage divider gives evenly space voltage reference points

Example:
Input = 2.35 V

Result is “3” (1.875V)

Why Use Different Converter Types?

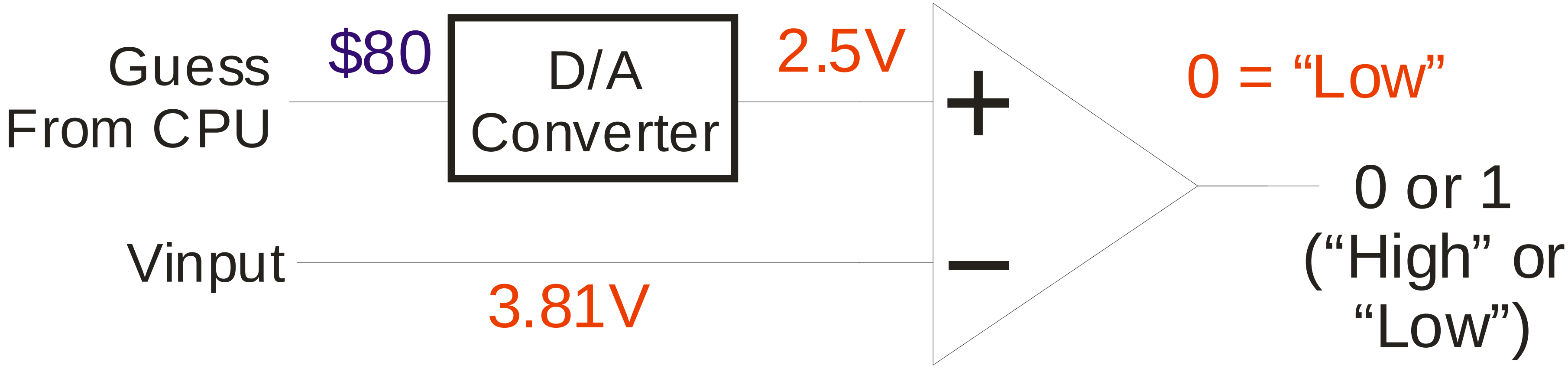
- **Flash ADC is fast, but expensive**
  - Need 2^k copies of sample logic (resistor, op amp) for k bit sample
  - Often only good to 8 bits because resistors have to be closely matched in value
    - 16-bit Flash ADC are common, but quite expensive
- **Other approaches provide slower conversion time for lower cost**
  - SAR is a common compromise points



http://www.maxim-ic.com/appnotes.cfm/appnote_number/810/

SAR – Successive Approximation Register

- **Majority of market – good cost/performance tradeoff**
  - Uses a DAC to generate an analog value – a “guess”
  - Uses Op Amp to compare “guess” to actual value
  - Generates “Dout” for whether guess is high or low
  - Use binary search technique to find voltage – one iteration per bit of output
  - Usually gives 8 to 16 bits of useful output

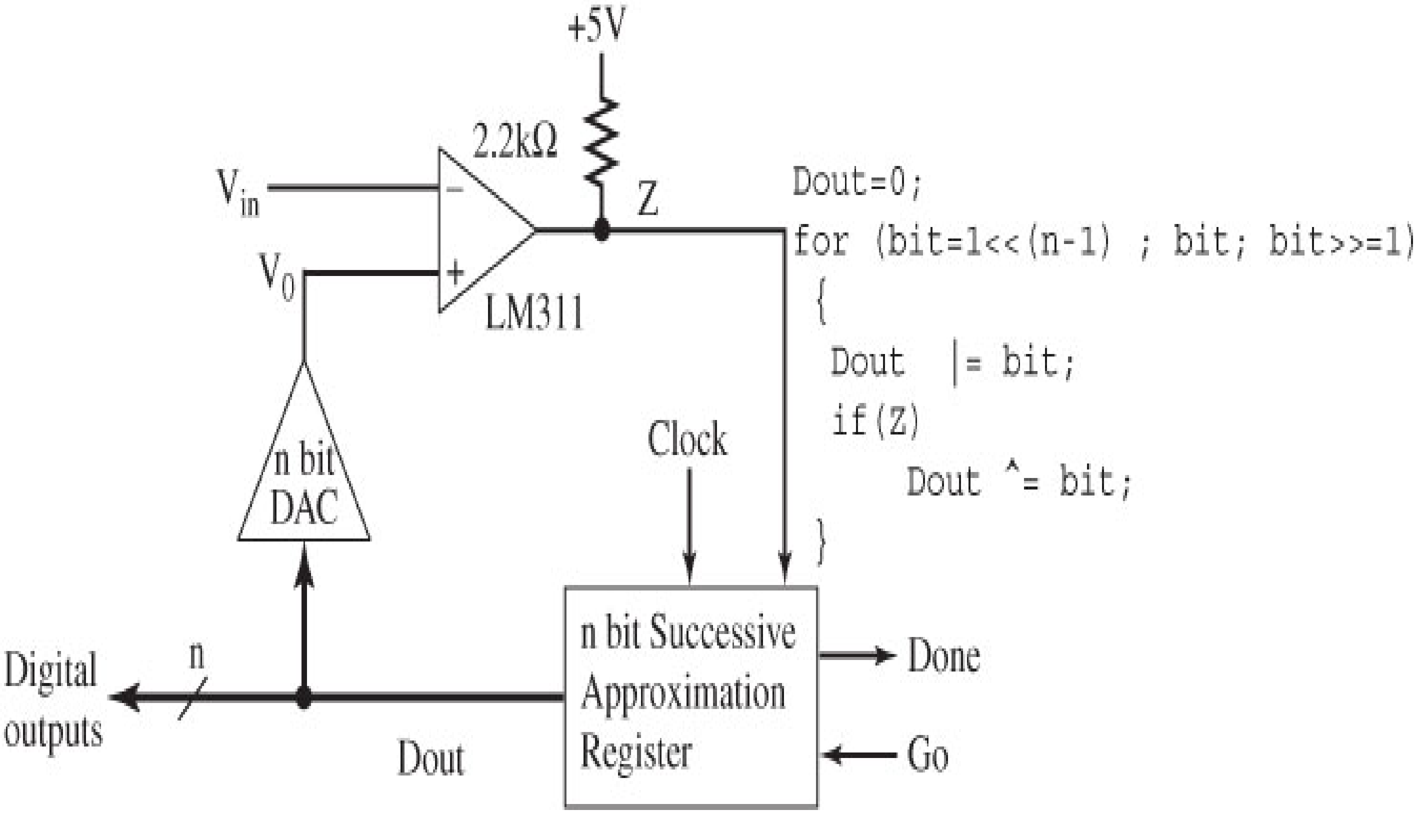


- **Think of this as:**
  - DAC creates what is logically a single point on the flash ADC voltage divider
  - Instead of trying all possible values at once, use binary search of values

Successive Approximation ADC Operation

- **Guess a value**
  - Do binary search based on guess
  - Time to find value for b bits is b steps
  - Hardware cost: $\log_2 b$ (inside the DAC) for b bits

Figure 11.54 Successive approximation ADC.



```
Dout=0;
for (bit=1<<(n-1) ; bit; bit>>=1)
 {
  Dout  |= bit;
  if(Z)
      Dout ^= bit;
 }
```

[Valvano]

