A/D Conversion (Analog Inputs)

ECE 376 Embedded Systems

Jake Glower - Lecture #13

Please visit Bison Academy for corresponding lecture notes, homework sets, and solutions

A/D Conversion (Analog Inputs)

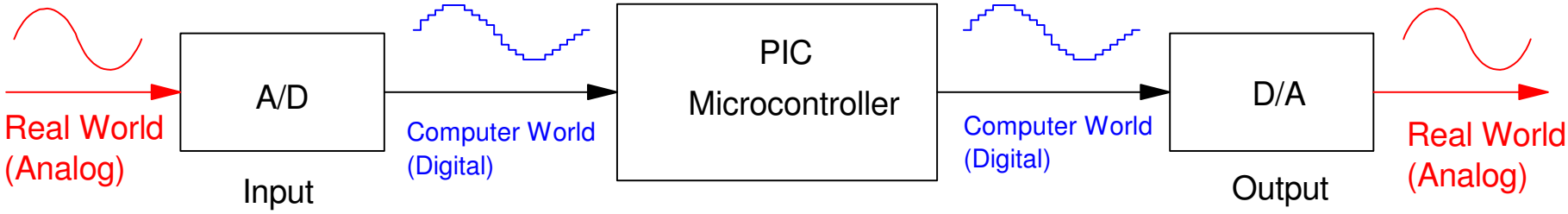
Real World

- Analog

Computer World

- Digital

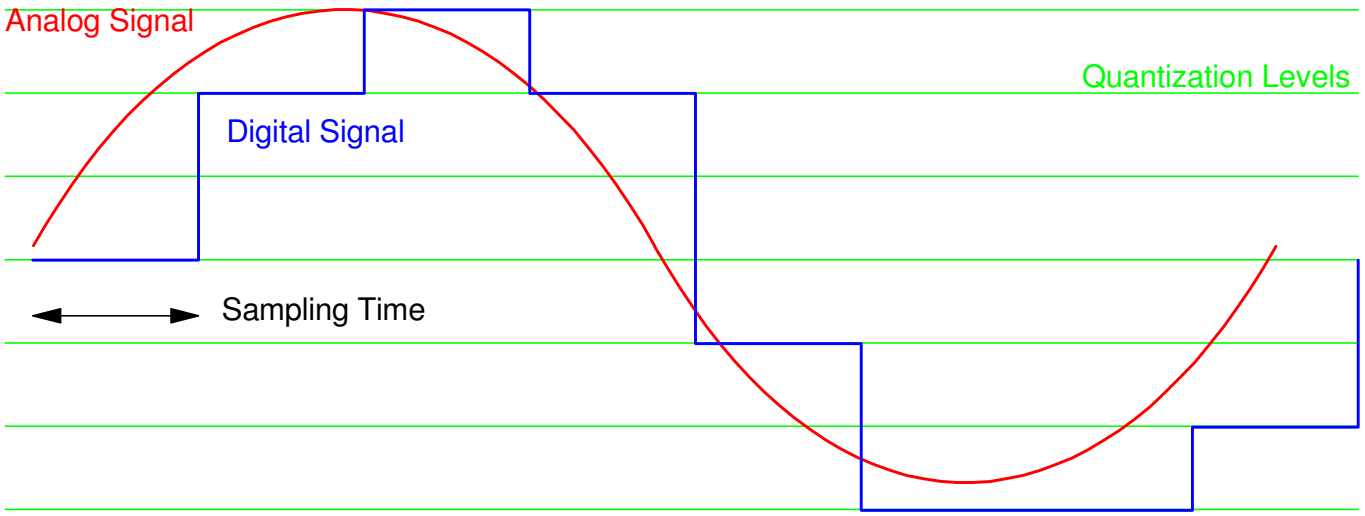
A/D allows a computer to see what's happening in the real world



A/D converters convert analog signals to digital (computer input).
D/A converters convert digital signals to analog (computer output)

Definitions

- A/D: Analog to Digital. Converts real world (analog) to the computer world (digital).
- D/A: Digital to Analog. Converts computer world (digital) to the real world (analog).
- Sampling Rate: The number of samples per second.
- Quantization Level: The resolution of the A/D or D/A converter.
- Quantization Noise: The difference between the analog & digital signal (rounding)



