ADC – Analogue to Digital Conversion

- 13 pins support ADC operations
- ANn
- AN2 and AN3 may be used to provide reference voltages V_{REF-} and V_{REF+} for ADC. Can’t be then used for ADC.

DIO – 1 bit (2^1 values) voltmeter

1 – 5 V

0 – 0 V

ADC – 10 bit (2^{10} values) voltmeter

1023 – 5 V

1022

•

•

•

1

0 – 0 V

- What will 3.125 V be?
- What voltage is 632?
- ADC is digital or grainy
- Voltages are continuous or smooth
- Conversions will always be imprecise
- Digital values are bins. If voltage falls in a bin, assign that number.

- How big is each bin?

$$\frac{5V - 0V}{2^{10}} = 0.00488V$$

- Which bin would have 2.272 V?

$$\frac{2.272\,V}{5V - 0V} \times 2^{10} = 465$$

- What voltages lie in bin 762?

$$0 = \left[0 \leftrightarrow 0.00488\ V\right]$$
$$1 = \left[0.00488\ V \leftrightarrow 0.00976\ V\right]$$
$$\vdots = \vdots$$
$$b = \left[b \times 0.00488\ V \leftrightarrow (b+1) \times 0.00488\ V\right]$$
$$\therefore 762 = \left[3.7201\,V \leftrightarrow 3.7256\,V\right]$$

- We ordinarily don't know V_{input}, so we should assign voltage to middle of bin and then precision is half the bin size

$$V = \left(b + \frac{1}{2} \right) \times \frac{(5V - 0V)}{2^n} \pm \frac{1}{2} \frac{(5V - 0V)}{2^n}$$

$$\therefore 762 \leftrightarrow V = 3.7231 \pm 0.0024V$$

- **Warning** Don't try to find *b* from known *V* using this formula, we find *b* by simply dividing *V* by bin size

$$b = V \div \frac{(5V - 0V)}{2^n}$$

$$b = \frac{2.272\,V}{5V - 0V} \times 2^{10} = 465.31 = 465$$

Check:

$$V = \left(465 + \frac{1}{2}\right) \times \frac{(5V - 0V)}{2^{10}} \pm \frac{1}{2}\,\frac{(5V - 0V)}{2^{10}}$$

$$= 2.2729 \pm 0.0024$$

Bin	Voltage (V)
178	
0b10 1010 1111	
0x1FA	
	1.7152
	4.6850
	-1.0037
	5.7214

We have used 0V to 5 V as the range of the ADC conversions. We actually have some control over the range.

Suppose we use 1.2 V to 4.0 V instead. What is the bin size? Convert the following.

1000	
22	
418	
	1.5121 V
	3.1780 V
	2.2222 V

Suppose we use 8-bit ADC instead on a range of 0 V to 5.0 V. What is the bin size? Convert the following.

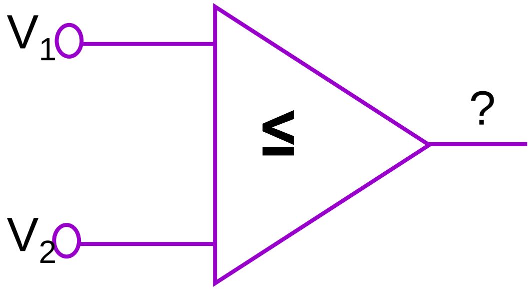
52	
112	
41	
	2.773 V
	4.245 V
	1.222 V

How ADC Works (Simplified)

