COMP-462
Embedded Systems

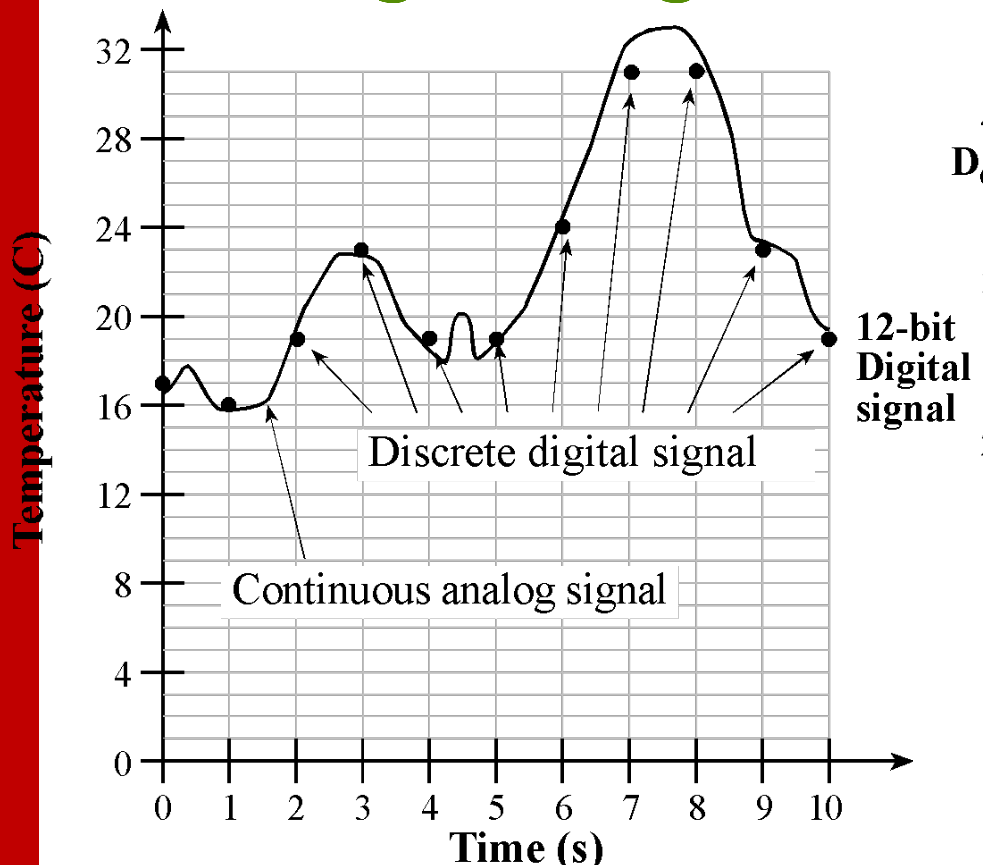
Lecture 11: Sampling, Analog-to-Digital Conversion

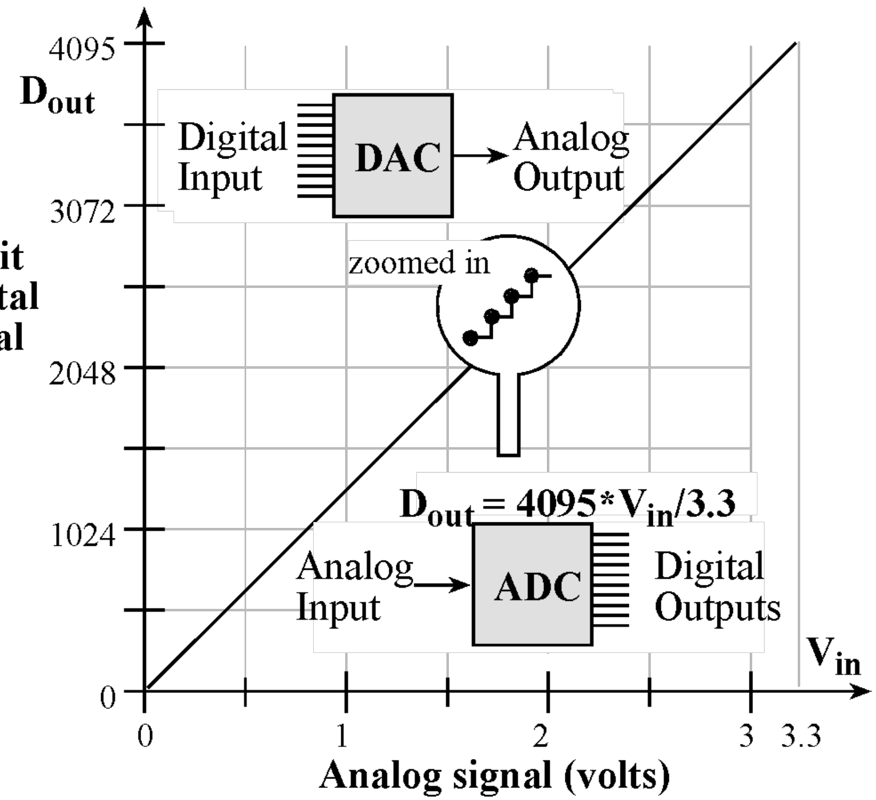
Agenda

- Recap
  - Local Variables
  - Stack frames
  - Recursion
  - Fixed-point numbers

- Outline
  - Sampling, Nyquist theorem
  - Analog to Digital Conversion

Analog to Digital Converter (ADC)





Four limitations of digital sampling

- Finite precision (4096 alternatives)
- Finite voltage range (0 to 3.3V)
- Discrete time sampling, f_s
- Finite number of samples, N