Course Chip A/D

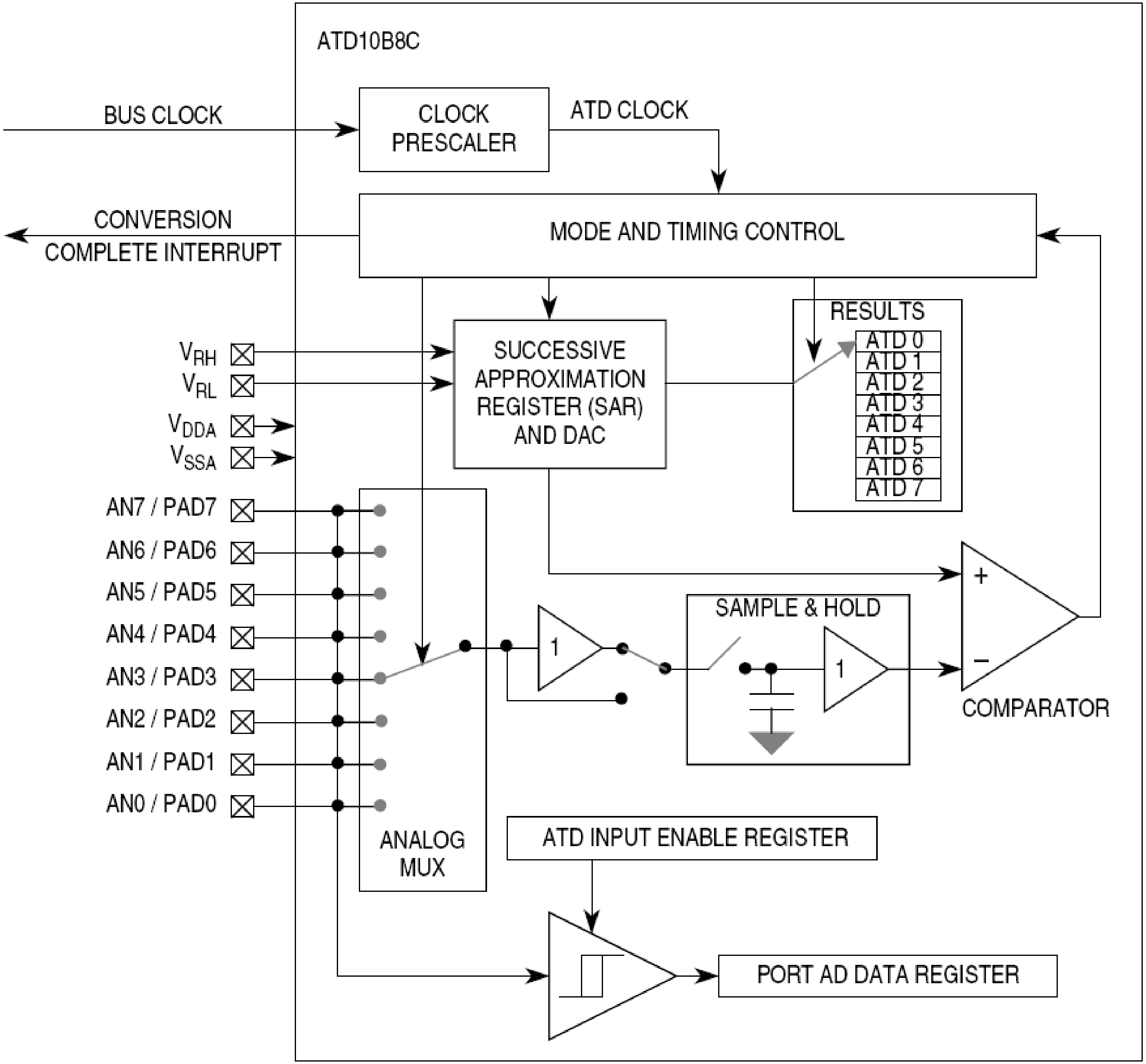


Figure 8-1. ATD10B8C Block Diagram

[Freescale]

ADC Usage

- **Multi-channel polled A/D reading**

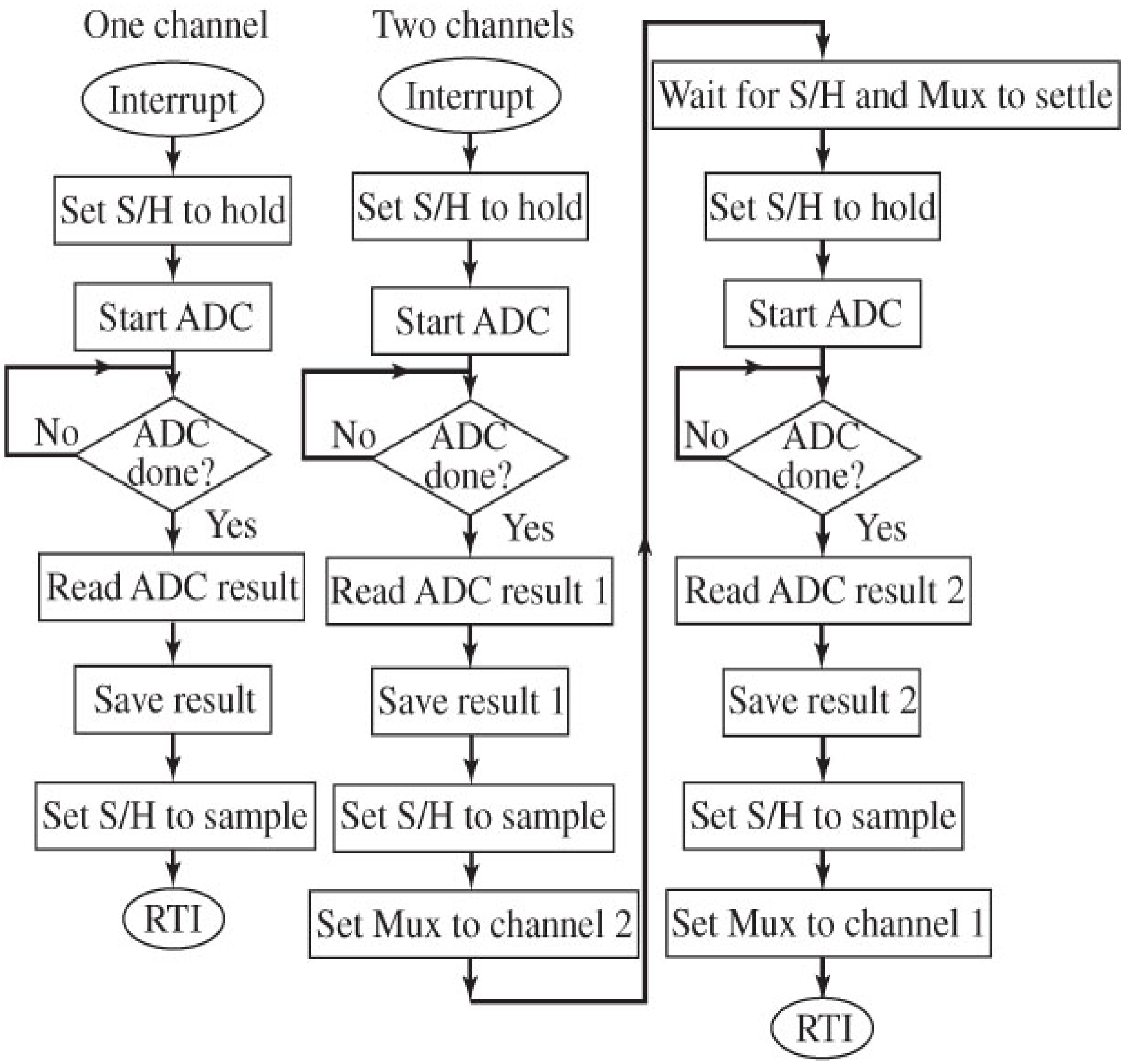


Figure 11.66
Flowcharts for ADC interrupt software when a S/H is needed.

[Valvano]

ADC Hardware Interface

- **Two types of interfaces**
  - Tell ADC “Go” and data available some fixed time later
  - Tell ADC “Go” and monitor “done” bit (or get interrupt) to know when it is ready

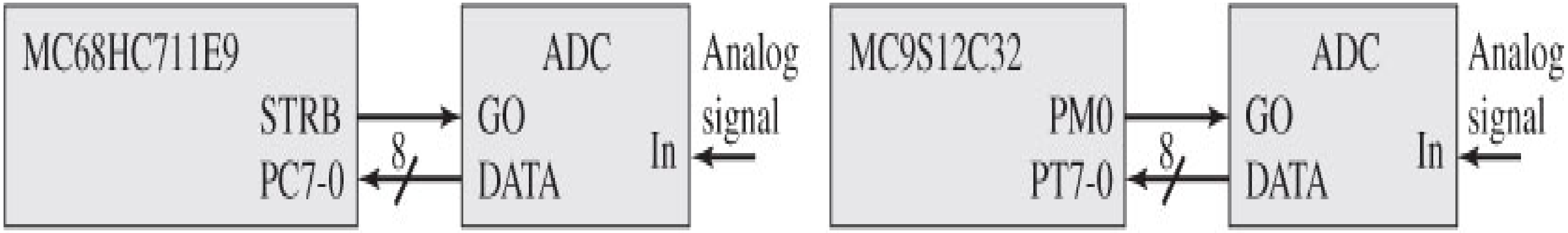
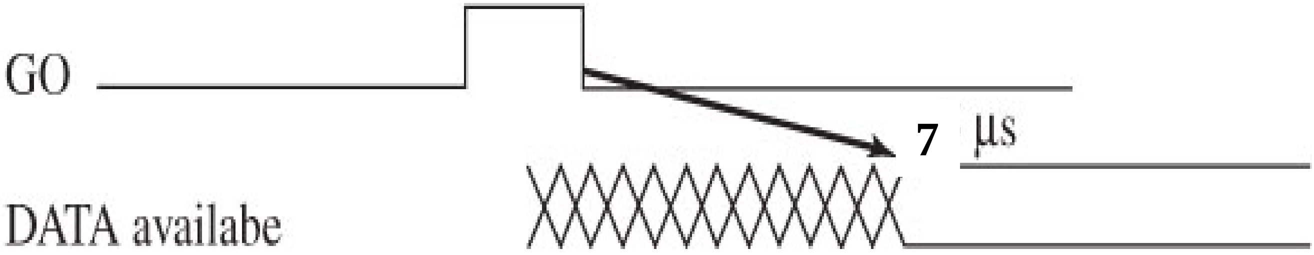


Figure 3.13
Hardware interface between an ADC and the microcomputer.

Figure 3.14
Timing diagram for an ADC interface.