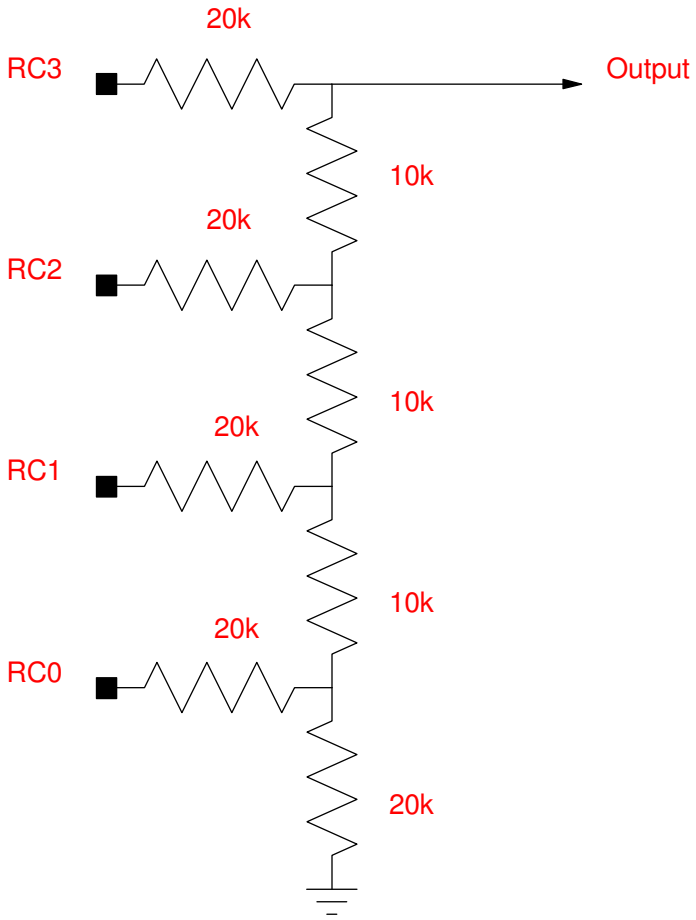
Definitions: Due to using integers, the analog signal (red) is slightly different from the digital signal (blue)

D/A Converters

- Heart of an A/D
- Simplest is R-2R Ladder

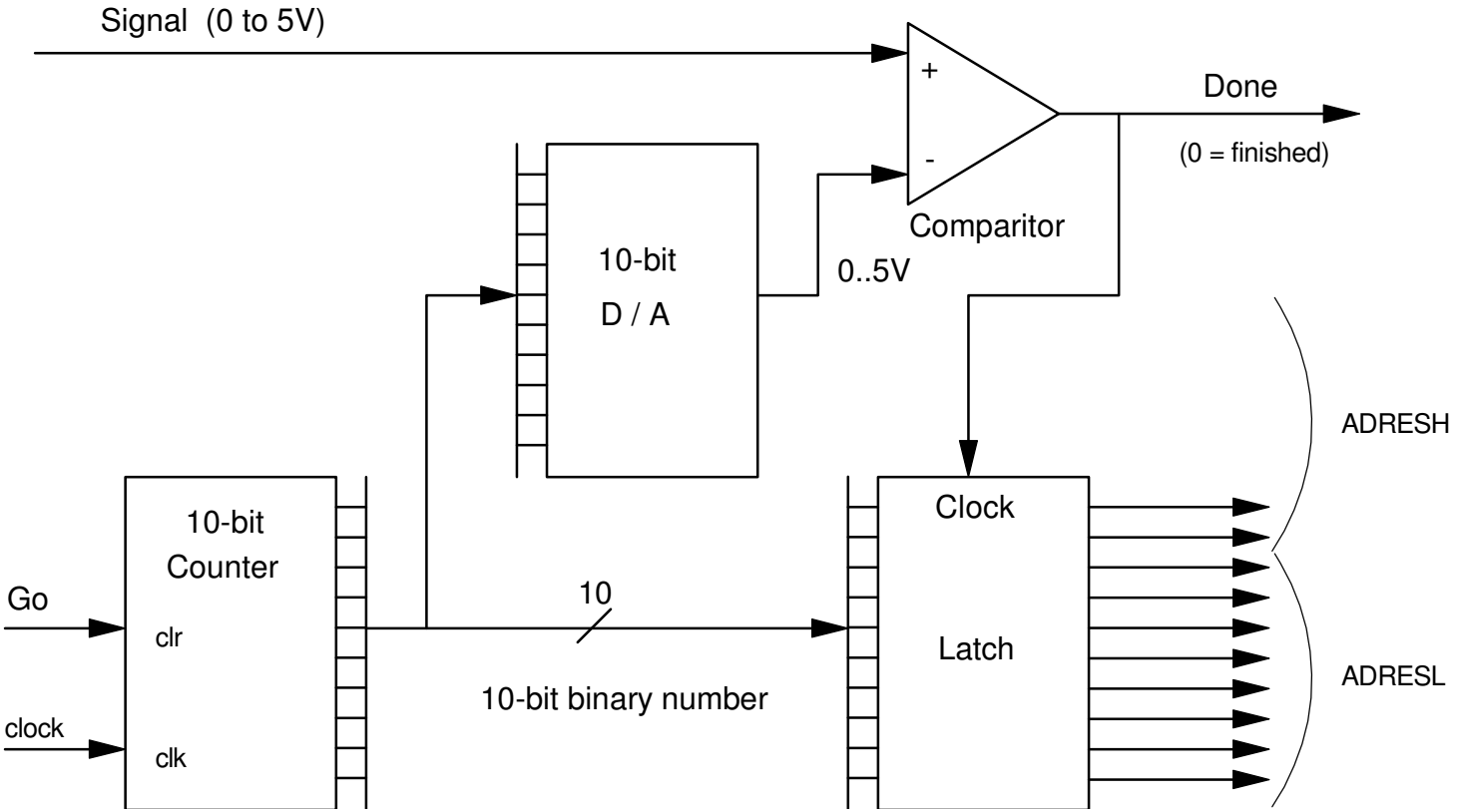
$$V_o = \frac{1}{2}(RC3) + \frac{1}{4}(RC2) + \frac{1}{8}(RC1) + \frac{1}{16}(RC0)$$

