More Fun with A/D Converters

The A/D input allows you to input numbers (0 to 1023) into the PIC processor with a potentiometer. This illustrates some of the things this allows you to do:

- 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).
  - This also makes the stepper motor a temperature indicator if the input voltage is temperature
  - Or a light indicator
- Stepper Motor Speed Controller: Have the speed of the stepper motor set by the analog input
- 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 as you press RB0. The frequency is continuously adjustable using the analog input (potentiometer on your PIC board).

From previous code, the following routine plays notes C2 to C2

```
while(1) {
 if (PORTB) RC0 = !RC0;
 if (RB0) for(i=0; i<4771; i++);
 if (RB1) for(i=0; i<4250; i++);
 if (RB2) for(i=0; i<3786; i++);
 if (RB3) for(i=0; i<3574; i++);
 if (RB4) for(i=0; i<3184; i++);
 if (RB5) for(i=0; i<2837; i++);
 if (RB6) for(i=0; i<2527; i++);
 if (RB7) for(i=0; i<2385; i++);
 }
```

If you replace the hard-coded numbers with a number based upon the A/D reading, you can vary the frequency on the fly. In C, the executing time depends upon the code. To get an accurate measure, start with code close to what we'll need in the end:

```
while(1) {
   A2D = A2D_Read(0);
   N = 2000 - 0.5678*A2D;
   if(PORTB)  RC0 = !RC0;
   for(i=0; i<N; i++);
   }
```

Experimentally, the extremes produce:

- A/D = 0 N = 2000 f = 116.04Hz
- A/D = 1023 N = 1419 f = 154.39Hz

From this,

$$Hz \approx -0.0660N + 248$$

or

- 100Hz N = 2243 A/D = 0
- 200Hz N = 728 A/D = 1023

giving the function

$$N = 2243 - 1.4809 \cdot A/D$$

This results in

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

A little more adjustment would get you from 100Hz to 200Hz as you adjust the potentiometer.

LED Flashlight: Brightness Control

Previous code allowed us to drive a NeoPixel using a PIC processor. The global variables RED, GREEN, and BLUE set the brightness of the NeoPixel from 000 (off or 0mA) to 255 (100% on or 20mA).

To change the brightness of the NeoPixels using the A/D converter, use the following code.

```
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

Instead of making all colors the same intensity, producing white light, update each color one at a time. As you hold down one of the buttons, the brightness of that color changes according do the A/D input:

- RB2 Blue
- RB1: Green
- RB0: Red

One version of the main routine to do this:

```
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);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);
   NeoPixel_Display(RED, GREEN, BLUE);

   Wait(5);
}
```

Stepper Motor: Position Control (Telerobotics)

Connect the potentiometer to your arm so that as you move, the potentiometer voltages changes with you. Have the stepper motor follow the potentiometer as

- 0V = 0 steps
- 5V = 200 steps
- Proportional in-between

```
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

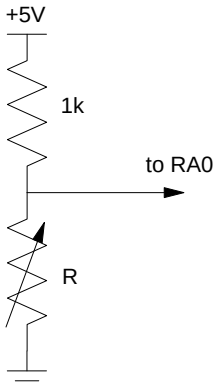
$$Volt = 0.0048876 \cdot A/D$$

If you want to display this to 2 decimal places, scale this by 100 (so 100 means 1.00 Volts)

Code:

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

Ohm Meter: You can convert resistance to voltage using a voltage divider. Assuming a 1k resistor



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

or the A/D reading will be

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

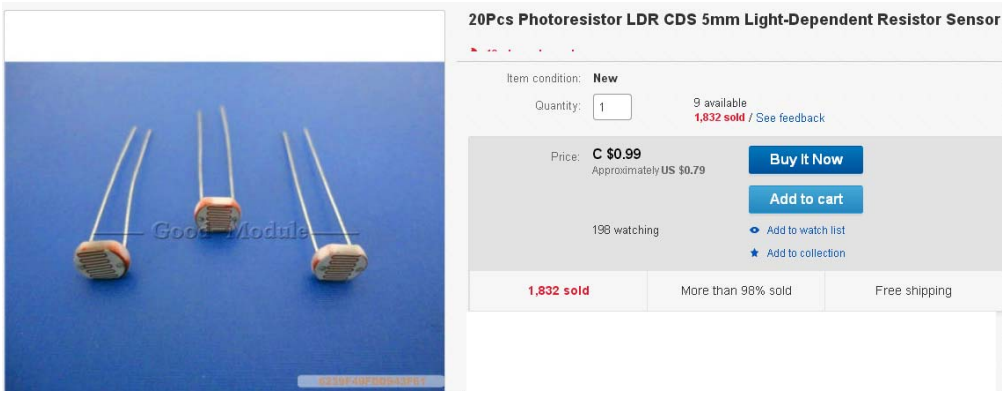
Solving backwards, you can compute the resistance given the A/D reading

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

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: The light sensor in your lab kit has a light-dependent resistor:



with a light - resistance relationship of

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

Then

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

Substituting for R from the previous sensor

$$Lux = \frac{100{,}000}{\left(\frac{A/D}{1023 - A/D}\right)1000}$$

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

Code:

```
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: Also in your lab kits is a temperature sensor:

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

where T is the temperature in degrees Kelvin

$$T_{Kelvin} = Celsius + 273$$

Solving for T

$$T = \left(\frac{3930}{\ln\left(\frac{R}{1000}\right) + \frac{3930}{298}}\right) \quad \text{Kelvin}$$

$$T = \left(\frac{3930}{\ln\left(\frac{R}{1000}\right) + \frac{3930}{298}}\right) - 273 \quad \text{Celsius}$$

Substituting for R

$$T = \left(\frac{3930}{\ln\left(\frac{\left(\frac{A/D}{1023-A/D}\right)1000}{1000}\right) + \frac{3930}{298}}\right) - 273 = \left(\frac{3930}{\ln\left(\frac{A/D}{1023-A/D}\right) + \frac{3930}{298}}\right) - 273$$

Code: Include Math.h for the log function (note: in C, log() is base e, log10() is base 10)

```
#include   <pic18.h>
#include    <math.h>
```

The main routine is then (multiplying T by 10x so you can display temperature to one decimal point)

```
while(1) {

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

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