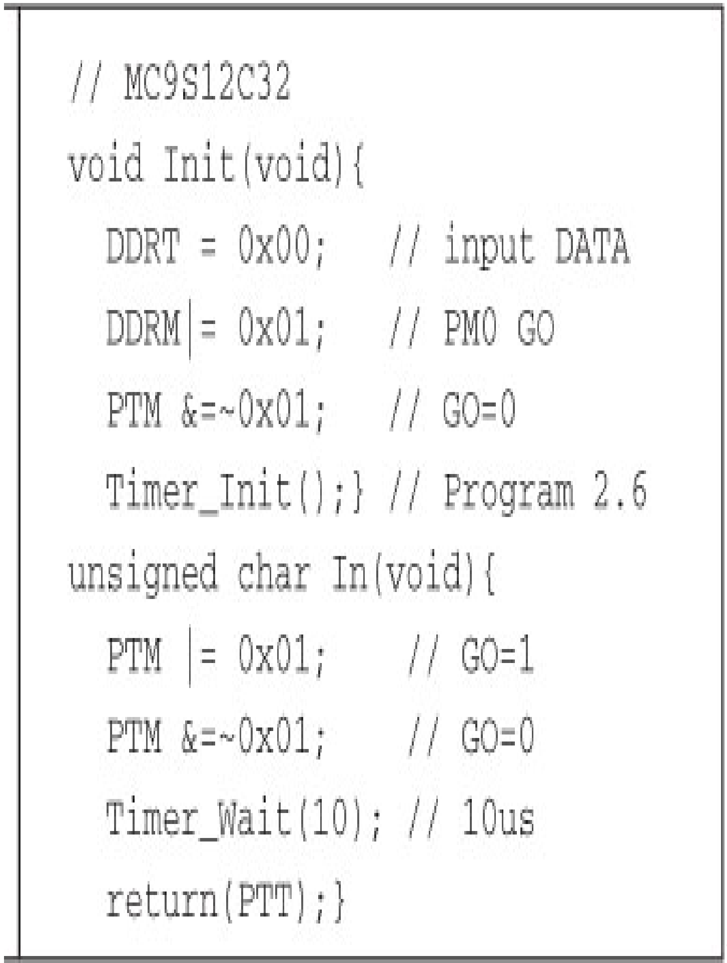
[Valvano]

ADC Example

- **“Timer_Init” is program 2.10 in [Valvano]**
  - Enables TCNT; sets TCNT to 1 microsecond tick rate
- **“Timer_Wait” is also program 2.10**
  - Waits for N TCNT increments   (Timer_Wait(10) waits for 10 microsecs)

Program 3.6

C language routines to initialize and read from an ADC.



```
// MC9S12C32
void Init(void){
  DDRT = 0x00;    // input DATA
  DDRM|= 0x01;    // PM0 GO
  PTM &=~0x01;    // GO=0
  Timer_Init();} // Program 2.6
unsigned char In(void){
  PTM |= 0x01;     // GO=1
  PTM &=~0x01;     // GO=0
  Timer_Wait(10); // 10us
  return(PTT);}
```

[Valvano]

ADC Sharing – Mutex Approach

- **ADC is a shared resource**
  - Need to worry about concurrency!
  - Here are some common approaches

- **One approach:**
  Use a mutex to control access to ADC
  - Task waits for ADC to be free, then gets a sample
  - Stalls tasks if they get unlucky
  - Complicates real time scheduling
    - Priority inversion
    - Longer blocking time for each task based on resource
  - Gets complicated if you want to ensure fair (round robin) or prioritized access

  - Generally this is an “event triggered” approach
    - ADC sample based on the “event” of a task wanting a sample

ADC Sharing – Time Triggered Approach

- **Instead, use:**
  periodic background ADC sampling task and “mailboxes”
  - Background task does round robin ADC for all possible inputs
    - Possibly at different rates depending on input time constants
    - Think “multi rate cooperative scheduling” for inputs
  - Results of ADC are put in a global variable array, e.g., ADCout[8]
  - Other tasks just use value on ADCout[i] for Analog sample
  - “Time Triggered” because samples happen based on time, not based on task request

- **Pros:**
  - Blocking time limited to interrupts masked while reading ADCout[i] value
  - Tasks don’t have to request an ADC value and wait for conversion
  - Much simpler to design and get right for real time operation
  - Creates periodic samples, which are what you need for filtering (see later slides)

- **Cons:**
  - Samples might be stale (what if you look at ADCout just before next sample?)
  - Not all samples will be processed – only most recent sample
  - Need a way to deal with errors (mark ADCout[i] as “stale”)

- **Overall, this is a good way to go**

How Good Are Samples?

- **Sample quality depends on hardware quality**
  - "Resolution" – size of one measurement quantum
    - e.g., 1/256 of V_{ref} for 8-bit
  - "Accuracy" – how close measured value is to actual value
    - Can't do better than resolution
    - Very often do several bits worse
    - For example, 12-bit ADC might only give 8-bits of "clean" output data

- **Example: course processor**
  - Data sheet summary says:
    8 bit or 10 bit resolution with "7 microsecond" conversion time
  - But – take a look at Appendix A Electrical Characteristics…

Conversion Time

- If we just wait 10 µsec, will that code always work?

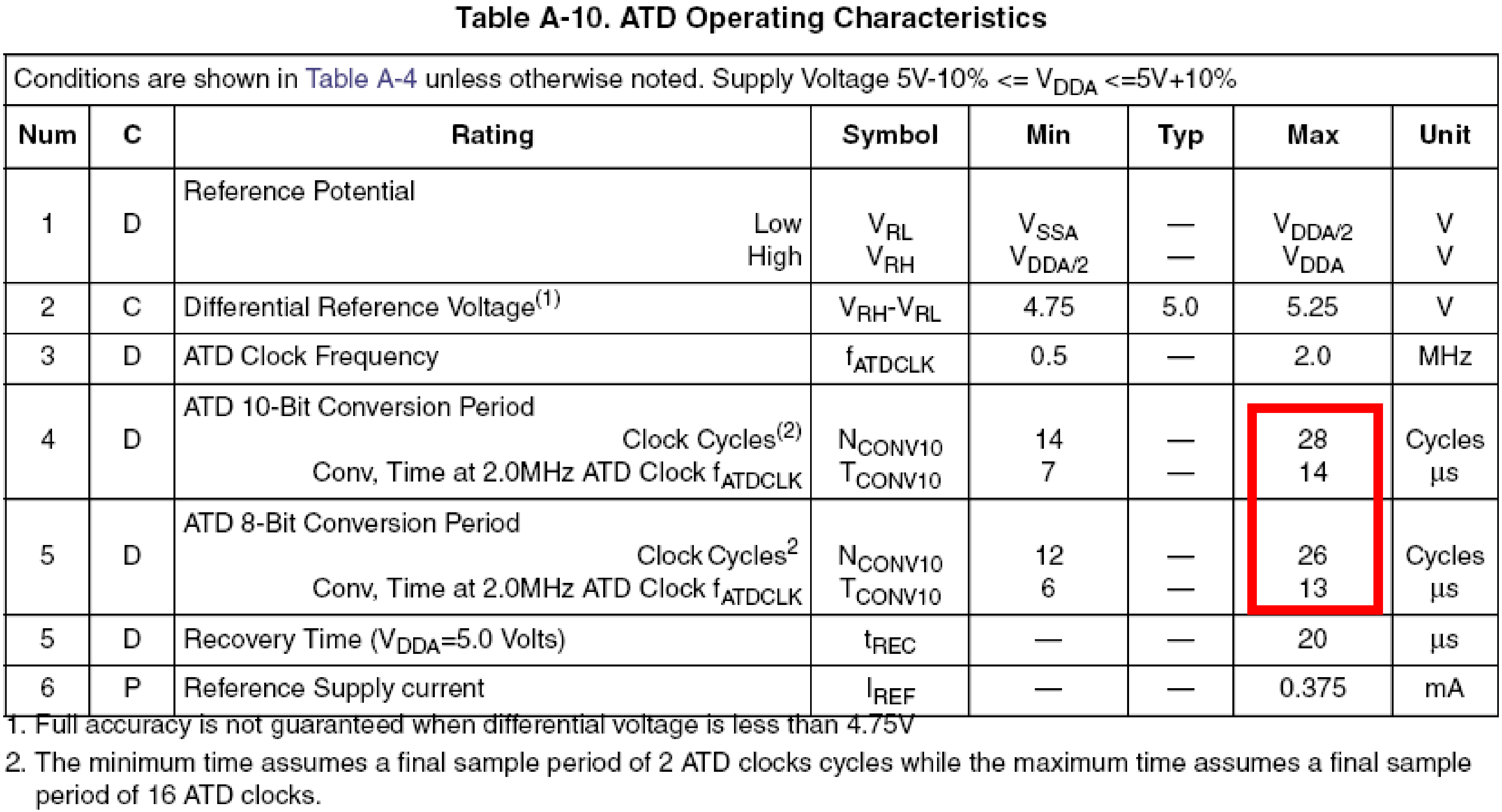


Table A-10. ATD Operating Characteristics

Conditions are shown in Table A-4 unless otherwise noted. Supply Voltage 5V-10% <= V_{DDA} <=5V+10%

Num	C	Rating	Symbol	Min	Typ	Max	Unit
1	D	Reference Potential Low High	V_{RL} V_{RH}	V_{SSA} $V_{DDA/2}$	— —	$V_{DDA/2}$ V_{DDA}	V V
2	C	Differential Reference Voltage[1]	V_{RH}-V_{RL}	4.75	5.0	5.25	V
3	D	ATD Clock Frequency	f_{ATDCLK}	0.5	—	2.0	MHz
4	D	ATD 10-Bit Conversion Period Clock Cycles[2] Conv, Time at 2.0MHz ATD Clock f_{ATDCLK}	N_{CONV10} T_{CONV10}	14 7	— —	28 14	Cycles µs
5	D	ATD 8-Bit Conversion Period Clock Cycles[2] Conv, Time at 2.0MHz ATD Clock f_{ATDCLK}	N_{CONV10} T_{CONV10}	12 6	— —	26 13	Cycles µs
5	D	Recovery Time (V_{DDA}=5.0 Volts)	t_{REC}	—	—	20	µs
6	P	Reference Supply current	I_{REF}	—	—	0.375	mA

[1] Full accuracy is not guaranteed when differential voltage is less than 4.75V

[2] The minimum time assumes a final sample period of 2 ATD clocks cycles while the maximum time assumes a final sample period of 16 ATD clocks.