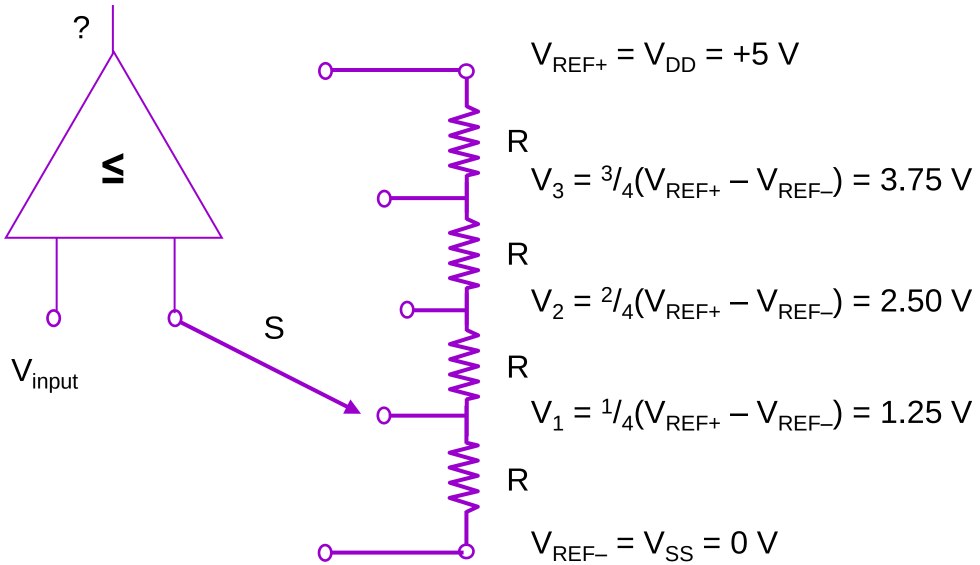
- Two elements
  - Comparator
  - Voltages to compare to (use a voltage divider)
- Algorithm (simple example is Method of Successive Approximation or the High-Low Game Strategy)

Comparator

- V_1 is input and V_2 is reference.
- Output is 0 ($V_1 < V_2$) or 1 ($V_1 \geq V_2$)

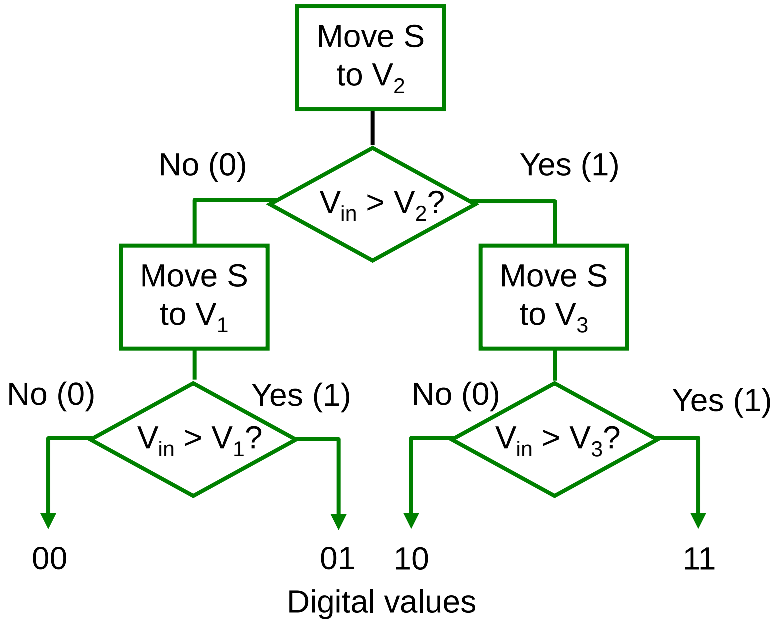


Voltage Divider to Compare to



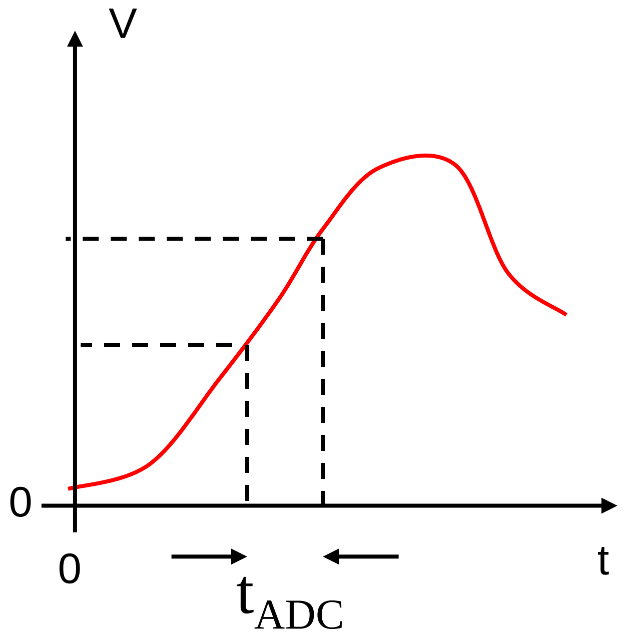
- 2 bit – 2^2 resistors; 2^n bit – 2^n resistors

Algorithm



Move S to V_2
No (0)
$V_{in} > V_2$?
Yes (1)
Move S to V_1
Move S to V_3
No (0)
$V_{in} > V_1$?
Yes (1)
No (0)
$V_{in} > V_3$?
Yes (1)
00
01
10
11
Digital values