Voltage resolution = 3.3V/4096 =0.8 mV

Frequency range = 0 to ½ f_s

Frequency resolution = f_s / N

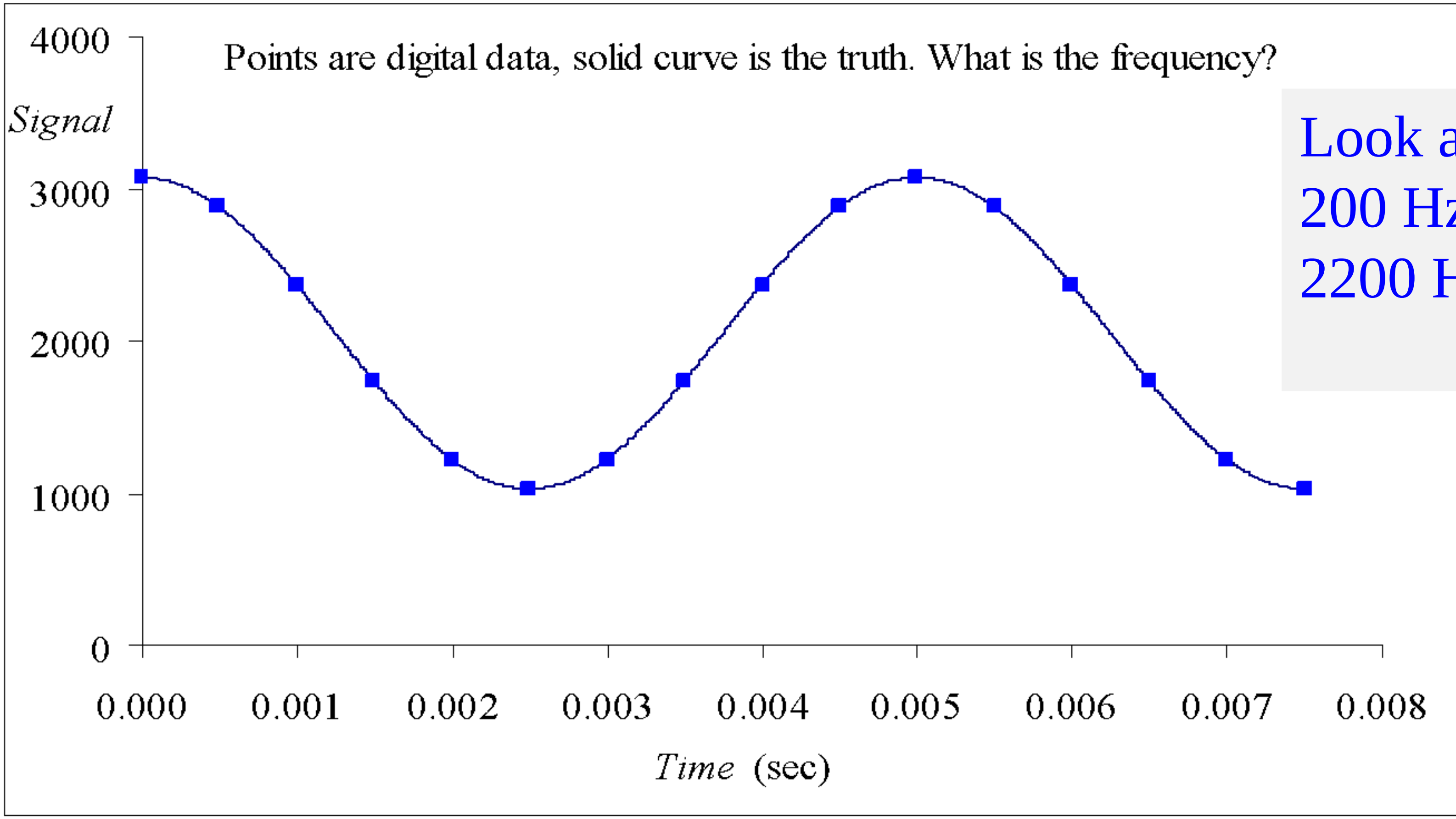
Nyquist Theorem

- A *bandlimited* analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate f_s exceeds $2f_{max}$ samples per second, where f_{max} is the highest frequency in the original signal.
  - If the analog signal does contain frequency components larger than $(1/2)f_s$, then there will be an **aliasing** error.
  - Aliasing is when the digital signal appears to have a different frequency than the original analog signal.

- **Valvano Postulate**: If f_{max} is the largest frequency component of the analog signal, then you must sample more than ten times f_{max} in order for the reconstructed digital samples to look like the original signal when plotted on a voltage versus time graph.

Sampling (option 1)

- 200Hz signal sampled at 2000Hz

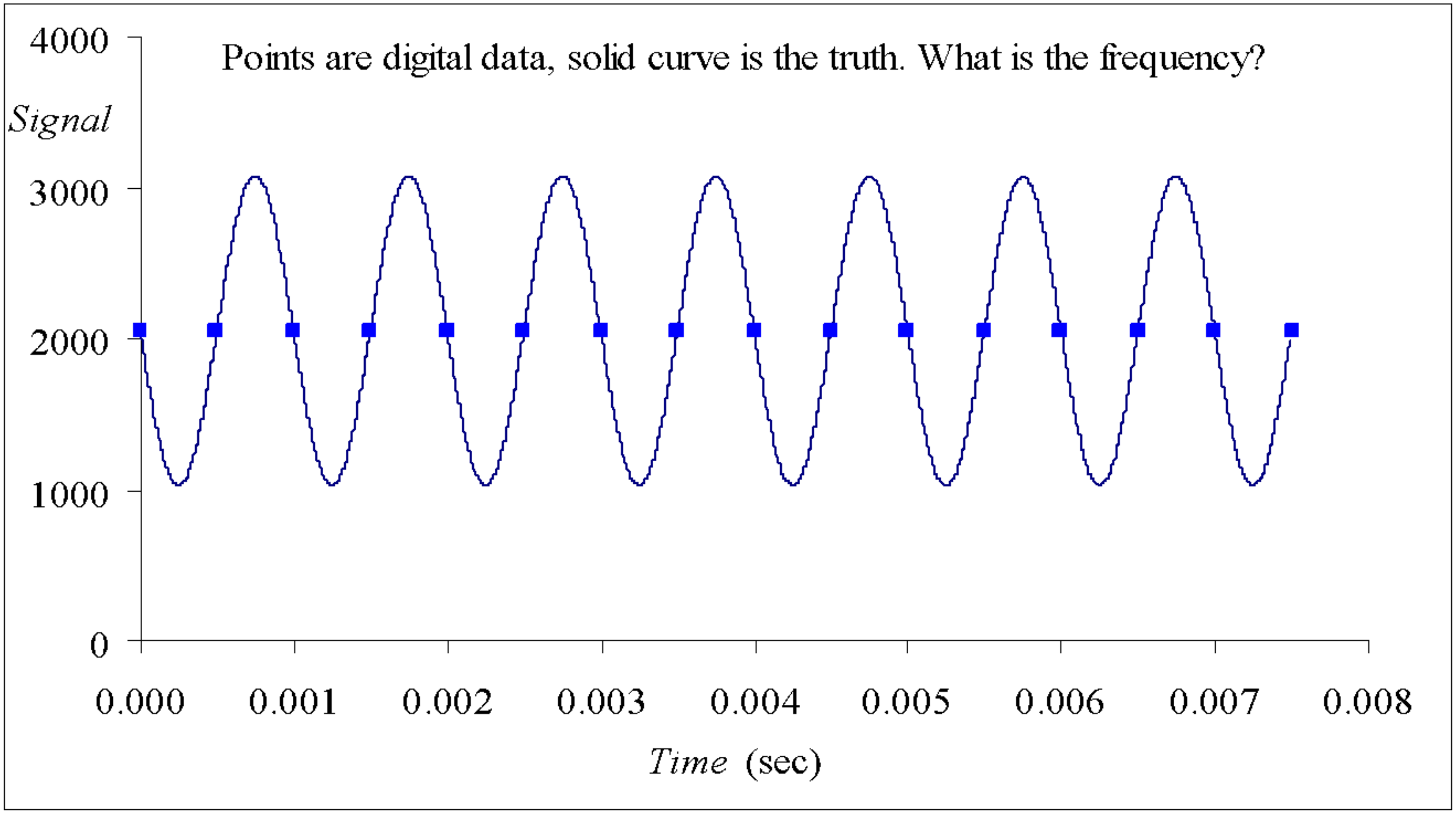


Look at
200 Hz
2200 Hz

http://www.ece.utexas.edu/~valvano/Volume1/Nyquist.xls

Sampling (option 1)