[Freescale]

Accuracy

- **This specification is a pretty good A/D**
  - Do we believe the spec in real-world applications?
  - Assumes perfect signal input – doesn’t account for noise in signal itself

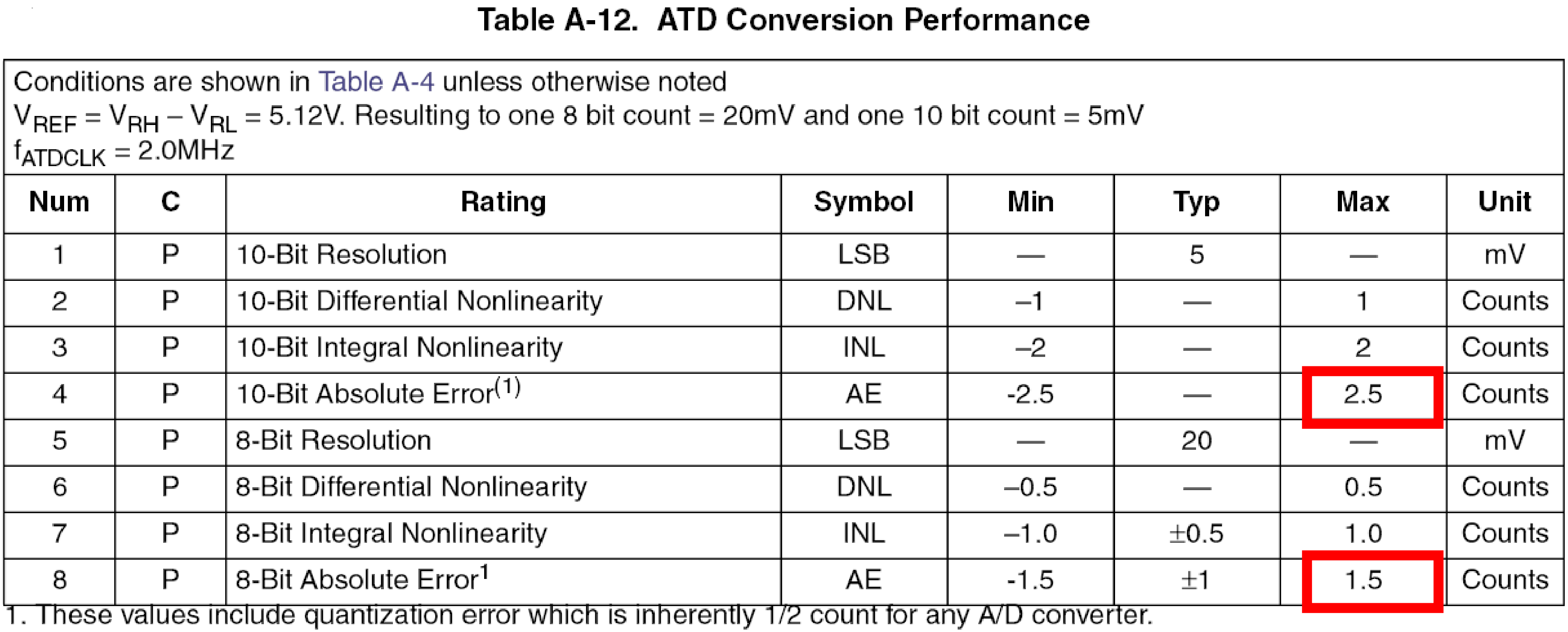


Table A-12. ATD Conversion Performance

Conditions are shown in Table A-4 unless otherwise noted
$V_{REF} = V_{RH} - V_{RL}$ = 5.12V. Resulting to one 8 bit count = 20mV and one 10 bit count = 5mV
f_{ATDCLK} = 2.0MHz

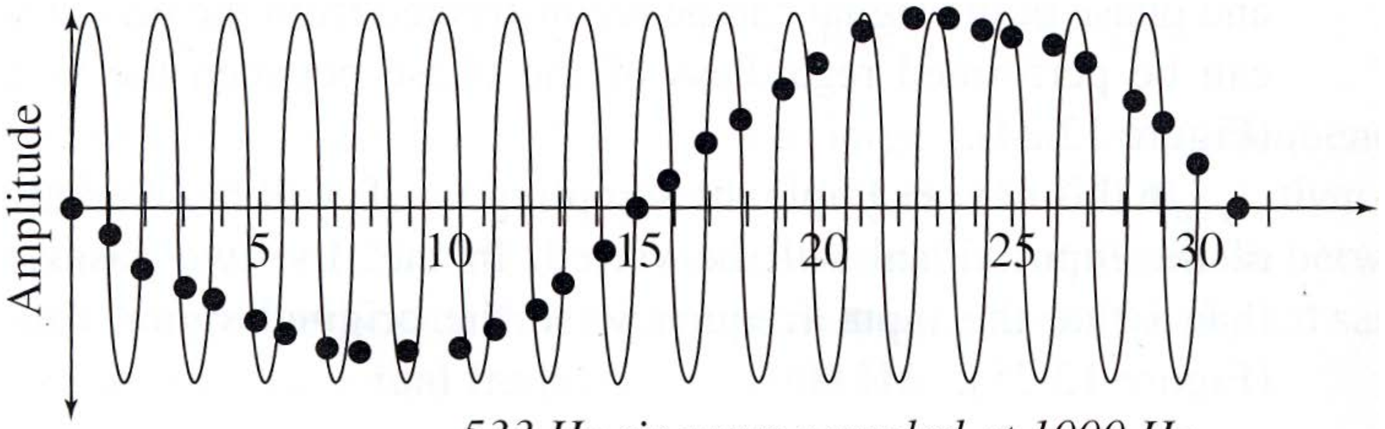
Num	C	Rating	Symbol	Min	Typ	Max	Unit
1	P	10-Bit Resolution	LSB	—	5	—	mV
2	P	10-Bit Differential Nonlinearity	DNL	–1	—	1	Counts
3	P	10-Bit Integral Nonlinearity	INL	–2	—	2	Counts
4	P	10-Bit Absolute Error[(1)]	AE	-2.5	—	2.5	Counts
5	P	8-Bit Resolution	LSB	—	20	—	mV
6	P	8-Bit Differential Nonlinearity	DNL	–0.5	—	0.5	Counts
7	P	8-Bit Integral Nonlinearity	INL	–1.0	±0.5	1.0	Counts
8	P	8-Bit Absolute Error[1]	AE	-1.5	±1	1.5	Counts

1. These values include quantization error which is inherently 1/2 count for any A/D converter.

[Freescale]

How Fast Do We Need To Sample?

- **Nyquist criterion – sample *at least* 2x signal frequency**
  - Sampling too slowly results in alias problems
  - Sampling at least 2x faster avoids aliasing …. But isn’t perfect



533 Hz sin wave sampled at 1000 Hz.

[Valvano]

- **Practical sampling rates**
  - For non-sinusoidal signals, need *at least 2x* of highest frequency component you want to reproduce (that’s Nyquist criterion – which is a theoretical bound)
  - But, also usually want to sample perhaps 5x-10x faster than the *basic* signal, if you can, to get a good smooth curve

A Very Gentle Introduction To Filters

- **Analog inputs have noise**
  - Quantization noise
  - Noise in the signal itself
  - Noise in the conversion process (which is an analog process)

- **What if we want to get a better value?**
  - In other words …
    what's the input version of the R/C analog low pass output filter?
  - Well, it's a filter! (but for inputs we want a *digital* filter)

- **Input filtering**
  - Look at multiple sequential inputs and …
    … use that information to make a better guess as to the actual signal value
  - Generally requires storing most recent *n* samples and combining them

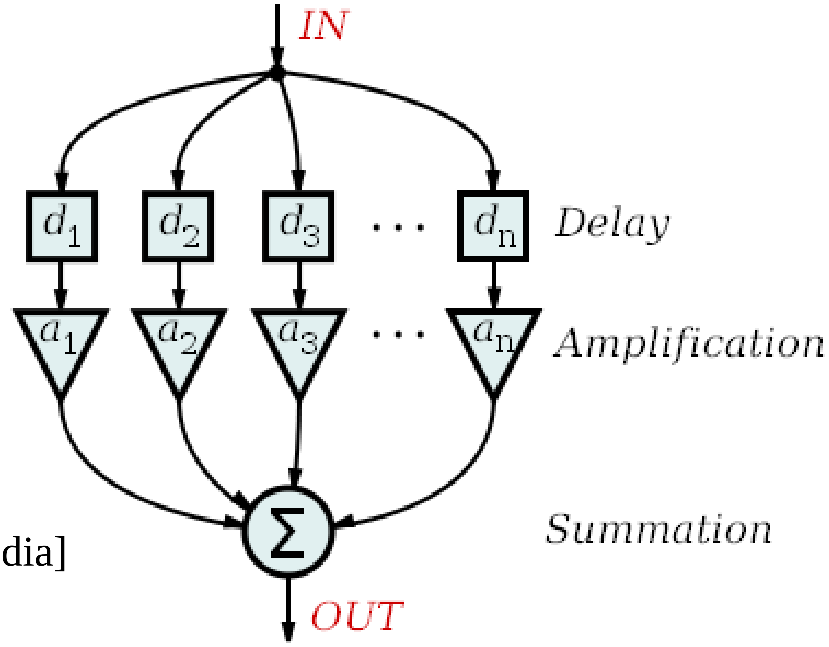
Finite Impulse Response (FIR) Filter

- **Simple FIR: moving average   (most frequently used filter in practice)**
  - Take most recent *n* samples and average them
  - Filters out noise by reducing effect of any single errant data point