$$V_0 = \left(\frac{\text{binary number 0 .. 15}}{16}\right) 5V$$



A/D Converters

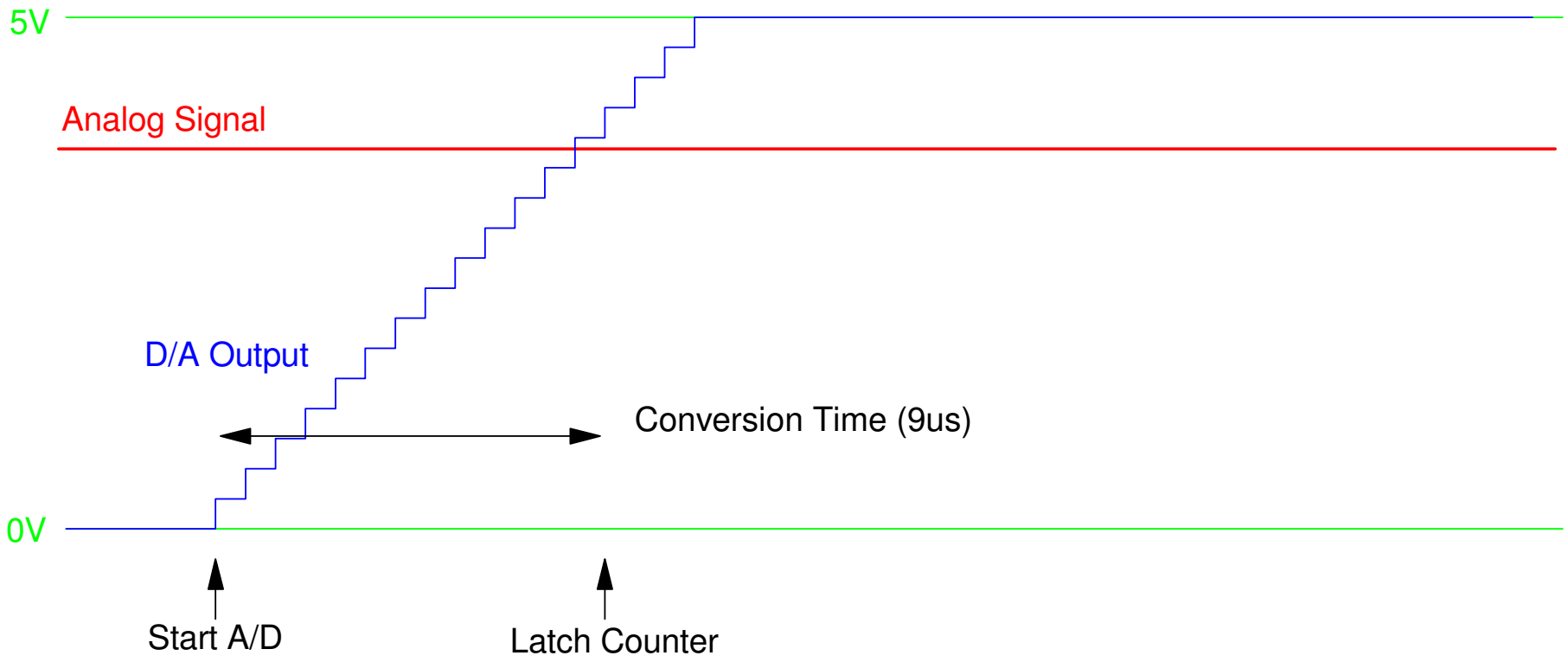
- Start counting at t = 0
- Conversion is Count Value when D/A > Signal