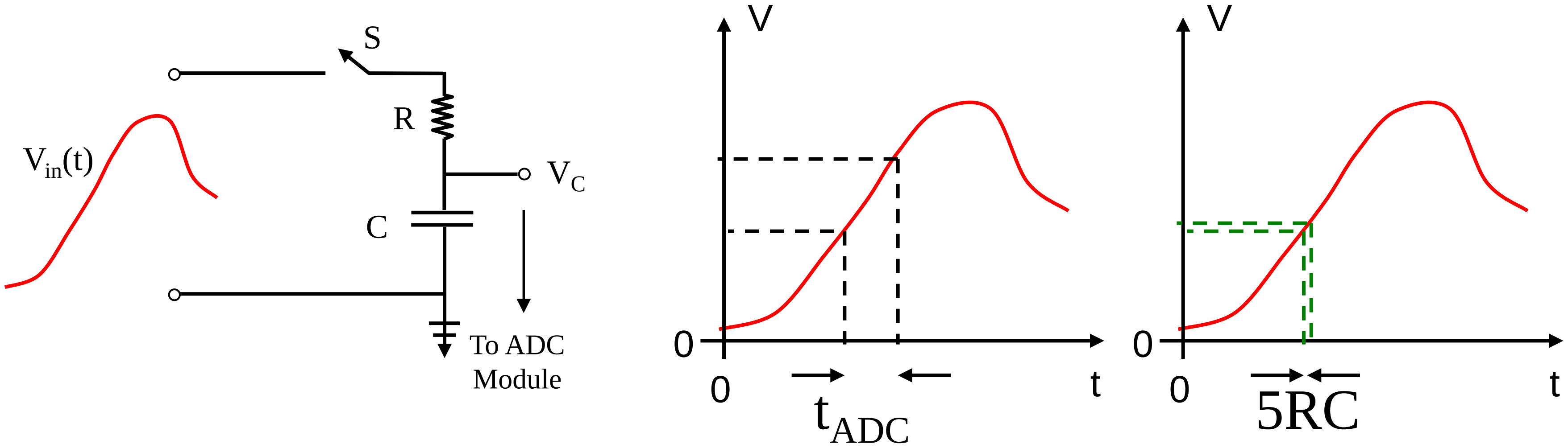
Acquisition time



- Varying voltage signals
- Cannot accurately do comparison

Sample and Hold Circuit

- Have an RC circuit with small value of RC (time constant).
- Will fully charge C in 5 to 10 RC.
- Want RC << t_{ADC}.
- Also want RC to be small that ΔV is small