  + ➔ Simple, intuitive, fast; this is usually what you see in embedded systems

  – ➔ Single outlier can make it way off; time lag proportional to # samples
  - Embellishments:
    - Weighted average (weight newest samples more than old ones)
    - Drop outliers (obviously way off data points; e.g., drop highest & lowest)

- **Generalized FIR**
  - Each input has a gain and is summed to create output
  - Simple moving average: all the gains are the same



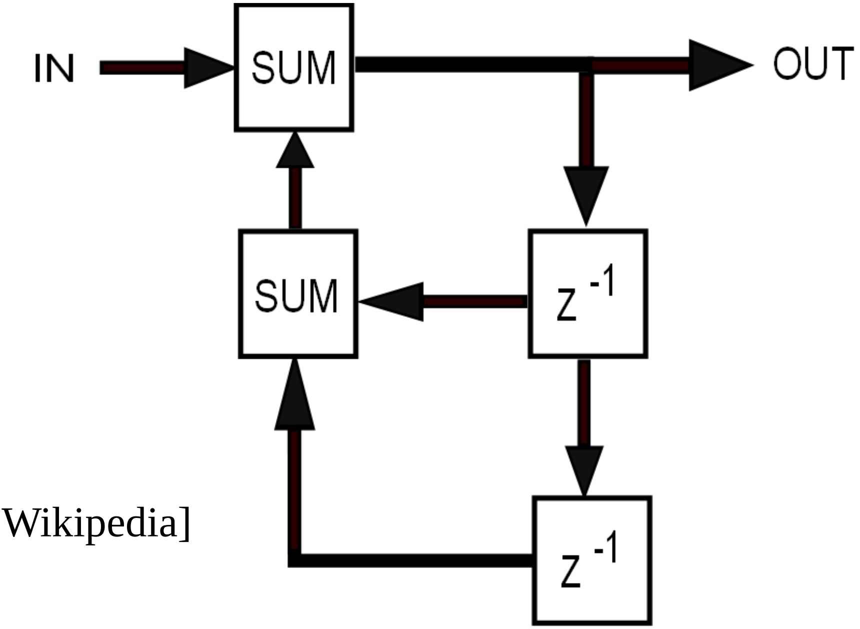
[Wikipedia]

Infinite Impulse Response (IIR) Filter

- **Note that FIR is only based on input values**
  - Values older than n samples cannot affect output

- **IIR includes feedback from output**
  - That means that *all previous inputs* affect output (to some degree)
  - In other words, even a single pulse input will affect infinitely many outputs
    - Simple example: R/C low pass filter is an analog IIR
    - But, in practice, effect decays to nearly zero over time
  - In general, IIR has some form of "integrator" that remembers all previous history

- **IIR implementation:**
  - Take a signals course!
  - Bring some math with you
  - 18-290 applies here



[Wikipedia]

Filtering Tradeoffs

- **Latency**
  - The more samples in your filter, the longer it takes to see an output
  - Simplistically, you have to wait about *n* samples to see a filtered output
  - That means you need to sample at least *n* times faster than system time constant!

- **Memory**
  - All those samples have to be stored in RAM, which might be limited

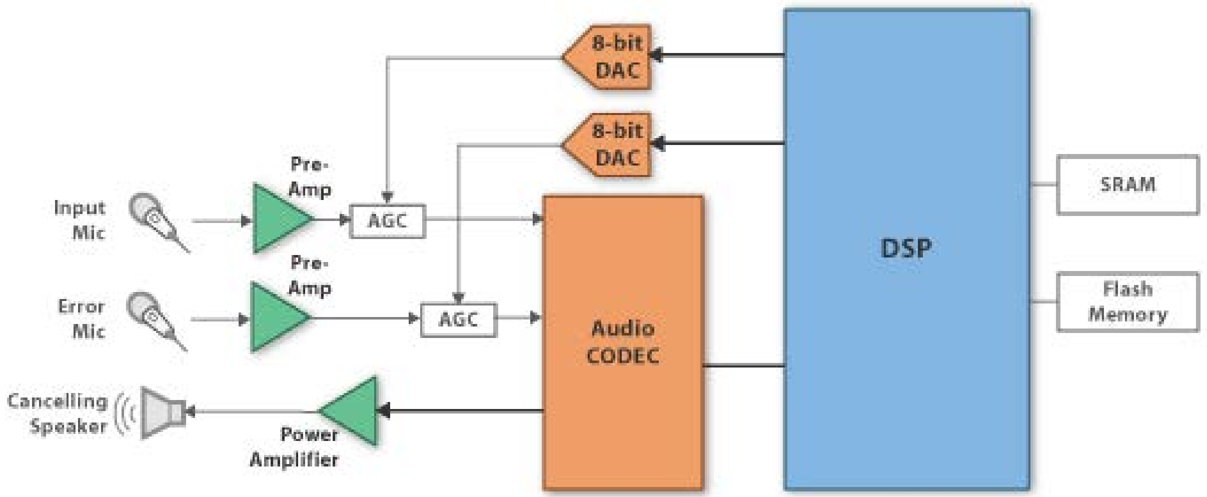
- **Analysis complexity**
  - You need to know the math to use a complex filter
    - (Or, get the code from the Internet – and hope it actually works!)

- **Run-time complexity**
  - More complex math means more computation
  - But, complex filters often give more accurate estimates of real input value

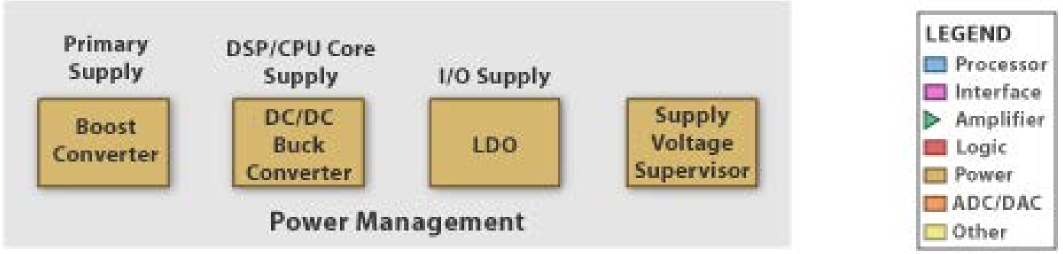
A Word About DSP Chips

- **DSP = Digital Signal Processing**
  - Special chips to support DSP
  - They are also a microcontroller (e.g., have a program counter)
  - Hardware optimized for DSP tasks – filtering, FFT, etc.

Active Noise Cancellation (ANC)