Hardware for an A/D converter

A/D Timing

- Takes about 9us per A/D conversion



PIC 10-Bit A/D

The PIC has a 10-bit A/D

- Result = 0 to 1023

The range is 0 .. 5V

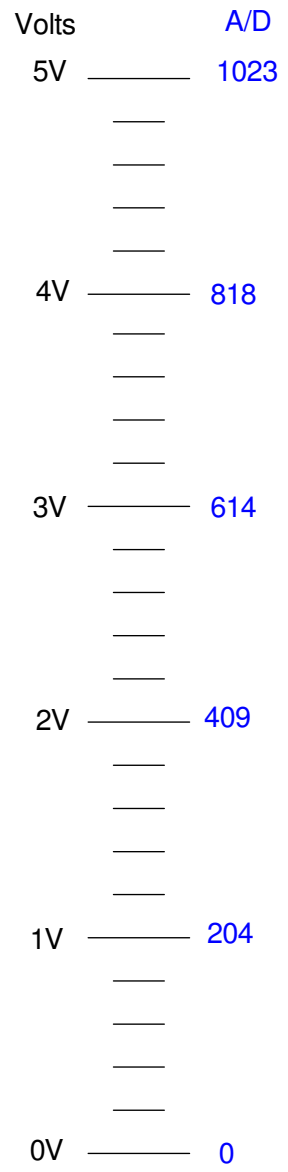
- 0V = 0
- 5V = 1023

The resolution is 4.88mV

$$\frac{5V}{1024} = 4.88; \frac{mV}{count}$$

Converstion time = 9us (approx)

- max sampling rate = 111kHz
- less if you do something with the result



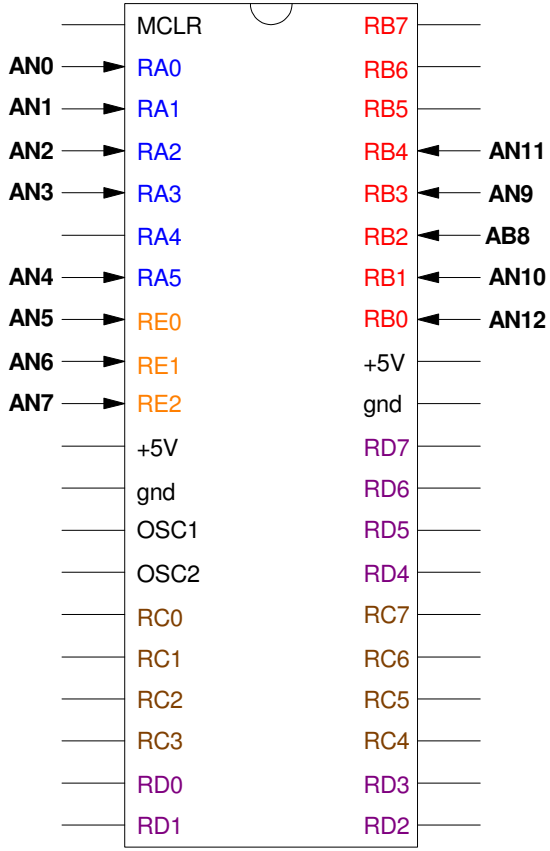
A/D Conversion on the PIC18F4626:

Registers related to A/D conversion

Address	Register Name	Bit							
		7	6	5	4	3	2	1	0
0xFC0	ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
0xFC1	ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
0xFC2	ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GODONE	ADON
0xFC3	ADRES	16 bit register (0..65535)							

ADCON0:

- CHS: Channel to convert: You must wait 14us if you change channels
  - 0000 = Channel 0 (RA0/AN0)
  - 0001 = Channel 1 (RA1/AN1)
  - 0010 = Channel 2 (RA2/AN2)
  - 0011 = Channel 3 (RA3/AN3)
  - 0100 = Channel 4 (RA5/AN4)
  - 0101 = Channel 5 (RE0/AN5)
  - 0110 = Channel 6 (RE1/AN6)
  - 0111 = Channel 7 (RE2/AN7)
  - 1000 = Channel 8 (RB2/AN8)
  - 1001 = Channel 9 (RB3/AN9)
  - 1010 = Channel 10 (RB1/AN10)
  - 1011 = Channel 11 (RB4/AN11)
  - 1100 = Channel 12 (RB0/AN12)
- ADON: 1 = turn on the A/D (and draw an additional 180uA)
- GODONE: Start the A/D conversion. Conversion is complete when bit GODONE = 0 (about 9us later)