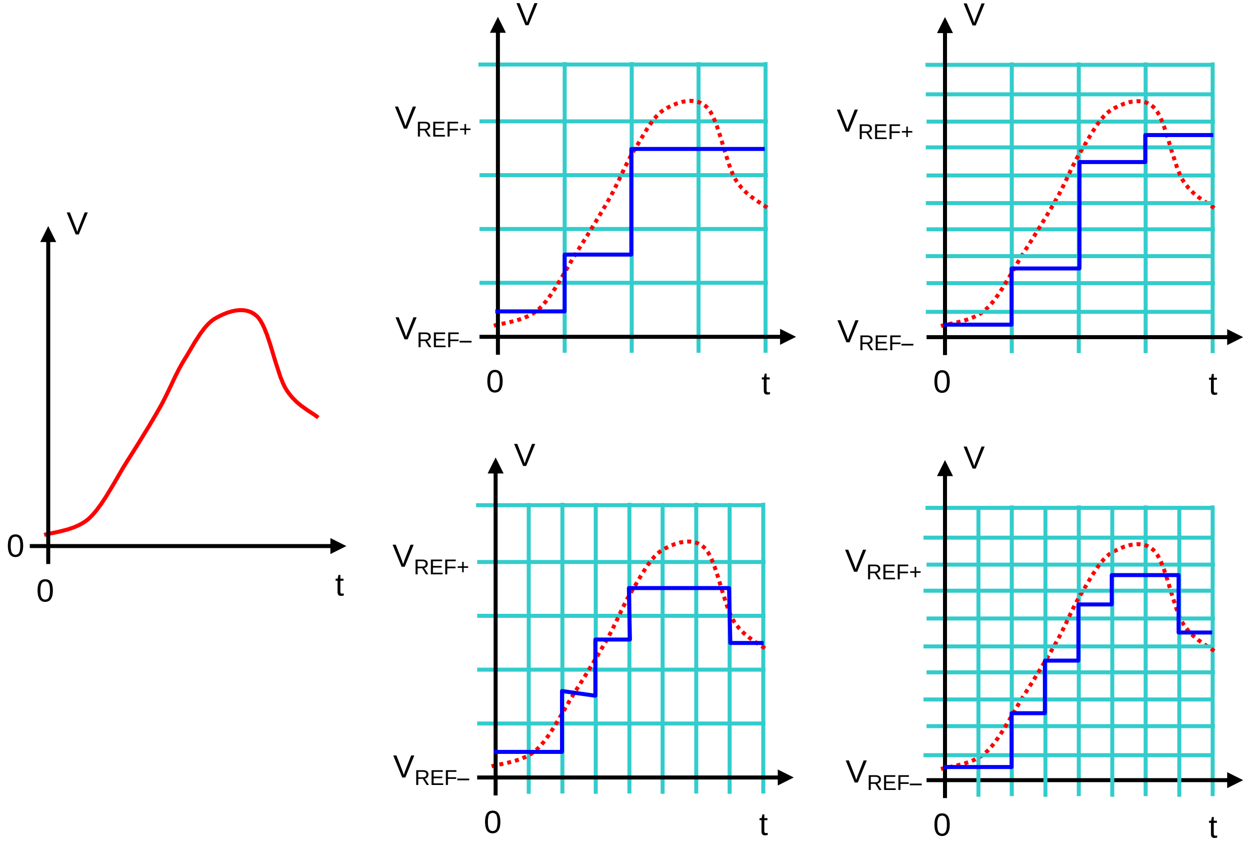
According to the datasheet on p. 227, C = 25 pF and R (actually the series sum of the R, the input resistance, and the switch resistance) is about 4 kΩ. What is τ? How long does it take to fully charge the capacitor (i.e. to more than 99%)?

$\tau = RC = 4000 \times 25E\text{-}12 = 1 \times 10^{-7}$ s = 0.1 μs

So

$t_{99} = 5RC = 0.5$ μs

- Higher precision (more bits) takes more comparisons and more time. Tradoff.



Software

```
#include <xc.h>
OpenADC(unsigned char config, unsigned char config2,
   unsigned char portconfig )
SetChanADC(unsigned char channel) // which input pin
                                  // ANn to use
ConvertADC()  // start conversion
BusyADC()     // = 1 if working, = 0 if finished
ReadADC()     // returns 10-bit ADC result
CloseADC()    // finish
```

channel in SetChanADC()

- Parameter can have value ADC_CHn where n is 0 to 12 and refers to pin ANn.
- Note that while many pins can be ADC inputs, you can only read voltage from one pin at a time.
- Cannot be run simultaneously.

portconfig in OpenADC()

- Value 0 to 15
- Configures sets of pins as Ann
- Only certain sets available

port config	ANn												
	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A	A	A	A	A	A	A	A	A	A	A	A	A
1	A	A	A	A	A	A	A	A	A	A	A	A	A
2	A	A	A	A	A	A	A	A	A	A	A	A	A
3	D	A	A	A	A	A	A	A	A	A	A	A	A
4	D	D	A	A	A	A	A	A	A	A	A	A	A
5	D	D	D	A	A	A	A	A	A	A	A	A	A
6	D	D	D	D	A	A	A	A	A	A	A	A	A
7	D	D	D	D	D	A	A	A	A	A	A	A	A
8	D	D	D	D	D	D	A	A	A	A	A	A	A
9	D	D	D	D	D	D	D	A	A	A	A	A	A
10	D	D	D	D	D	D	D	D	A	A	A	A	A
11	D	D	D	D	D	D	D	D	D	A	A	A	A
12	D	D	D	D	D	D	D	D	D	D	A	A	A
13	D	D	D	D	D	D	D	D	D	D	D	A	A
14	D	D	D	D	D	D	D	D	D	D	D	D	A
15	D	D	D	D	D	D	D	D	D	D	D	D	D
D – Digital A – Analogue Input (ADC)													

config2 in OpenADC()