“Codec” = “Coder/Decoder”; just an A/D, D/A pair

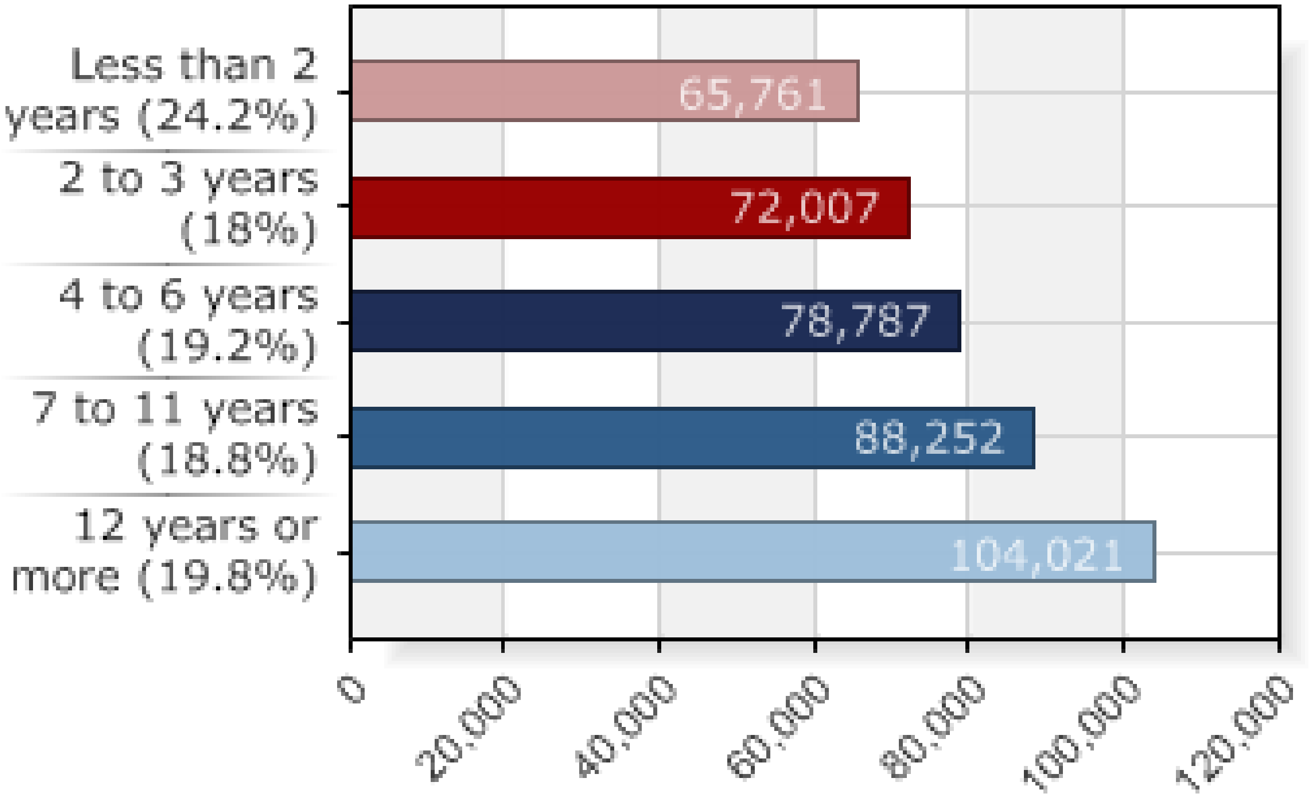


[TI]

A Word About Careers

- **Embedded System Engineers, 2013 data:**

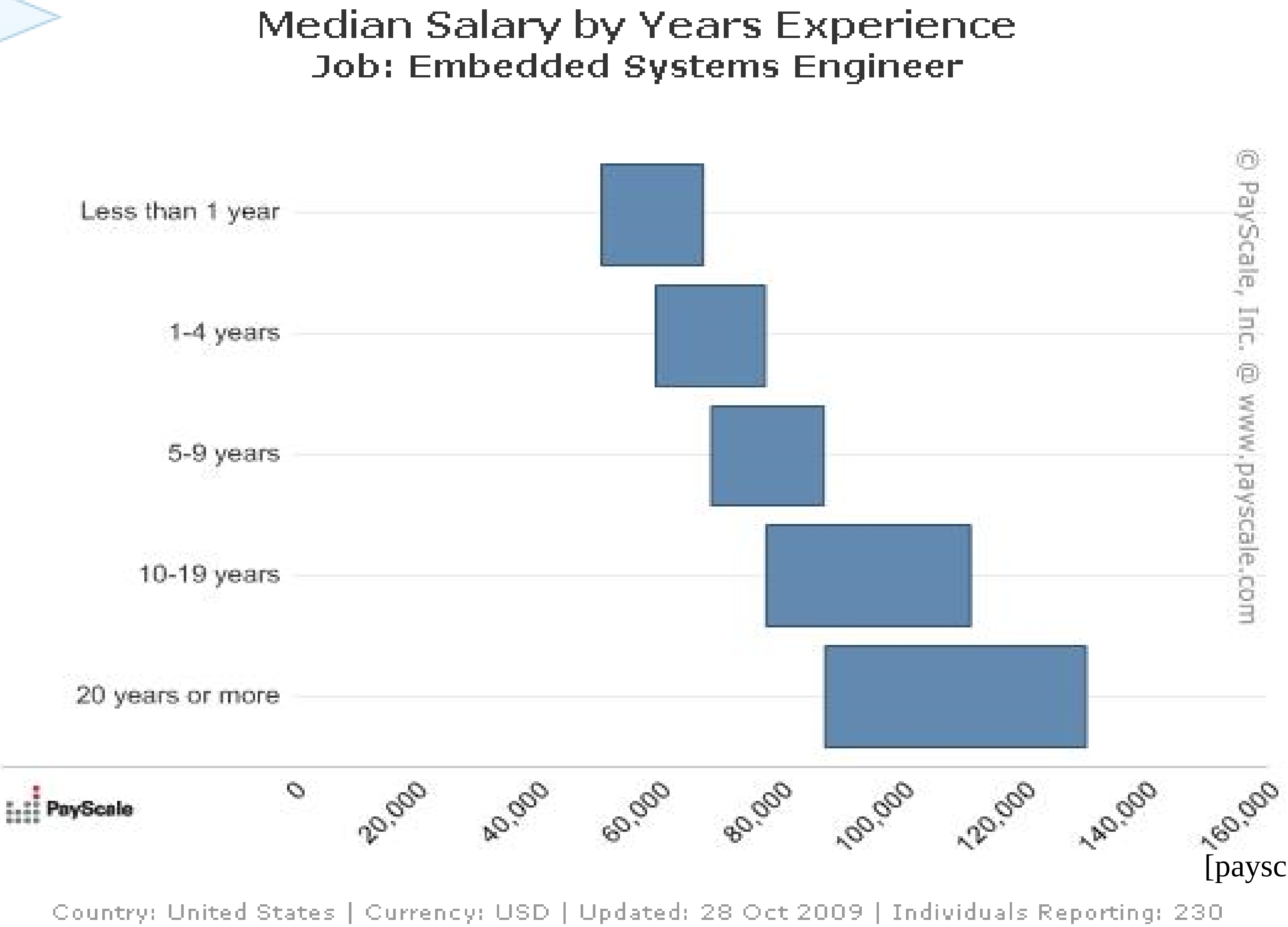
Average Total Cash By Experience



Number Reporting: 64 Currency: U.S. Dollar (USD)

[payscale.com]

Embedded System Engineer Salary Profile



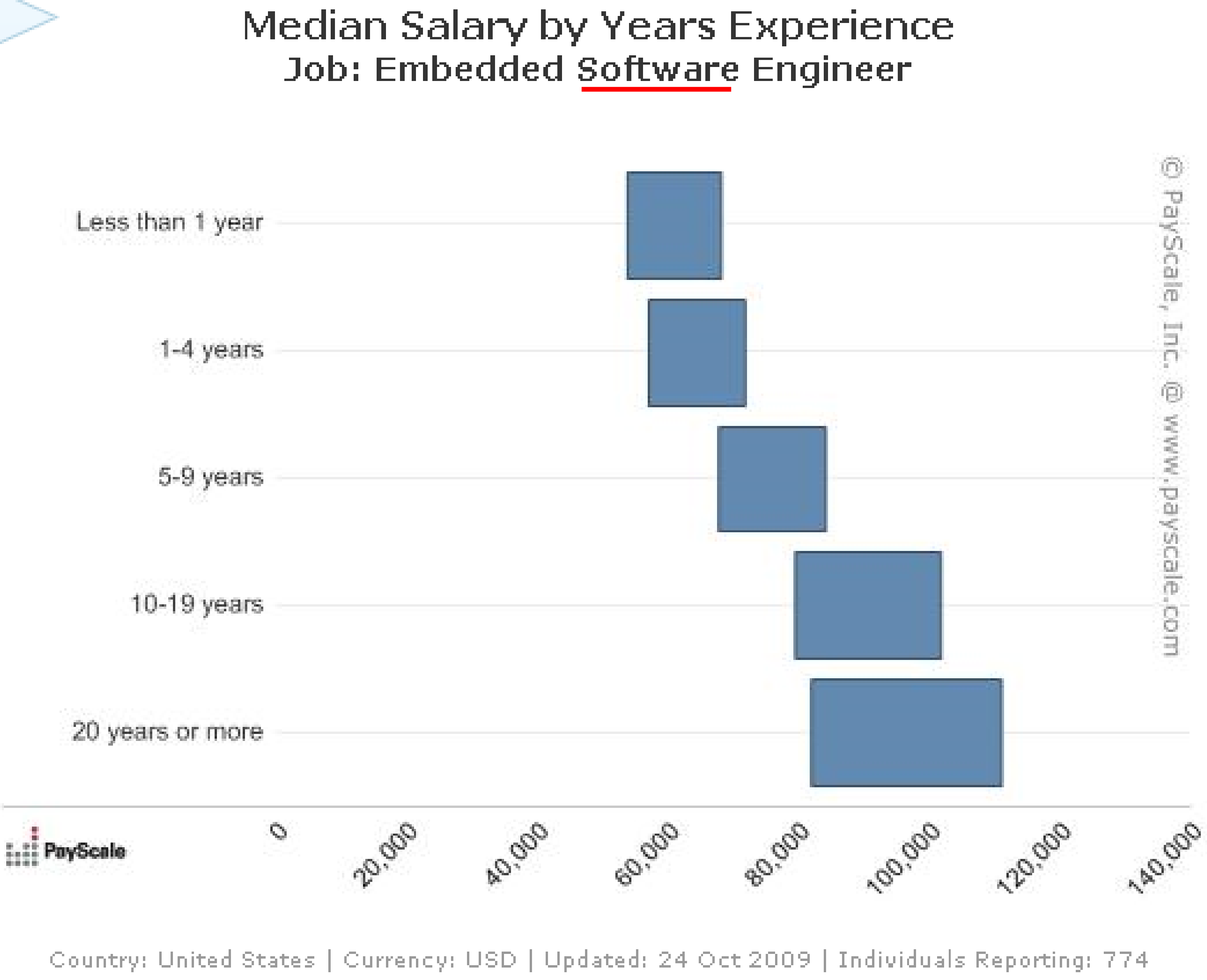
Median Salary by Years Experience

Job: Embedded Systems Engineer

Country: United States | Currency: USD | Updated: 28 Oct 2009 | Individuals Reporting: 230