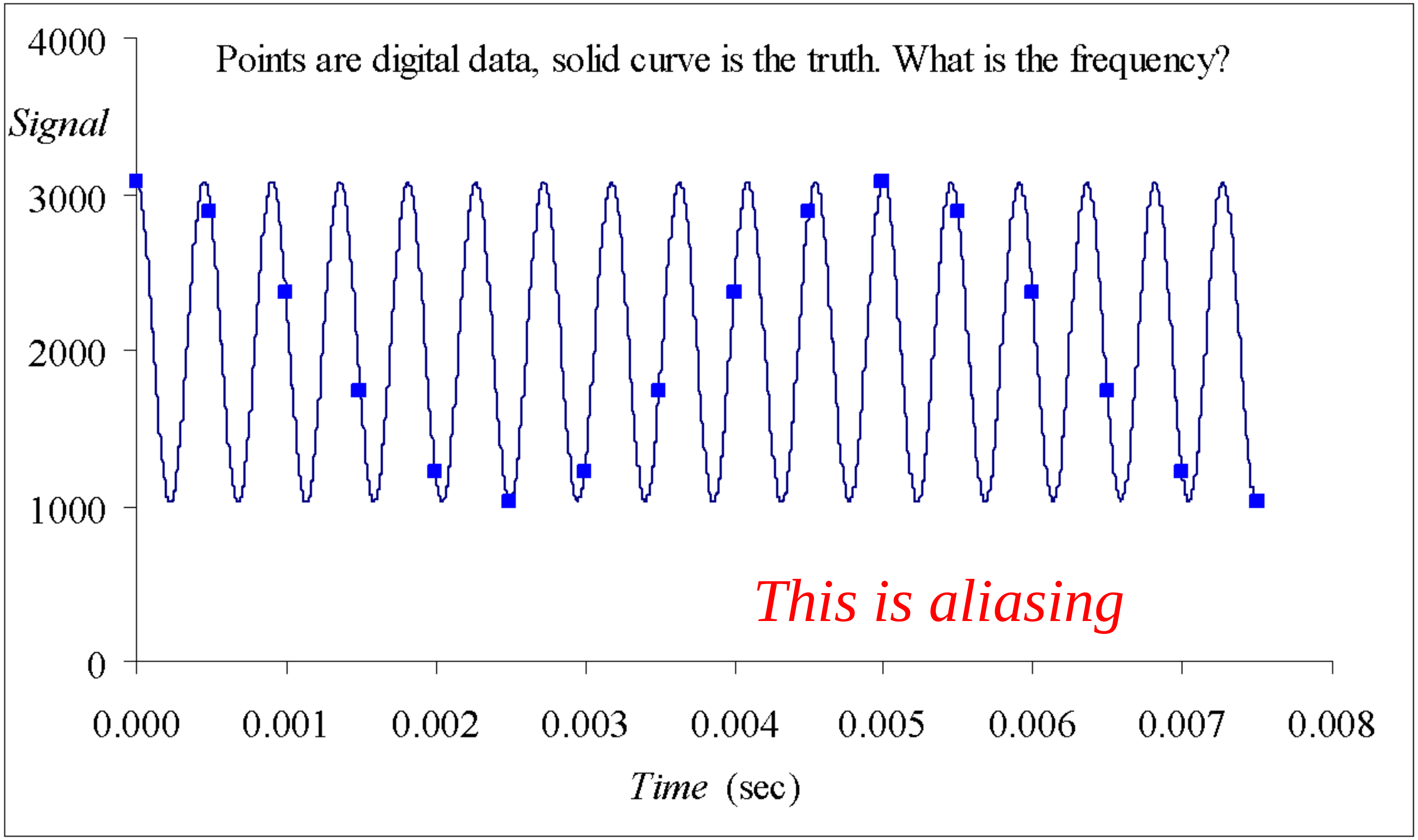
- 1000Hz signal sampled at 2000Hz



http://www.ece.utexas.edu/~valvano/Volume1/Nyquist.xls

Sampling (option 1)

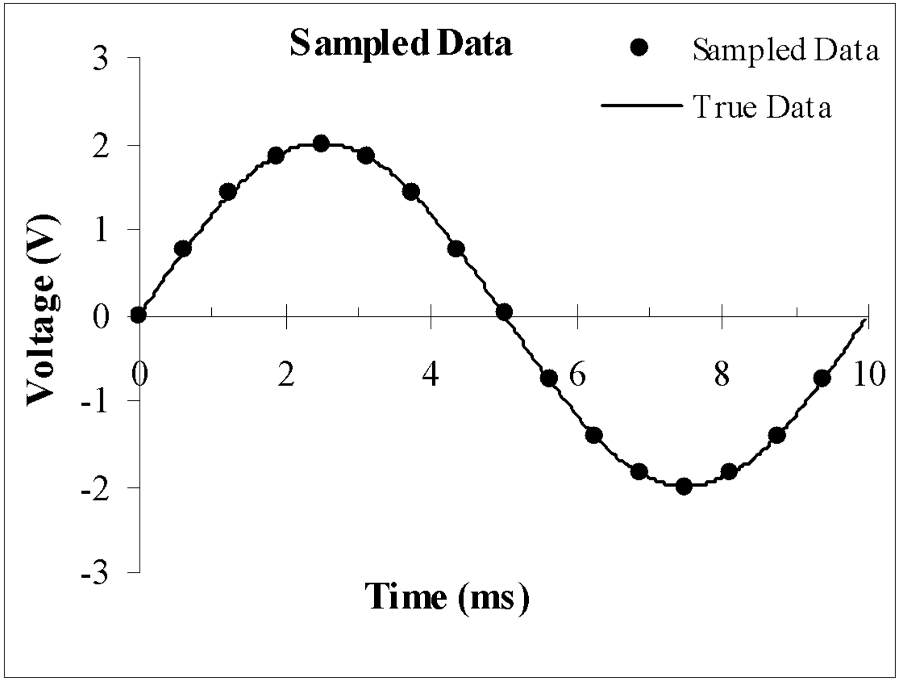
- 2200Hz signal sampled at 2000Hz

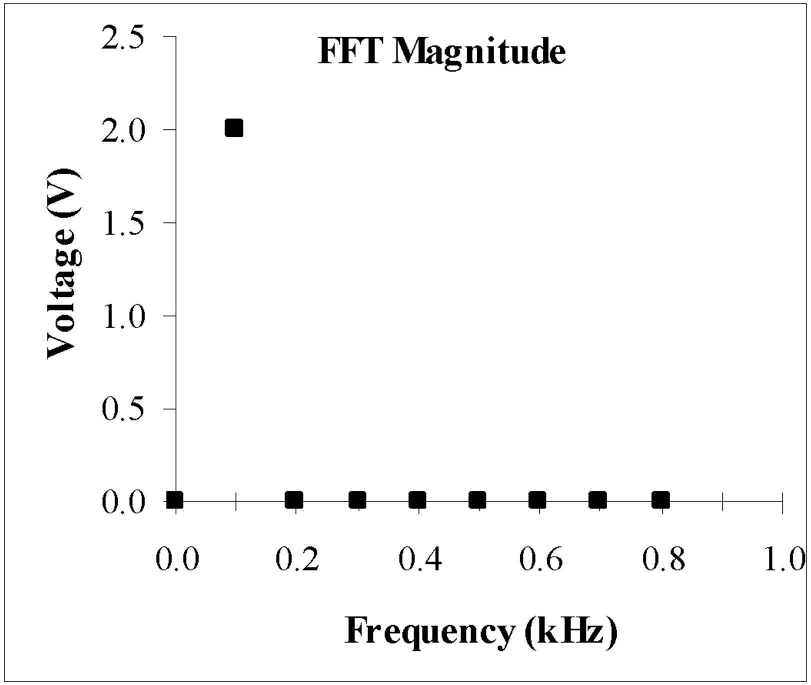


http://www.ece.utexas.edu/~valvano/Volume1/Nyquist.xls

Sampling (option 2)

- 100Hz signal sampled at 1600Hz

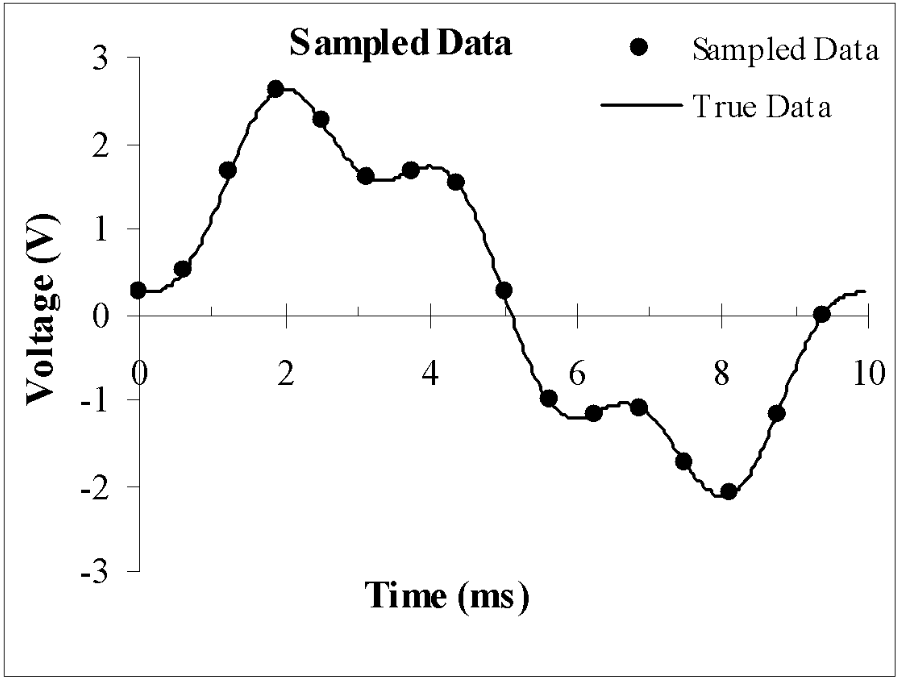


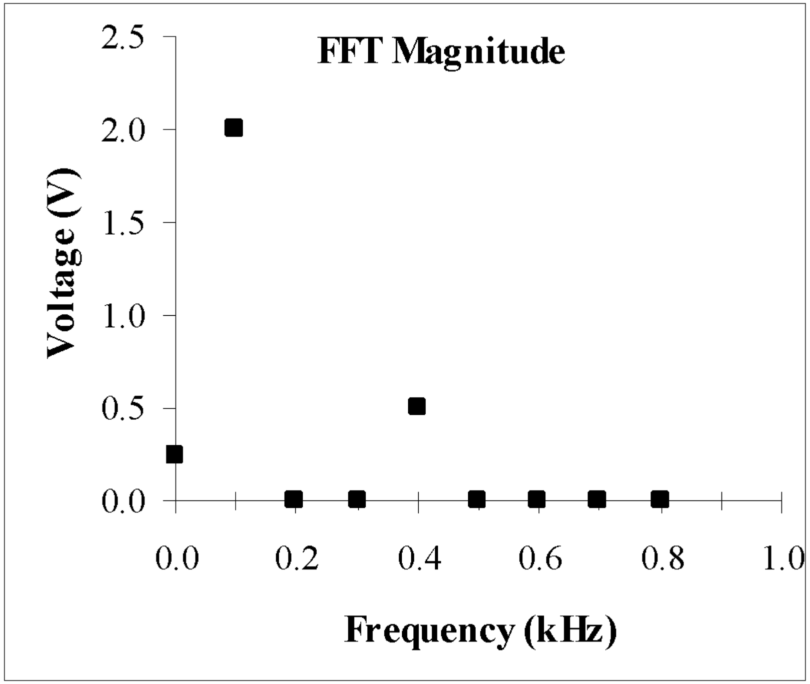


http://www.ece.utexas.edu/~valvano/EE345L/Labs/Fall2011/FFT16.xls

Sampling (option 2)