3 choices

- ADC_INT_ON or ADC_INT_OFF √
- ADC_CHn, n = 0 to 12, default ADC channel/pin
- ADC_VREFPLUS_VDD, √ (use V_{DD})
  ADC_VREFMINUS_VSS, √ (use V_{SS})
  ADC_VREFPLUS_EXT, (+ Ref is Pin 5/AN3)
  ADC_VREFMINUS_EXT (– Ref is Pin 4/AN2)

config in OpenADC()

3 choices

- ADC_LEFT_JUST or ADC_RIGHT_JUST √
- ADC_FOSC_n, n = 2, 4, 8, 16, or 32
  allow time for AD conversion to complete
  $T_{AD} = n \times T_{OSC}$
- ADC_n_TAD, n = 0, 2, 4, 6, 12, 16, or 20
  allow time for capacitor to fully charge
  $T_{CAP} = n \times T_{AD}$

Choosing ADC_FOSC_N

- Measured in units called T_{AD}
- Need 1 T_{AD} + 1 T_{AD}/bit = 11 T_{AD} per conversion.
- T_{ADmin} is fixed ~ 0.7 μs (from ref. sheet)
- We have limited choices for T_{AD}
- T_{AD} = ADC_FOSC_N * $T_{OSC} \geq 0.7$ μs
- Or T_{AD} = ADC_FOSC_N / f_{OSC}
- Or ADC_FOSC_N ≥ 0.7 fOSC (fOSC in MHz)

fosc (MHz)	Minimum ADC_FOSC_N	T_{AD}
32	32	1 µs
16	16	1 µs
8	8	1 µs
4	4	1 µs
≤ 2	2	1 µs