ADCON1

- Analog vs. Digital

PCFG3:PCFG0	RB0 AN12	RB4 AN11	RB1 AN10	RB3 AN9	RB2 AN8	RE2 AN7	RE1 AN6	RE0 AN5	RA5 AN4	RA3 AN3	RA2 AN2	RA1 AN1	RA0 AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

ADCON2

ADFM: A/D Result Format Select bit

- 1 = Right justified (C programming - 10 bit result)
- 0 = Left justified (Assembler programming - 8 bit result)

Result from an A/D Conversion 10-bit Result = abcdefghij		
ADFM	ADRESH	ADRESL
0	abcd efgh	ij00 0000
1	0000 00ab	cdef ghij

bit 6 Unimplemented: Read as '0'

bit 5-3 ACQT2:ACQT0: A/D Acquisition Time Select bits

- 110: Automatically restart the A/D conversion every 16th clock
- 000: Manual operation of the A/D (user must set GODONE to start conversions)

bit 2-0 ADCS2:ADCS0: A/D Conversion Clock Select bits

- 101 = FOSC/16 (use with a 20MHz crystal)

Example: Turn on A/D

- PORTA/E are analog inputs, PORTB/C/D are binary
- The conversion will be right justified (ADFM = 1)
- A 20MHz crystal is used. (ADCS = 10: F_{OSC} / 32)

```
// Turn on the A/D converter
    TRISA  = 0xFF;
    TRISE  = 0x0F;
    ADCON2 = 0x85;
    ADCON1 = 0x07;
    ADCON0 = 0x01;

unsigned int A2D_Read(unsigned char c)
{
   unsigned int result;
   unsigned char i;
   c = c & 0x0F;
   ADCON0 = (c << 2) + 0x01;     // set Channel Select
   for (i=0; i<3; i++);          // wait 2.4us (approx)
   GODONE = 1;                   // start the A/D conversion
   while(GODONE);                // wait until complete (approx 8us)
   return(ADRES);
}
```

Fun with A/D Converters

Electronic Trombone:

- Set the frequency using the analog input. Play a note when you press RB0.

LED Flashlight:

- Vary the brightness of a NeoPixel using the potentiometer from 0% to 100% on in 255 steps.

LED Flashlight (take 2):

- Vary the color of the NeoPixel using the potentiometer

Stepper Motor Position Control (Telerobotics):

- Have a stepper motor follow the position of the potentiometer from 0 steps (A/D = 0) to 200 steps (A/D = 1023).

Multi-Meter.

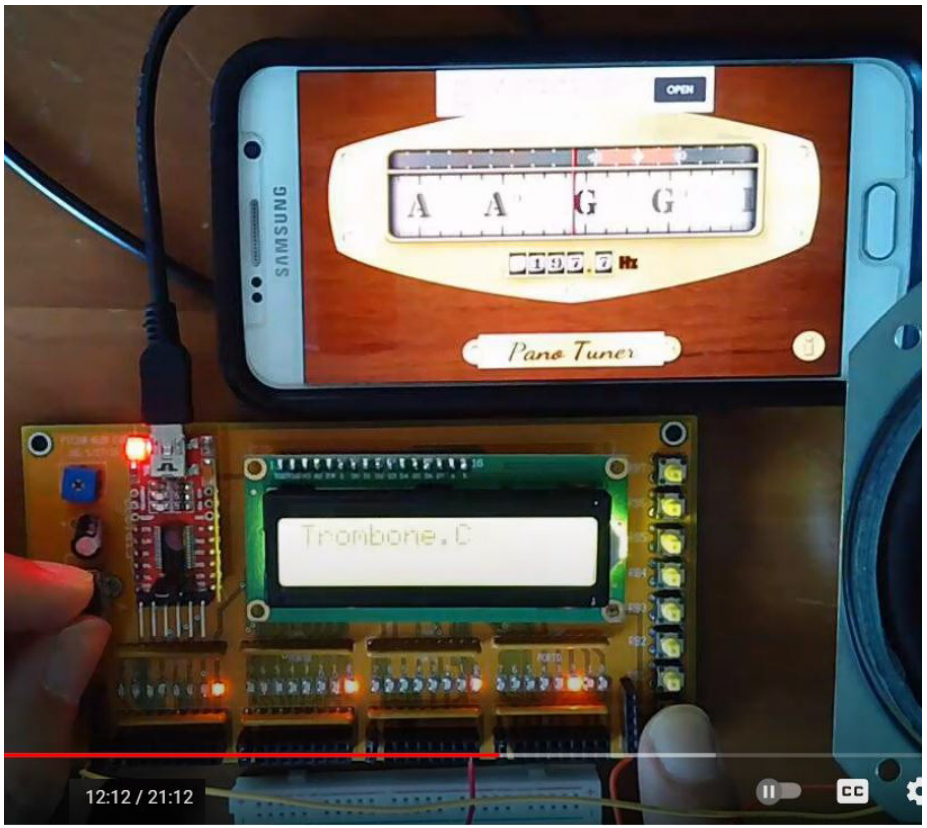
- Turn your PIC into a volt / ohm / light / temperature meter.