- A signal with DC, 100Hz and 400Hz sampled at 1600Hz



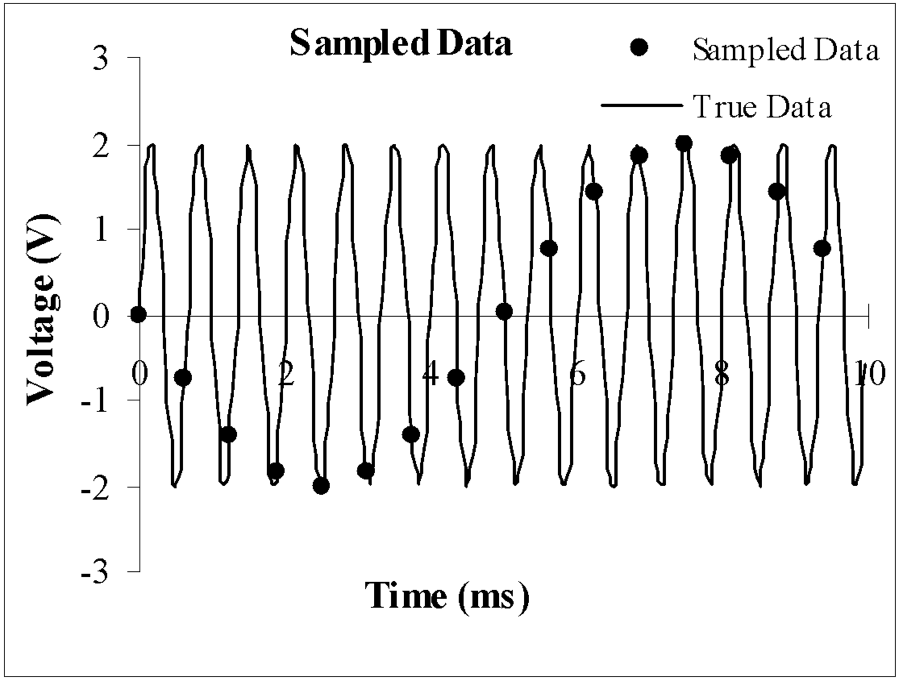


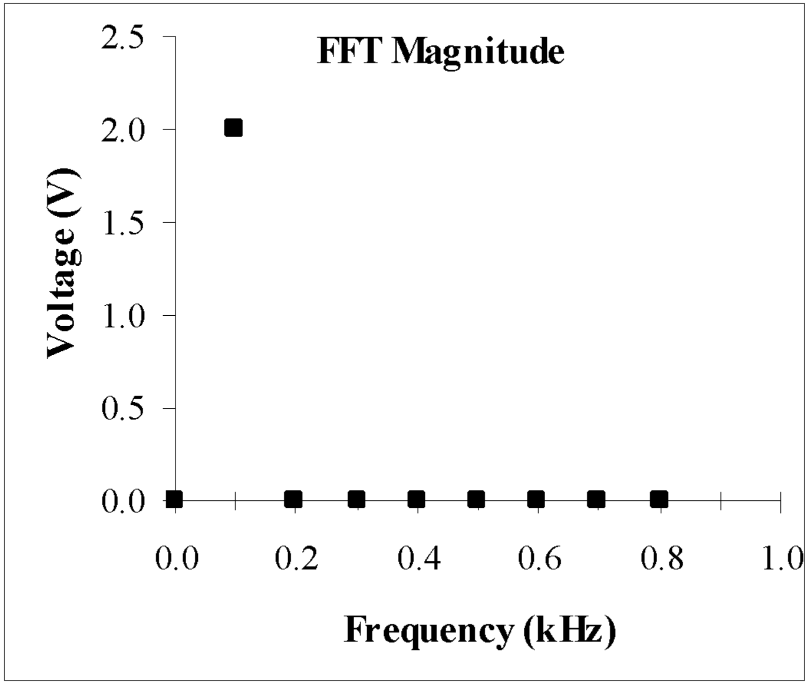
http://www.ece.utexas.edu/~valvano/EE345L/Labs/Fall2011/FFT16.xls

Sampling (option 2)

- 1500Hz signal sampled at 1600Hz

This is aliasing



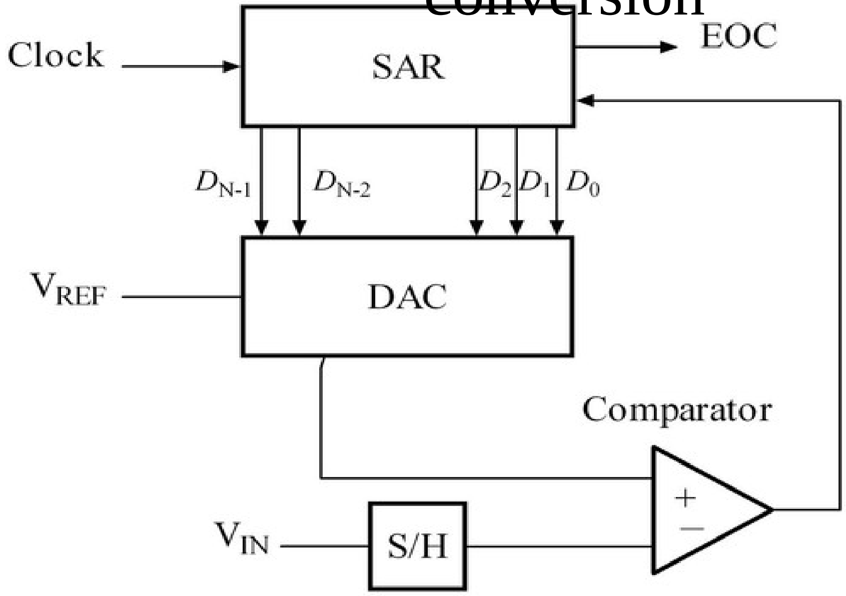


http://www.ece.utexas.edu/~valvano/EE345L/Labs/Fall2011/FFT16.xls

Analog to Digital Converter (ADC)

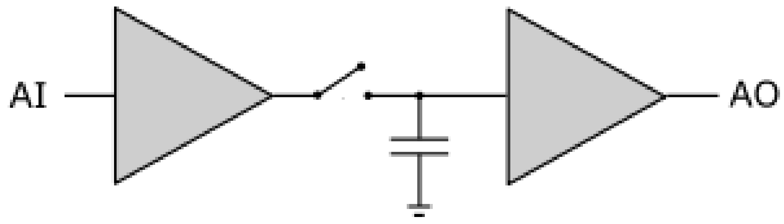
- Successive approximation ADC
  - V_{IN} is approximated as a static value in a *sample and hold* (S/H) circuit
  - the *successive approximation register* (SAR) is a counter that increments each *clock* as long as it is enabled by the *comparator*
  - output of the SAR is fed to a DAC that generates a voltage to compare with V_{IN}
  - when the output of the DAC = V_{IN} the value of SAR is the digital representation of V_{IN}

end of conversion



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)**

ADC on TM4C123

- Sampling Range/Resolution
  - 3.3V internal reference voltage
  - 0x000 at 0 V input
  - 0xFFF at 3.3 V
  - resolution = range/precision