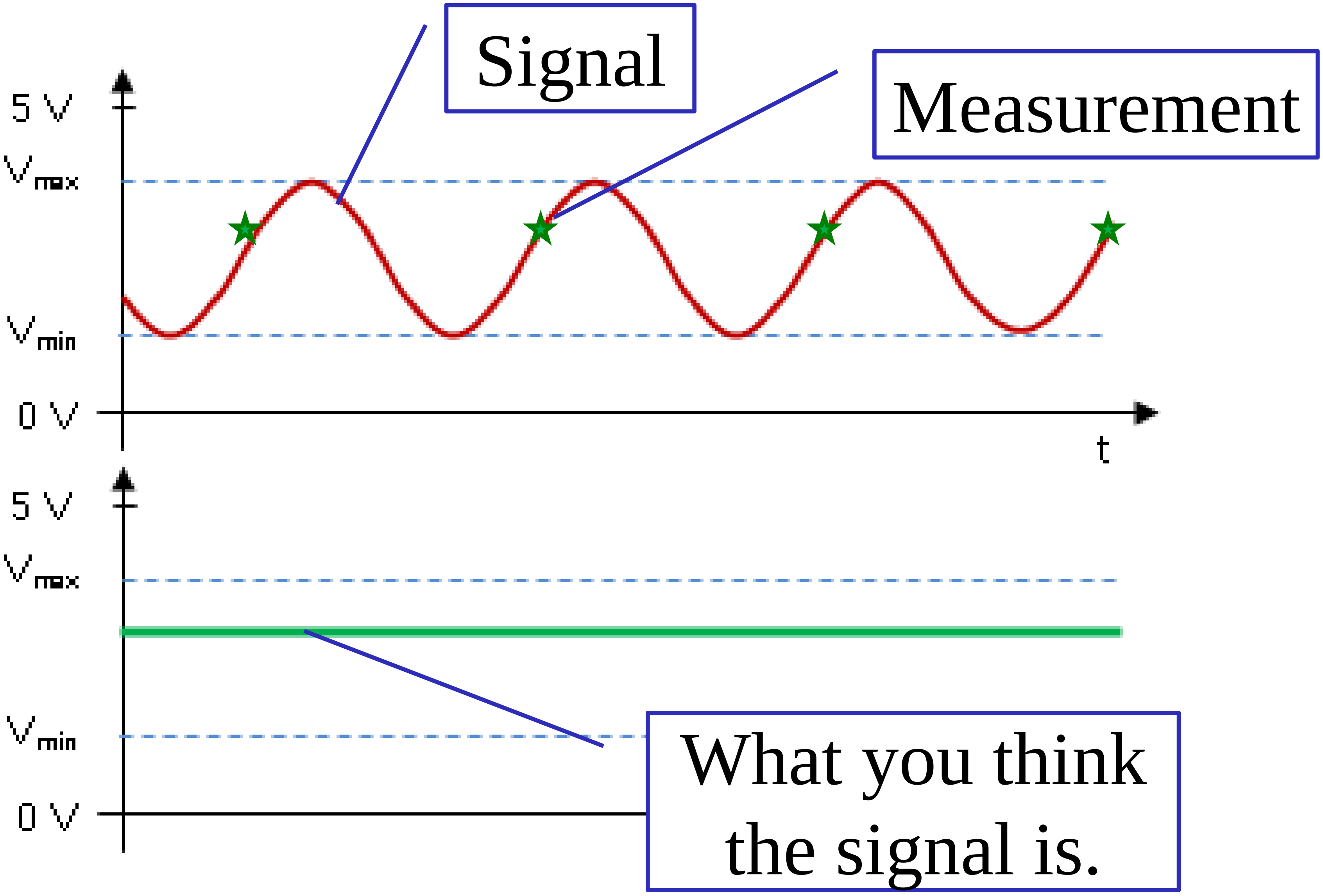
Choosing ADC_N_TAD

- T_{CAPmin} is fixed ~ 1.4 μs (from ref. sheet)
- T_{CAP} = ADC_N_TAD * T_{AD} ≥ 1.4 μs
- If we used previous table to choose ADC_FOSC_N, TAD = 1 μs
- ADC_2_TAD best choice, T_{CAP} = 2 T_{AD}

Sampling Rate

- If we pick the smallest ADC_FOSC_N and ADC_N_TAD for our f_{OSC}, T_{AD} = 1 μs
- Thus it takes (11 + 2) T_{AD} for each ADC measurement, 13 μs
- Sampling Rate = 1 / 13 μs = 0.077 MHz = 77 KHz
- Signal should be several times slower if you want to reproduce the signal

Too Slow Sampling Rate



```
#include <xc.h>
#include “osc.h”

int main (void)
{
  int firstADCvalue, secondADCvalue;
  set_osc_32MHz();                          // change the internal oscillator frequency
              // Configure pins AN0 and AN1 only for ADC operation using
              // VDD and VSS as the references. The digital value is right
              // justified. The other values are default settings
  OpenADC( ADC_FOSC_32 & ADC_2_TAD & ADC_RIGHT_JUST,
         ADC_CH0 & ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS, 13);
  Delay10TCYx(5);                           // Delay for 50TCY to stabilize
  while(1)
    {
     SetChanADC(ADC_CH0);                   // Next read from pin AN0
     ConvertADC();                          // Start ADC operation
     while( BusyADC() );                    // Wait for completion
     firstADCvalue = ReadADC();             // Read result
     SetChanADC(ADC_CH1);                   // Next read from pin AN1
     ConvertADC();                          // Start ADC operation
     while( BusyADC() );                    // Wait for completion
     secondADCvalue = ReadADC();            // Read result
    }
}
```

What is the sampling rate based on the following snippet?

```
set_osc_16MHz();
OpenADC( ADC_FOSC_64 & ADC_4_TAD …);
```

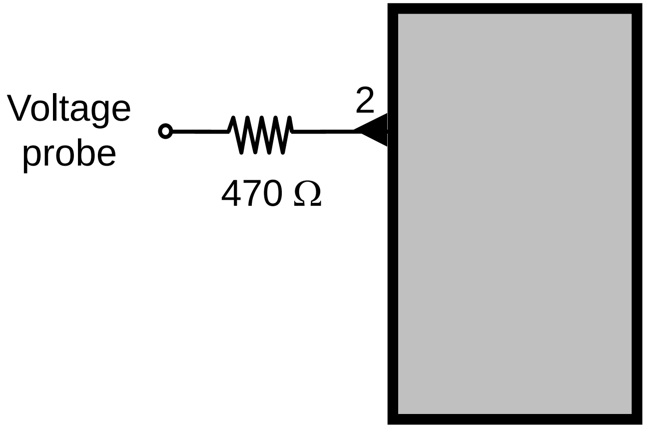
T_{AD} = 64 / 16 MHz = 4 μs

T_{CONV} = (11 + 4) T_{AD} = 60 μs

f_{ADC} = 1 / 60 μs = 16.7 KHz

ADC Probe Wiring

- Must buffer input pin so current is not too large.



Transducers

- Transducers are sensors that convert a physical parameter such as Temperature, pH, light intensity, etc into an electronic signal.
- An electronic signal is a change in resistance, current, or voltage.
- Signal to parameter (say V vs T for a temperature probe) may not be linear.
- May need reference points to calibrate signal, e.g. 0° C or 100° C.