Electronic Trombone:

Requirement: Play notes ranging from 100Hz to 200Hz on pin RC0

- 100Hz = 0V
- 200Hz = 5V

```
while(1) {
   A2D = A2D_Read(0);
   N = 2243 - 1.4809*A2D;
   if(PORTB)  RC0 = !RC0;
   for(i=0; i<N; i++);
   }
```



Experimentally: (needs adjustment)

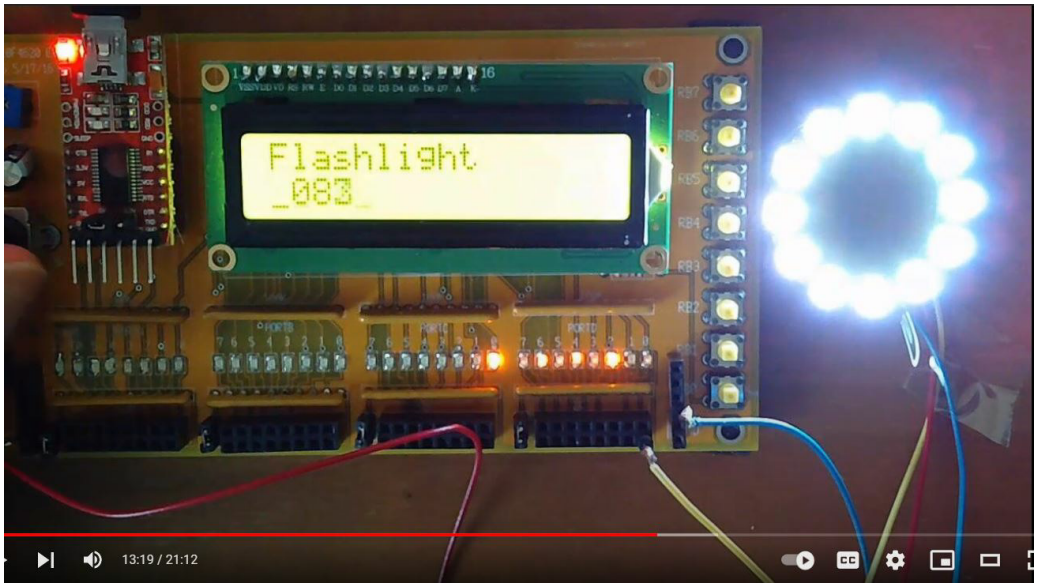
- 0V A/D = 0 f = 103.77 Hz
- 5V A/D = 1023 f = 278.82 Hz

LED Flashlight: Brightness Control

Vary the brightness of a NeoPixel from 0 (0V) to 255 (5V)

- Color = White

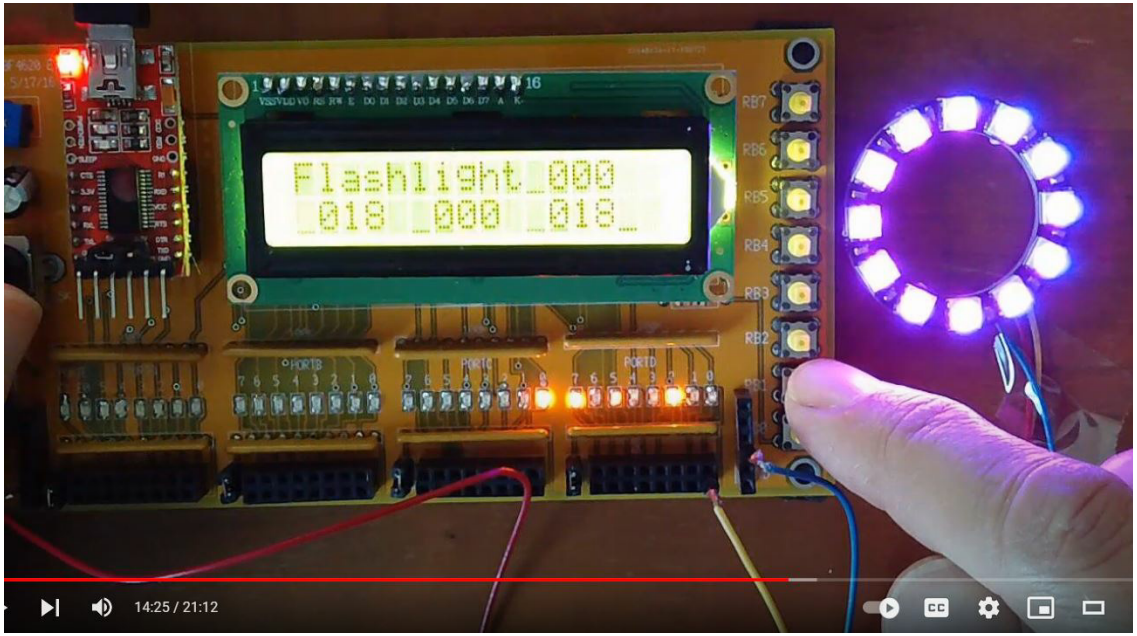
```
while(1) {
   A2D   =  A2D_Read(0);
   X  =  A2D/4;
   LCD_Move(1,0);
   LCD_Out(X, 0, 3);

   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);
   NeoPixel_Display(X, X, X);

   Wait(1);
}
```