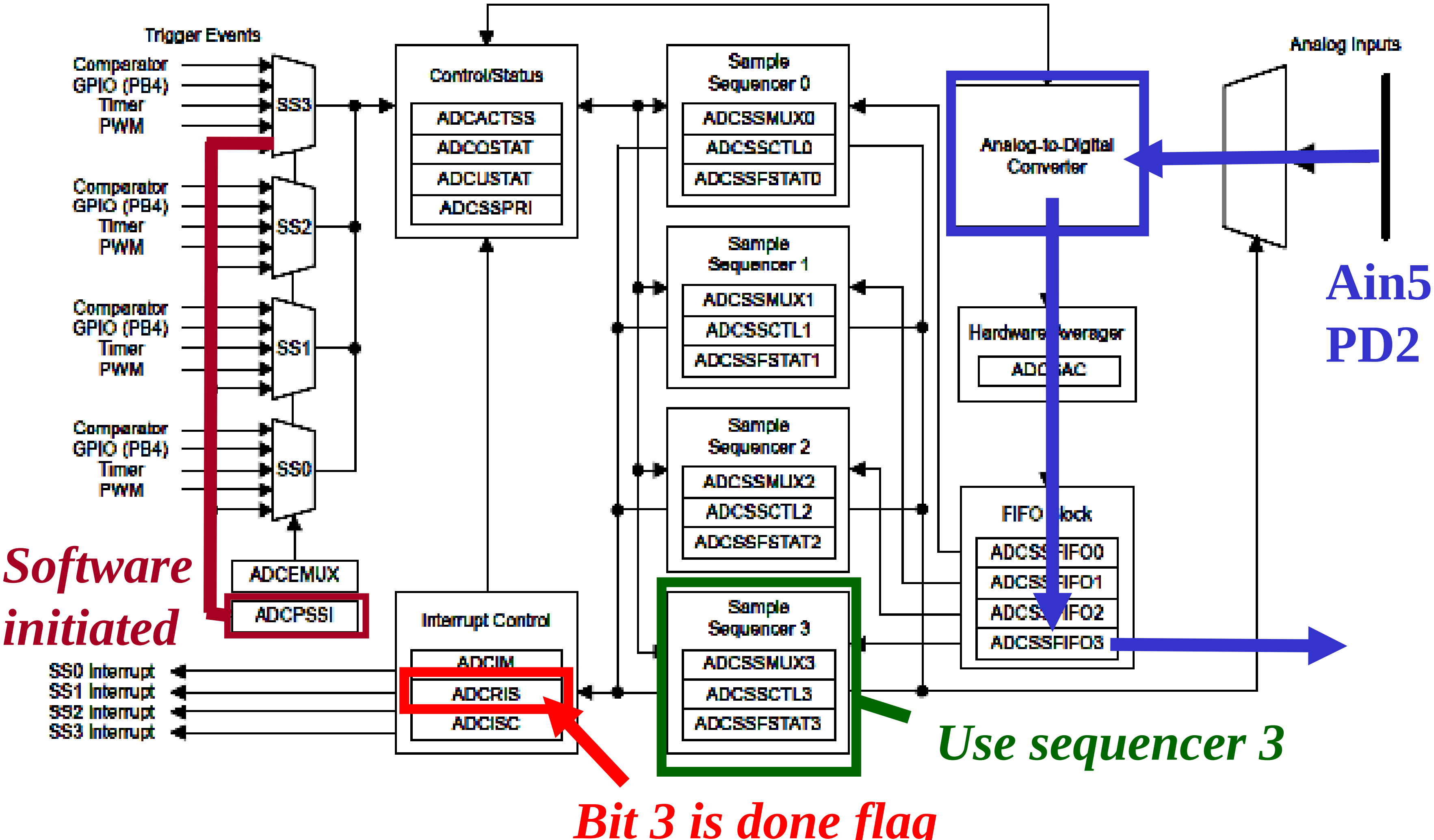
    = 3.3V/**4096** alternatives < 1mV
  - Actual resolution dominated by noise
- Improve signal to noise ratio (SNR)
  - Slow down ADC (take longer to sample)
  - Analog filtering, ground shield
  - Digital filtering (average multiple samples)

ADC on TM4C123

- Twelve analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate up to one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators (trigger if analog input crosses a threshold)
  - Pulse Width Modulator (another timer)
  - GPIO (input pin)
  - Continuous
- Hardware averaging of up to 64 samples for improved accuracy
- Converter uses an internal 3.3V reference

We will use software initiated trigger

ADC on TM4C123



ADC on TM4C123

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I_2C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I_2C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I_2C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I_2C2SDA	M0PWM5	M1PWM3			CAN0Tx		

PE4=Ain9 used in book and ADCSWTrigger_4C123

Twelve different pins can be used to sample analog inputs.

ADC on TM4C123

- TM4C ADC registers

Address	31-17	16	15-10	9	8	7-0	Name
0x400F.E100		ADC		MAXADCSPD			SYSCTL_RCGC0_R

	31-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0	
0x4003.8020		SS3		SS2		SS1		SS0	ADC0_SSPRI_R

	31-16	15-12	11-8	7-4	3-0	
0x4003.8014		EM3	EM2	EM1	EM0	ADC0_EMUX_R

	31-4	3	2	1	0	
0x4003.8000		ASEN3	ASEN2	ASEN1	ASEN0	ADC0_ACTSS_R
0x4003.80A0		MUX0				ADC0_SSMUX3_R
0x4003.80A4		TS0	IE0	END0	D0	ADC0_SSCTL3_R
0x4003.8028		SS3	SS2	SS1	SS0	ADC0_PSSI_R
0x4003.8004		INR3	INR2	INR1	INR0	ADC0_RIS_R
0x4003.800C		IN3	IN2	IN1	IN0	ADC0_ISC_R

	31-12	11-0	
0x4003.80A8		DATA	ADC0_SSFIFO3

ADC on TM4C123

- TM4C123 ADC Operation
  - select rate
  - select sequencer
  - select trigger
  - select channel
  - select sample mode
    - 0 not temperature
    - 1 set completion flag
    - 1 end sequence
    - 0 not differential

Value	*Description*
0x3	1M samples/second
0x2	500K samples/second
0x1	250K samples/second
0x0	125K samples/second

Speed bits in ADC0_PC_R

Value	*Event*
0x0	Software start
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Analog Comparator 2
0x4	External (GPIO PB4)
0x5	Timer
0x6	PWM0
0x7	PWM1
0x8	PWM2
0xF	Always (continuously sample)

EM3, EM2, EM1, and EM0 bits in ADC_EMUX_R

ADC0_SSCTL3_R = 0x06;

ADC on TM4C123

- Initialization (11 steps)
  - **Step 1.** We enable the port clock for the pin that we will be using for the ADC input.
  - **Step 2.** Make that pin an input by writing zero to the **DIR** register.
  - **Step 3.** Enable the alternative function on that pin by writing one to the **AFSEL** register.
  - **Step 4.** Disable the digital function on that pin by writing zero to the **DEN** register.
  - **Step 5.** Enable the analog function on that pin by writing one to the **AMSEL** register.
  - **Step 6.** We enable the ADC clock by setting bit 16 of the **SYSCTL_RCGC0_R** register
  - Step 7. Determine the rate using Bits 8 and 9 of the **SYSCTL_RCGC0_R** register
  - **Step 8.** We will set the priority of each of the four sequencers (ADC0_SSPRI_R).

ADC on TM4C123

- Initialization (11 steps)

- **Step 9. D**isable sequencer 3, write a 0 to bit 3 (**ASEN3**) in the **ADC_ACTSS_R** re
- Step 10. We configure the trigger event for the sample sequencer in the **ADC_EMUX_R** register (0000 to bits 15–12 (**EM3**))
- Step 11. Configure the corresponding input source in the **ADCSSMUXn** register (3–0 in the **ADC_SSMUX3_R**)
- Step 12. Configure the sample control bits in the corresponding nibble in the **ADC0SSCTLn** register (0110)

  - Bit 3 is the **TS0** bit =0, not measuring temperature.
  - Bit 2 is the **IE0** bit =1, we want to the **RIS** bit to be set when the sample is complete.
  - Bit 1 is the **END0** bit, the last (and only) sample in the sequence.
  - Bit 0 is the **D0** bit, not using differential mode.

- **Step 13.** We enable the sample sequencer logic, write a 1 to bit 3 (**ASEN3**) in the **ADC_ACTSS_R** register.