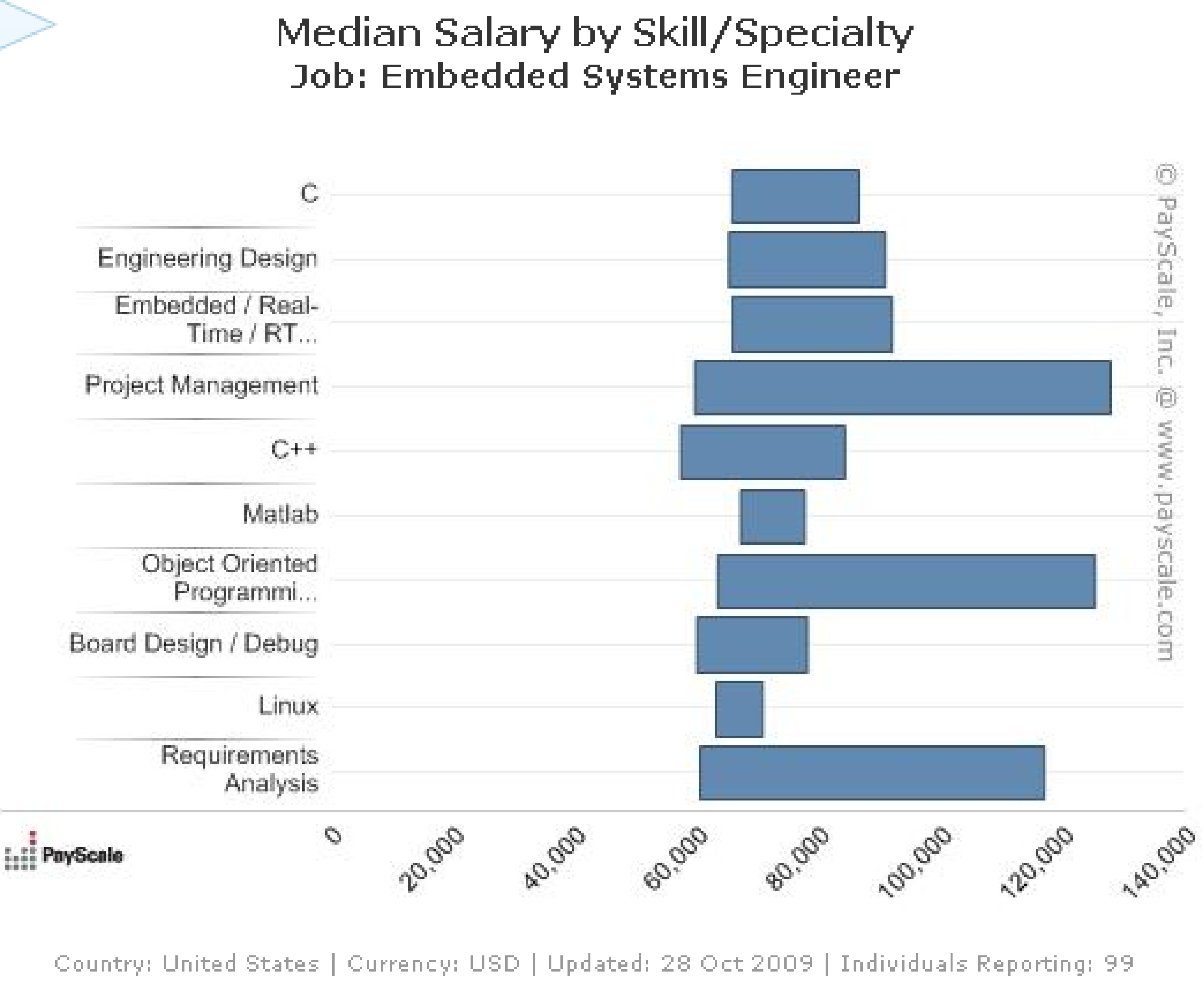
[payscale.com]

Software-Only Skills Pays Less Than SW+HW



[payscale.com]

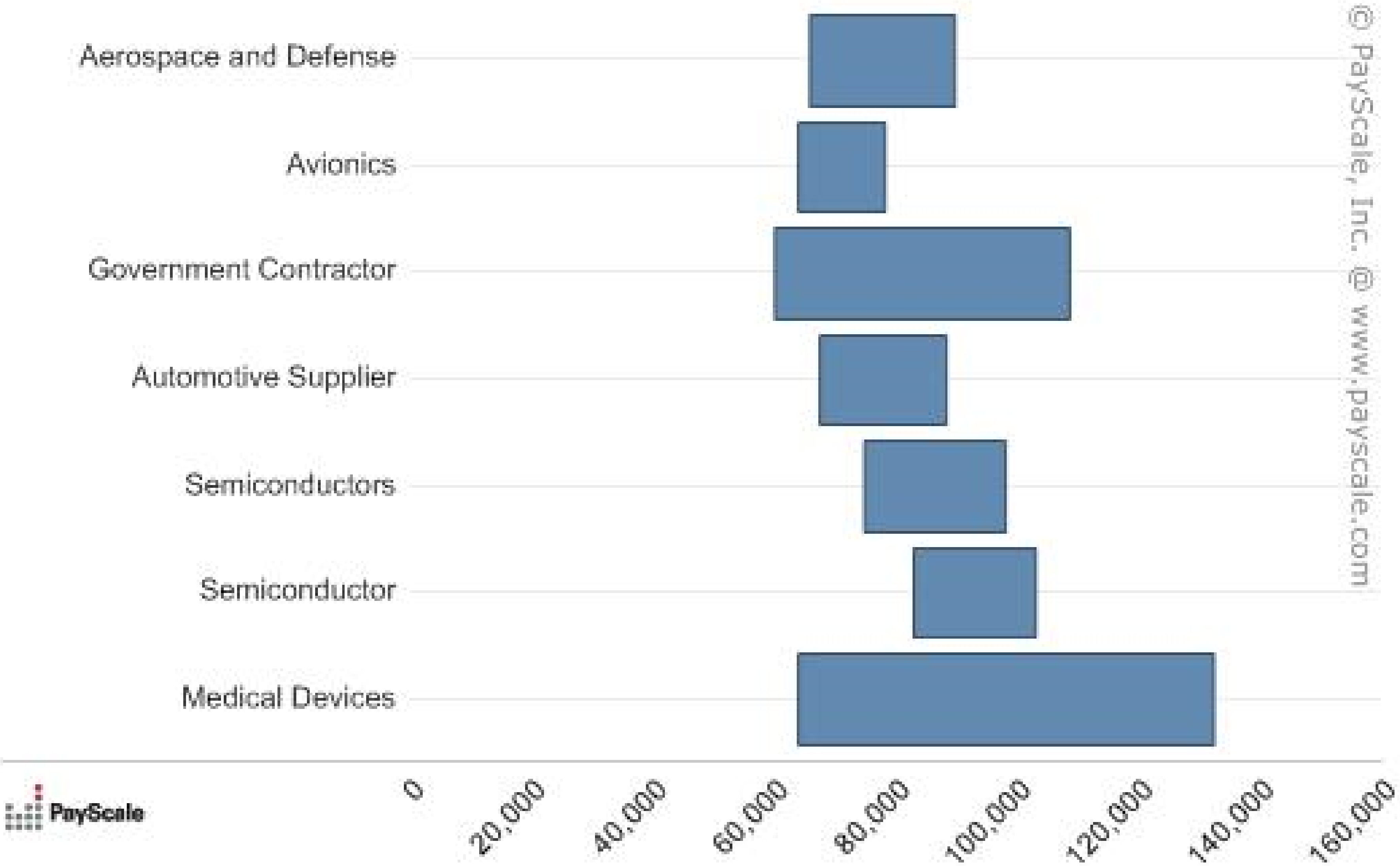
Requirements & Design Worth More Than Coding



[payscale.com]