LED Flashlight: Hue Control

Set each color separately

- RB2 Blue
- RB1: Green
- RB0: Red



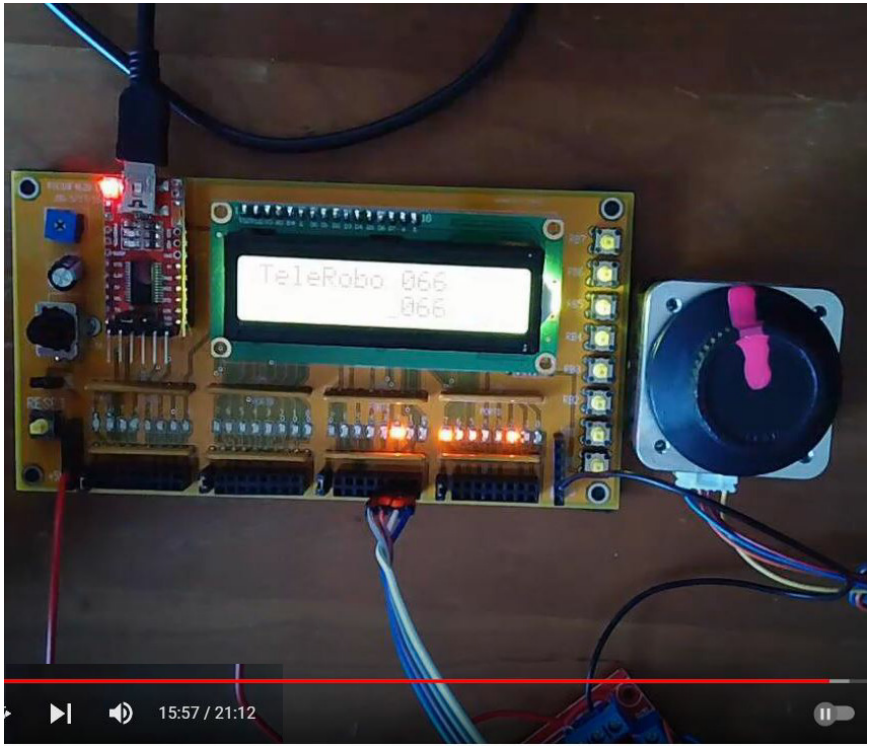

```
while(1) {
   A2D   =  A2D_Read(0);
   X = A2D / 4;

   if (RB0) RED   = X;
   if (RB1) GREEN = X;
   if (RB2) BLUE  = X;

   LCD_Move(0,10);  LCD_Out(X, 0, 3);
   LCD_Move(1, 0);  LCD_Out(RED, 0, 3);
   LCD_Move(1, 5);  LCD_Out(GREEN, 0, 3);
   LCD_Move(1,10);  LCD_Out(BLUE, 0, 3);

   NeoPixel_Display(RED, GREEN, BLUE);
   Wait_ms(50);
}
```

Stepper Motor: Position Control (Telerobotics)

Have the stepper motor mirror the potentiometer

- 0V = 0 steps
- 5V = 200 steps

```
while(1) {
   A2D = A2D_Read(0);
   REF = A2D * 0.1955;

   if (STEP < REF) STEP = STEP + 1;
   if (STEP > REF) STEP = STEP - 1;

   PORTC = TABLE[STEP % 4];

   LCD_Move(0,8);  LCD_Out(REF, 0);
   LCD_Move(1,8);  LCD_Out(STEP, 0);

   Wait_ms(20);

   }
```



Stepper Motor: Light Sensor

Make the stepper motor indicate the light level as

- 1 Lux 0 steps
- 100 Lux 200 steps

This is the same as the previous solution:

- First, convert light to voltage.
- Once it's a voltage, read the voltage with the A/D input and use that to control the stepper position.