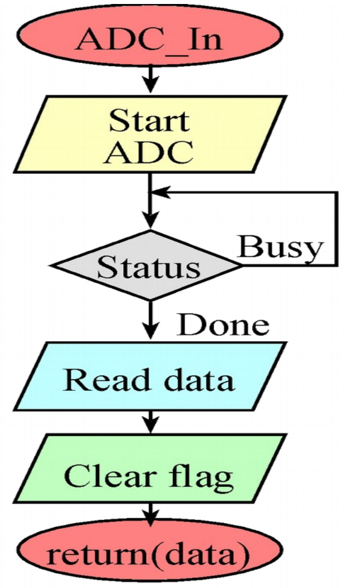
ADC on TM4C123

```
void ADC0_InitSWTriggerSeq3_Ch9(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;   // 1) activate clock for Port E
  delay = SYSCTL_RCGC2_R;         //    allow time for clock to stabilize
  GPIO_PORTE_DIR_R &= ~0x04;      // 2) make PE4 input
  GPIO_PORTE_AFSEL_R |= 0x04;     // 3) enable alternate function on PE2
  GPIO_PORTE_DEN_R &= ~0x04;      // 4) disable digital I/O on PE2
  GPIO_PORTE_AMSEL_R |= 0x04;     // 5) enable analog function on PE2
  SYSCTL_RCGC0_R |= 0x00010000;   // 6) activate ADC0
  delay = SYSCTL_RCGC2_R;
  SYSCTL_RCGC0_R &= ~0x00000300;  // 7) configure for 125K
  ADC0_SSPRI_R = 0x0123;          // 8) Sequencer 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger
  ADC0_SSMUX3_R &= ~0x000F;       // 11) clear SS3 field
  ADC0_SSMUX3_R += 9;             //    set channel Ain9 (PE4)
  ADC0_SSCTL3_R = 0x0006;         // 12) no TS0 D0, yes IE0 END0
  ADC0_ACTSS_R |= 0x0008;         // 13) enable sample sequencer 3
}
```

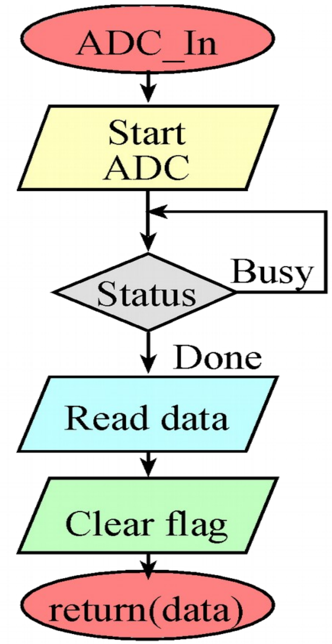
ADC on TM4C123

Analog to digital conversion

- ❑ **Step 1.** The ADC is started using the software trigger. The channel to sample was specified earlier in the initialization.
- ❑ **Step 2.** The function waits for the ADC to complete by polling the RIS register bit 3.
- ❑ **Step 3.** The 12-bit digital sample is read out of sequencer 3.
- ❑ **Step 4.** The RIS bit is cleared by writing to the ISC register.



ADC on TM4C123




```
//------------ADC_InSeq3------------
// Busy-wait analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
unsigned long ADC0_InSeq3(void){  unsigned long result;
  ADC0_PSSI_R = 0x0008;            // 1) initiate SS3
  while((ADC0_RIS_R&0x08)==0){};   // 2) wait for conversion done
  result = ADC0_SSFIFO3_R&0xFFF;   // 3) read result
  ADC0_ISC_R = 0x0008;             // 4) acknowledge completion
  return result;
}
```

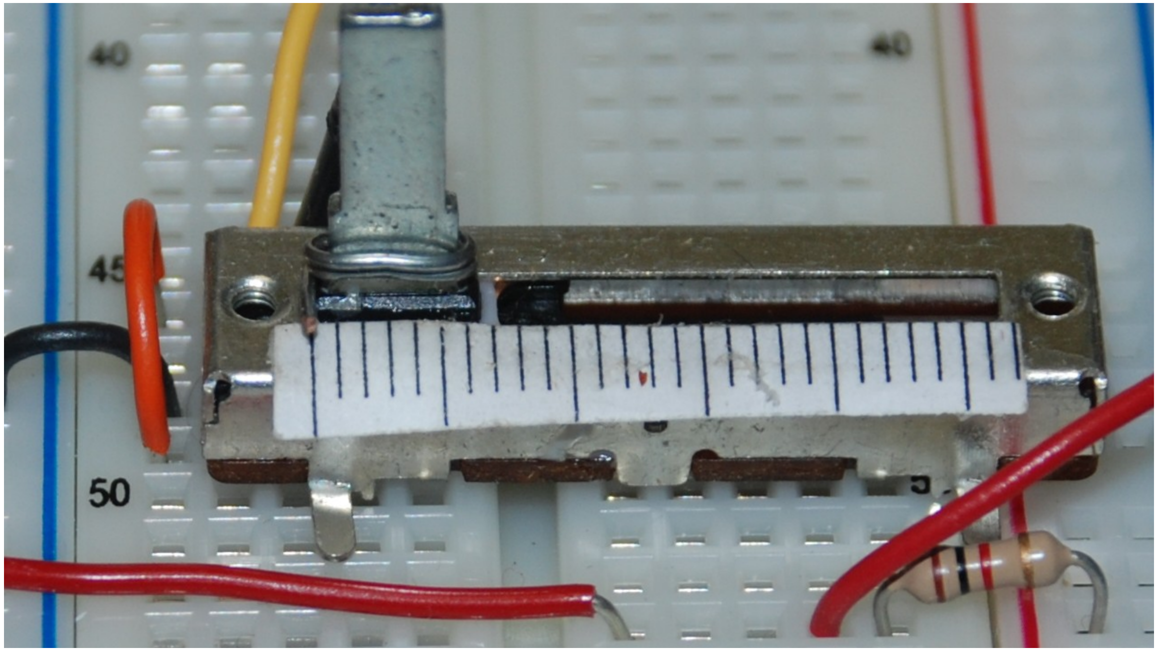
Data Acquisition System

- **Hardware**
  - **Transducer**
  - **Electronics**
  - **ADC**

- **Software**
- **ADC device driver**
- **Timer routines**
  - **Output compare interrupts**
- **LCD driver**
- **Measurement system**
  - **How fast to update**
  - **Fixed-point number system**
  - **Algorithm to convert ADC into position**