Median Salary by Industry

Job: Embedded Systems Engineer

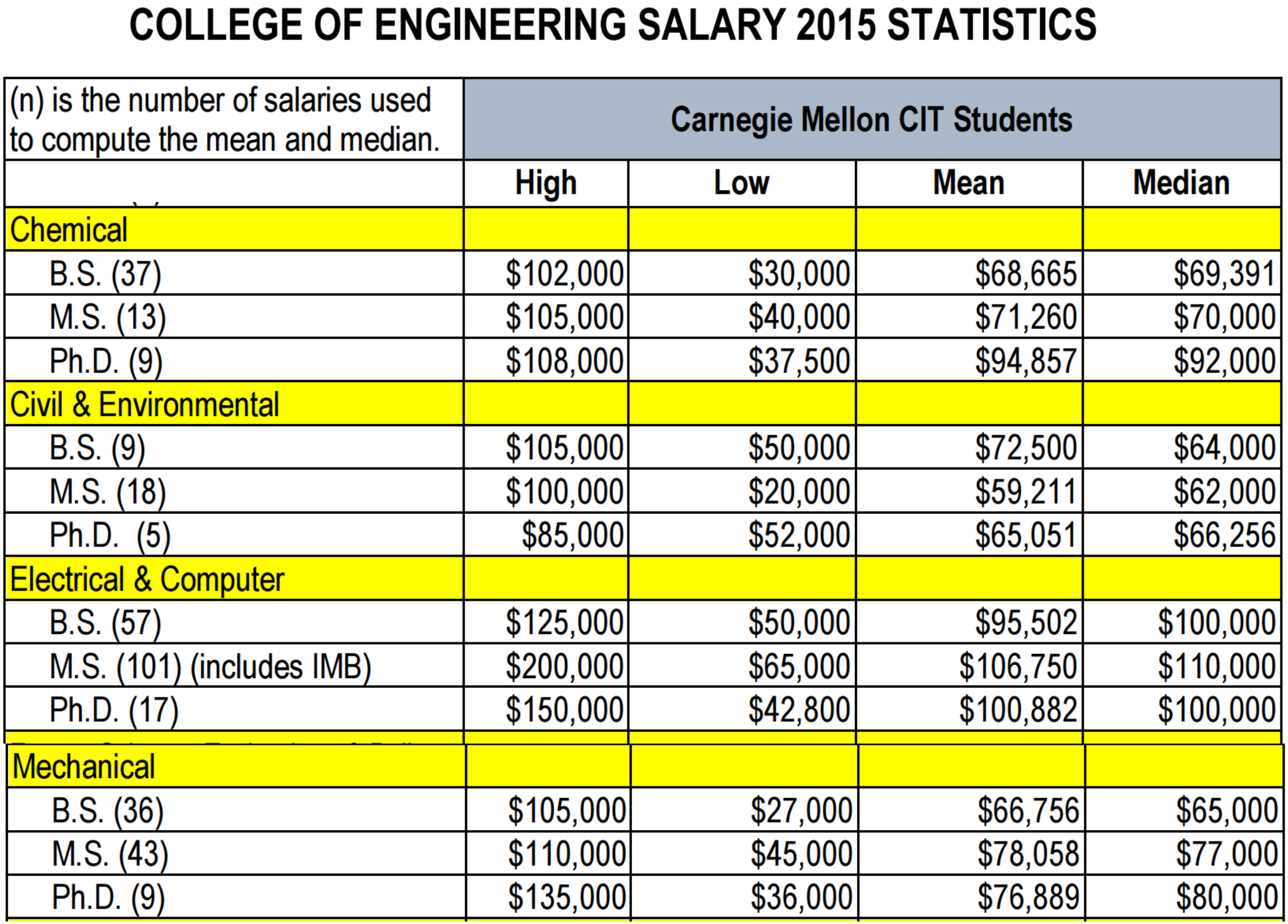


[payscale.com]

Country: United States | Currency: USD | Updated: 28 Oct 2009 | Individuals Reporting: 230

CIT Salary Data (Excerpts)

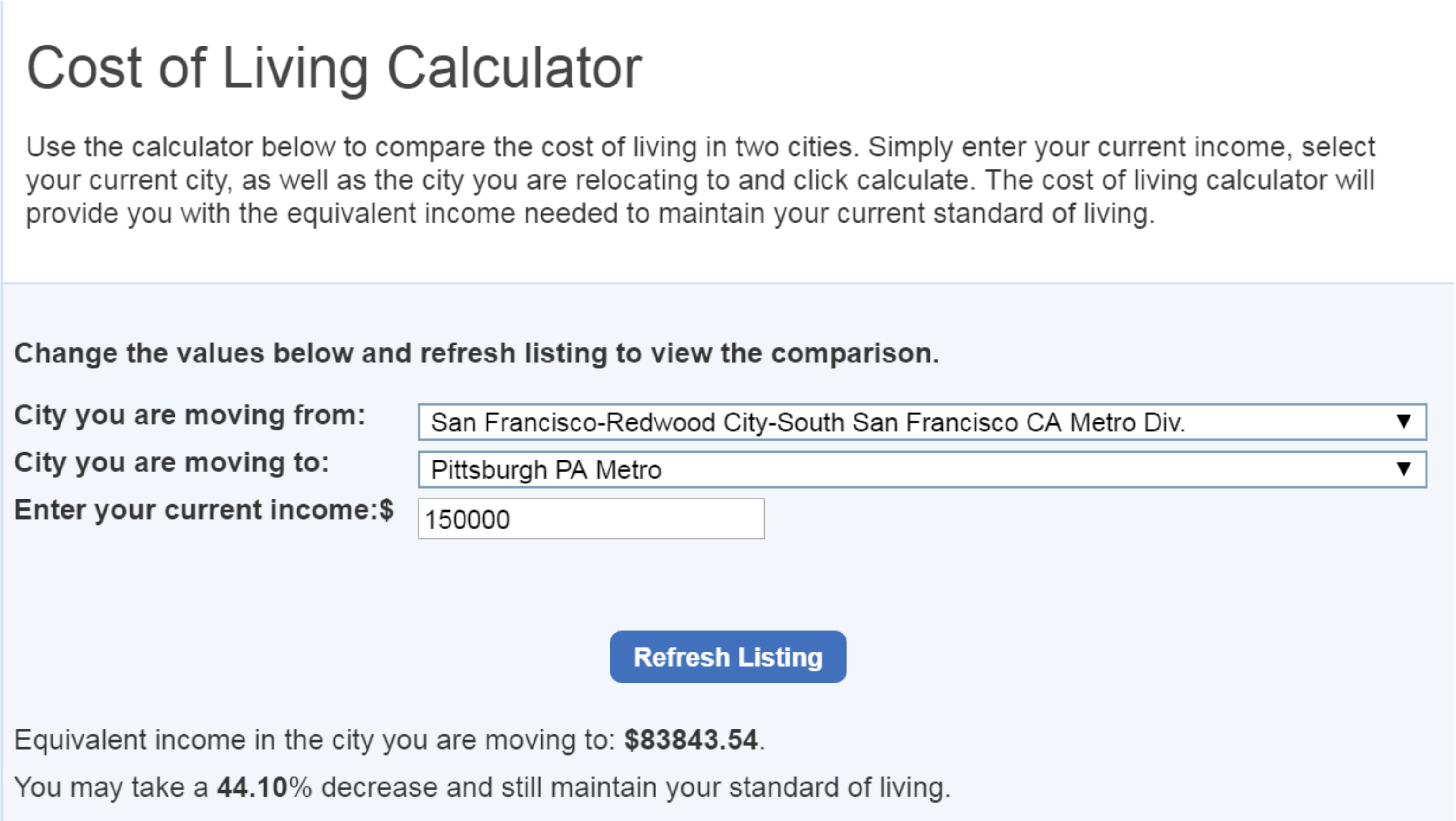
- https://www.cmu.edu/career/salaries-and-destinations/2015-survey/pdfs-one-pagers/All%20Engineering%20Salaries%202015.pdf



COLLEGE OF ENGINEERING SALARY 2015 STATISTICS

(n) is the number of salaries used to compute the mean and median.	Carnegie Mellon CIT Students			
	High	Low	Mean	Median
Chemical				
B.S. (37)	$102,000	$30,000	$68,665	$69,391
M.S. (13)	$105,000	$40,000	$71,260	$70,000
Ph.D. (9)	$108,000	$37,500	$94,857	$92,000
Civil & Environmental				
B.S. (9)	$105,000	$50,000	$72,500	$64,000
M.S. (18)	$100,000	$20,000	$59,211	$62,000
Ph.D. (5)	$85,000	$52,000	$65,051	$66,256
Electrical & Computer				
B.S. (57)	$125,000	$50,000	$95,502	$100,000
M.S. (101) (includes IMB)	$200,000	$65,000	$106,750	$110,000
Ph.D. (17)	$150,000	$42,800	$100,882	$100,000
Mechanical				
B.S. (36)	$105,000	$27,000	$66,756	$65,000
M.S. (43)	$110,000	$45,000	$78,058	$77,000
Ph.D. (9)	$135,000	$36,000	$76,889	$80,000

Don’t Forget Relative Cost of Living



Cost of Living Calculator

Use the calculator below to compare the cost of living in two cities. Simply enter your current income, select your current city, as well as the city you are relocating to and click calculate. The cost of living calculator will provide you with the equivalent income needed to maintain your current standard of living.

Change the values below and refresh listing to view the comparison.

City you are moving from: San Francisco-Redwood City-South San Francisco CA Metro Div. ▼

City you are moving to: Pittsburgh PA Metro ▼

Enter your current income:$ 150000

Equivalent income in the city you are moving to: **$83843.54**.

You may take a **44.10**% decrease and still maintain your standard of living.

- **Other factors:**
  - Opportunity density
  - Room-mate more common in high-cost area
  - …

Review

- **Analog to Digital Converters ADC; A/D**
  - Know general components and what they do
  - Understand operating parameters

- **Types of ADCs**
  - Know how an integrating ADC works
  - Know how a Flash ADC works
  - Know how a Successive Approximation ADC works

- **Engineering considerations**
  - Understand tradeoff between Flash and SAR approaches
    - Speed
    - Cost
  - Understand difference between resolution and accuracy
  - Be able to apply Nyquist Criterion

- **Understand tradeoffs of moving average filters**