Multi-Meter

Turn your PIC board into

- A volt meter
- An Ohm meter
- A light sensor
- A temperature sensor

Volt Meter:

The A/D reading is proportional to voltage

- 0 = 0.00V
- 1023 = 5.00V

The calibration function is then

Volt = 0.0047776 * A2D

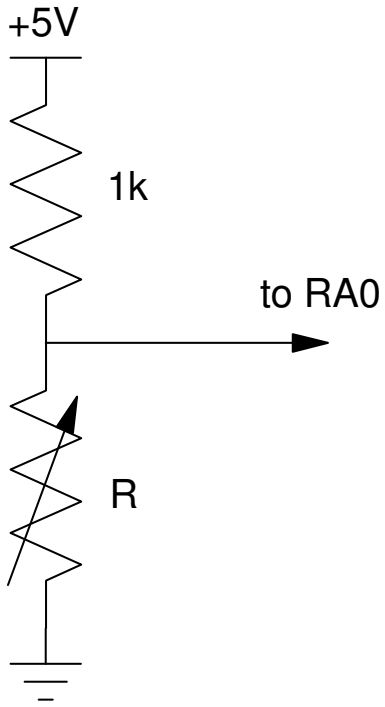
```
while(1) {
   A2D = A2D_Read(0);
   VOLT = 0.488 * A2D;
   LCD_Move(1,8);  LCD_Out(VOLT, 5, 2);
   }
```

Ohm Meter:

$$V = \left(\frac{R}{R+1000}\right)5$$

$$A/D = \left(\frac{R}{R+1000}\right)1023$$

$$R = \left(\frac{A/D}{1023-A/D}\right)1000\Omega$$



Code:

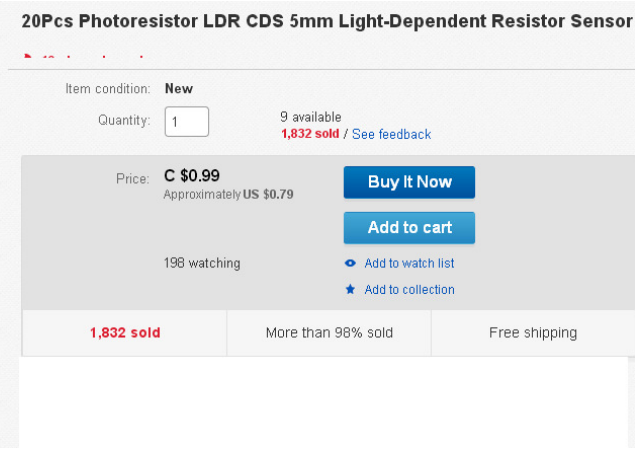
```
while(1) {

   A2D = A2D_Read(0);
   OHM = 1000.0 * (A2D / (1023.0 - A2D) );

   LCD_Move(1,8);  LCD_Out(OHM, 5, 0);

   Wait_ms(10);

 }
```

Light Meter:

$$R \approx \frac{100{,}000}{Lux}$$

$$Lux = \frac{100{,}000}{R}$$

$$Lux = \frac{(1023 - A/D)}{A/D} \cdot 100$$




```
while(1) {

   A2D = A2D_Read(0);

   LUX = ( (1023.0 - A2D) / A2D ) * 100;

   LCD_Move(1,0);  LCD_Out(A2D, 5, 0);
   LCD_Move(1,8);  LCD_Out(VOLT, 5, 2);

   Wait_ms(10);

 }
```

Temperature Sensor

$$R = 1000 \cdot \exp\left(\frac{3930}{T+273} - \frac{3930}{298}\right)\ \Omega$$

$$T = \left(\frac{3930}{\ln\left(\frac{A/D}{1023-A/D}\right)+\frac{3930}{298}}\right) - 273$$

Code: (needs math.h)

```
while(1) {

   A2D = A2D_Read(0);
   CELSIUS = 39300. / ( log( A2D / (1023. - A2D) ) + 13.1879 ) - 2730;

   LCD_Move(1,8);  LCD_Out(CELSIUS, 5, 1);
   }
```

Summary

The analog input on a PIC reads 0..5V as 0..1023 (10-bit A/D)

- This gives you another input to your program
- It's value can be adjusted as the program runs

The analog input also allows you to measure

- Voltage
- Resistance
- Temperature (with a thermistor),
- Lux (with a photo-resistor), or
- Anything else which can be converted to a voltage