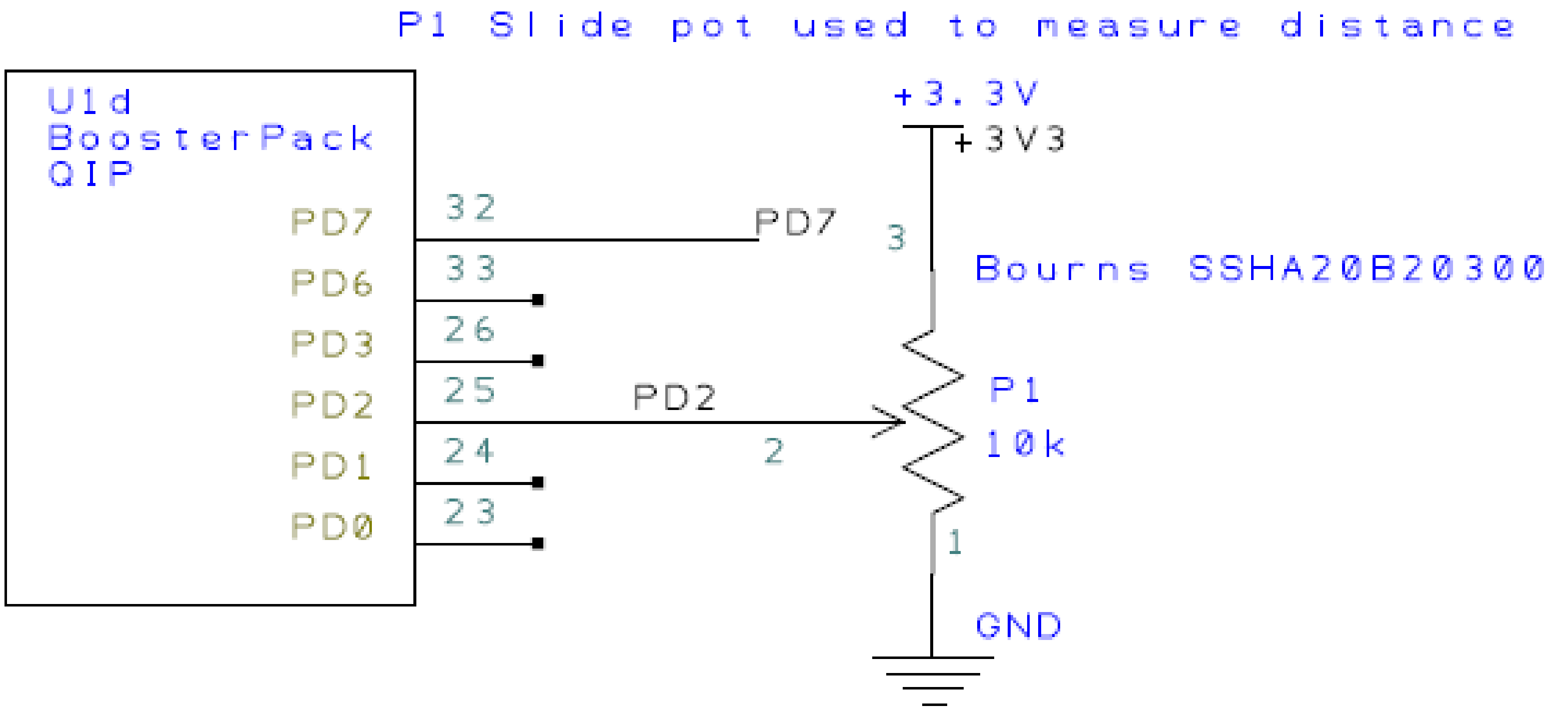
Analog Input Device

- Transducer – *A device actuated by power from one system that supplies power in the same or other form to another system.*



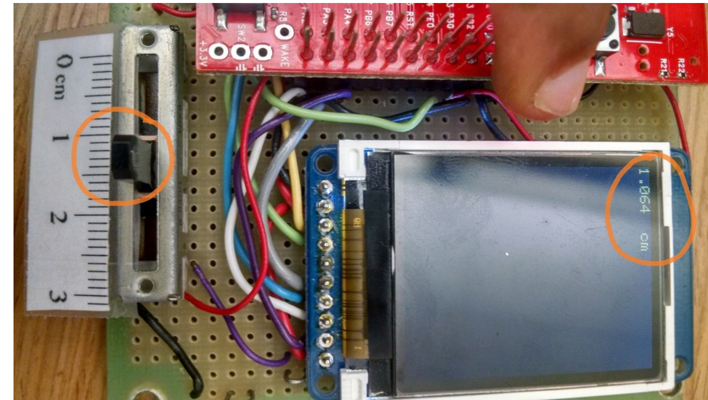
Transducer Circuit

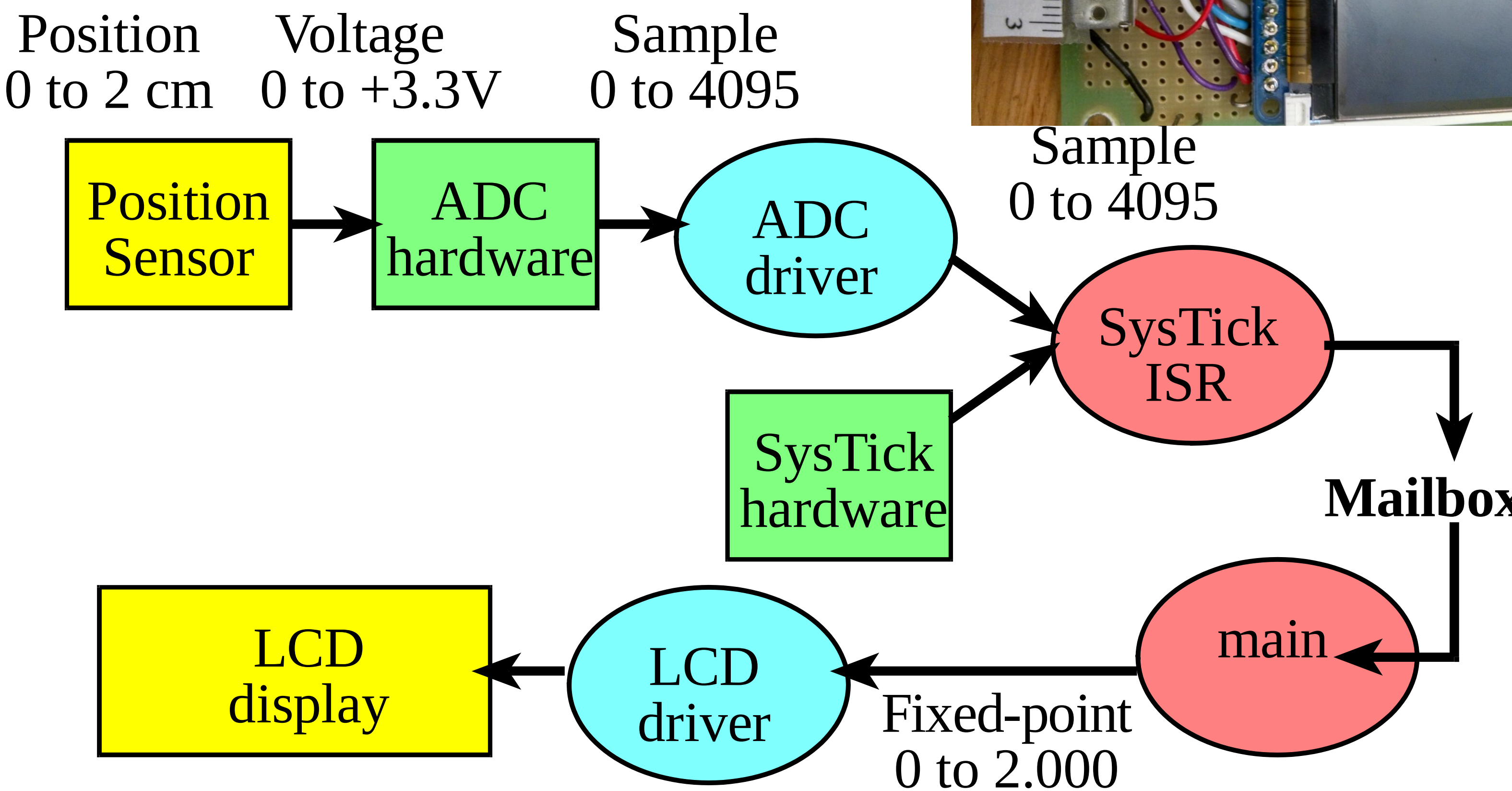
- Position to voltage



Data Acquisition System

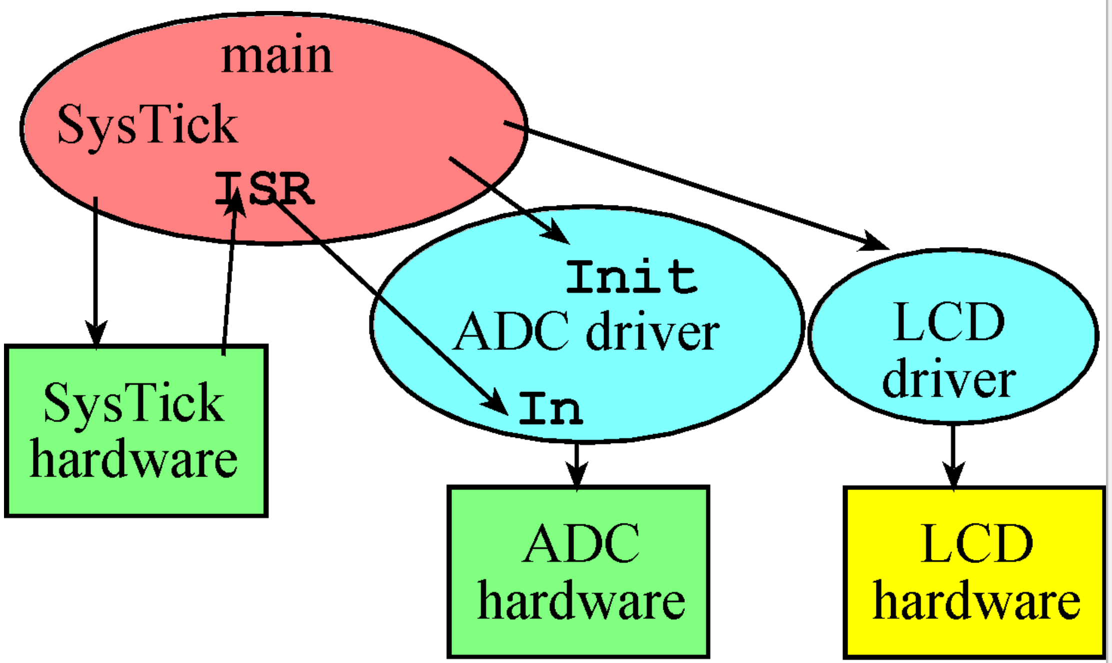
- ❑ Data flow graph

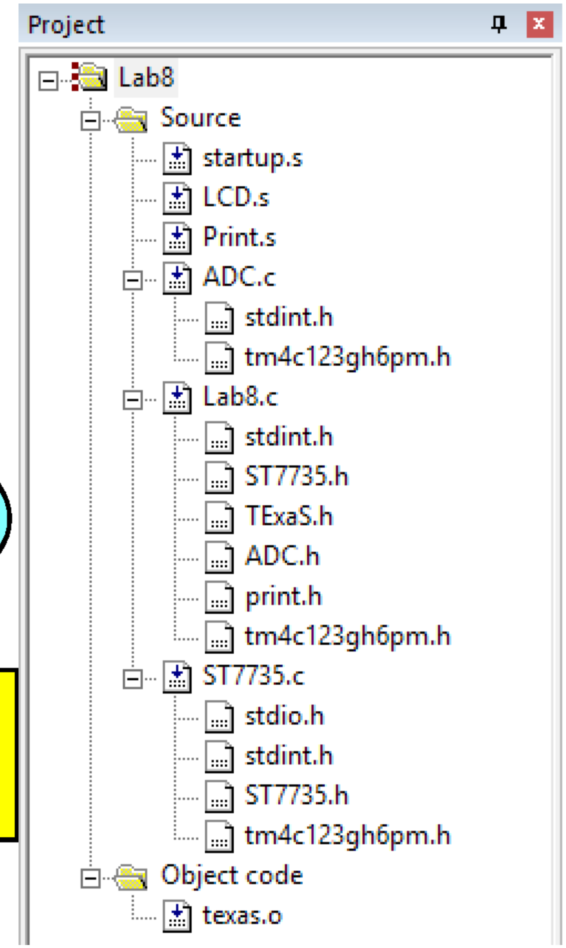




Data Acquisition System

- ❑Call graph





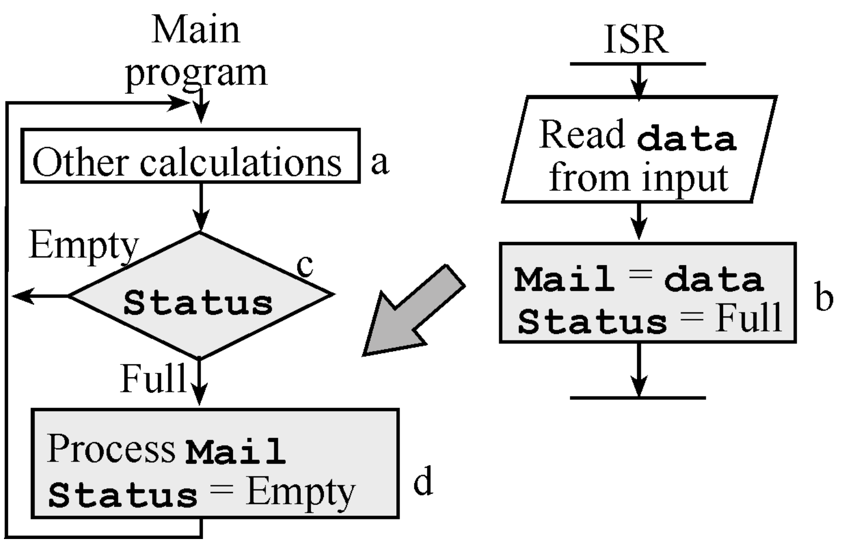
Thread Synchronization

Background thread

- SysTick ISR
  - Sample ADC
  - Store in ADCmail
  - Set ADCstatus

Foreground thread

- Main loop
  - Wait for ADCstatus
  - Read ADCmail
  - Clear ADCstatus
  - Convert to distance
  - Display on LCD

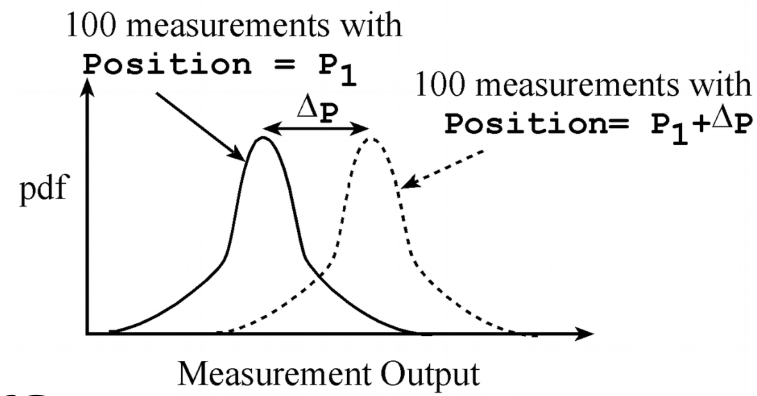


Sampling Jitter

- Definition of time-jitter, $\delta \mathbf{t}$:
  - Let $\mathbf{n}\Delta\mathbf{t}$ be the time a task is scheduled to be run and $\mathbf{t_n}$ the time the task is actually run
  - Then $\delta \mathbf{t_n} = \mathbf{t_n} - \mathbf{n}\Delta\mathbf{t}$
- Real time systems with periodic tasks, must have an upper bound, $\mathbf{k}$, on the time-jitter
  - $-\mathbf{k} \leq \delta \mathbf{t_n} \leq +\mathbf{k}$ for all $\mathbf{n}$

Measurement

- ❑ Resolution: Limiting factors
  - ❖ Transducer noise
  - ❖ Electrical noise
  - ❖ ADC precision
  - ❖ Software errors



- ❑ Accuracy: Limiting factors
  - ❖ Resolution
  - ❖ Calibration
  - ❖ Transducer stability

$$\text{Average accuracy (with units of x)} = \frac{1}{n}\sum_{i=0}^{n}\left|x_{ti} - x_{mi}\right|$$

Fixed-Point Revisited

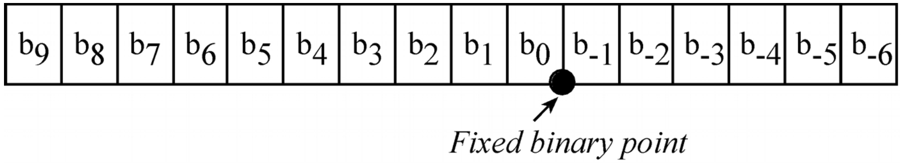
- **Why:**
  express non-integer values
  no floating point hardware support (want it to run fast)
- **When:**
  range of values is known
  range of values is small
- **How:**
  1) **variable integer**, called **I**.
     may be signed or unsigned
     may be 8, 16 or 32 bits (**precision**)
  2) **fixed constant**, called **Δ (resolution)**
     value is fixed, and can not be changed
     not stored in memory
     specify this fixed content using comments

$\text{value} \equiv \text{integer} \bullet \boldsymbol{\Delta}$

Fixed-Point Numbers

- The value of the fixed-point number:
  Fixed-point number ≡ **I•Δ**
  Smallest value = $I_{min} \cdot \Delta$, where I_{min} is the smallest integer
  Largest value = $I_{max} \cdot \Delta$, where I_{max} is the largest integer

- **Decimal fixed-point, Δ=**10^m
  Decimal fixed-point number = **I** • 10^m
  Nice for human input/output



- **Binary fixed-point, Δ=**2^m
  Binary fixed-point number = **I** • 2^m
  Easier for computers to perform calculations

Fixed-Point Math Example

Consider the following calculation.

$$C = 2*\pi*R$$

The variables C, and R are integers

$2\pi \approx 6.283$

$$C = (6283*R)/1000$$

Fixed-Point Math Example

Calculate the volume of a cylinder

$$\mathbf{V = \pi * R^2 * L}$$

The variables are fixed-point

$R = I*2^{-4}$ cm $L = J*2^{-4}$ cm

$V = K*2^{-8}$ cm^3 $\pi \approx 100*2^{-5}$

$$\mathbf{K = (100*I*I*J)>>9}$$