ACTIVITY #2: SIGNAL CONDITIONING

The analog to digital converter resolves an analog value to a digital value. The ADC0831, as configured, coverts an input voltage of 0 to 5 volts to an 8-bit value (byte) of 0 to 255, respectively, as shown in Figure 6-4. The reason for the value of 255 is that the ADC0831 is an 8-bit ADC. The maximum value of 8 bits is 255. When looking at a binary value, such as 11111111, each position from right to left is a higher power of 2, with the Least Significant Bit (LSB – right most) having a value of 2^0, or 1. The Most Significant Bit (MSB – left most) has a value of 2^7 or 128. A binary value of 11111111 would be the sum of each position with a 1 or:

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

A binary value of 10011001 converted to decimal would be:

$$128 + 0 + 0 + 16 + 8 + 0 + 0 + 1 = 153$$

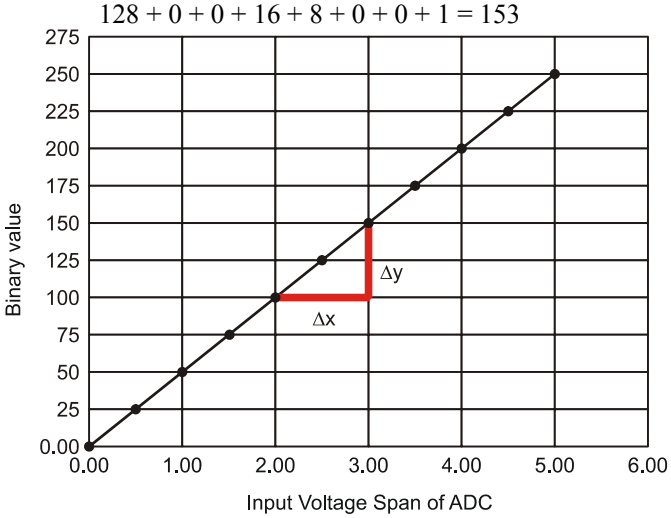


Figure 6-4
Voltage to Binary Conversion Transfer Function

Figure 6-4 shows that the input voltage (0-5 V) is resolved to a binary value (0-255). This linear transfer function can be written as a line equation:

$$y = mx + b$$

Where m is the slope of the line, and b is the y-intercept (where the line crosses the Y-axis), which is 0 in this case.

The slope, m, is the change in y for a given change in x. $m = \Delta y/\Delta x$. In Figure 6-4 the denoted Δy is around 50 for a given Δx of 1 V, for a slope of 50/1 V. While we are only approximating the changes from the graph, any two points on the linear line will yield the same slope. It's best to choose two known values. We know that a voltage of 0 V in will result in a binary value of 0, and a voltage of 5 V in will result in a binary value of 255. Given these two points:

$$m = \Delta y/\Delta x = (255-0)/(5\text{ V}-0\text{ V}) = 51/\text{V}$$

This provides a general equation of:

$$y = (51/\text{V})x$$

Testing the input value of 2 V:

$$y = (51/\text{V})2\text{V} + 0 = 102$$

Another way of stating the general line equation is as a transfer function for our specific system:

$$\text{Binary Value} = (\Delta\text{Output}/\Delta\text{Input})\text{Input}$$
$$\text{Binary Value} = (\Delta\text{Binary Value}/\Delta\text{Vin})\text{Vin}$$
$$\text{Binary Value} = (51/\text{V})\text{Vin}$$

Δy is ΔOutput since that is how much the ADC binary value will change for a given Δx, or change in input to the ADC.

Since there are only 255 possible steps between 00000000 and 11111111, the ADC is limited to how finely it can resolve a voltage. What would be the bit value for voltages of 1.05 V and 1.06 V? 53.55 and 54.06. But since only integer values can be used, they would both have byte counts of 54, rounding to the nearest integer. In terms of temperature, this change between 105 °F and 106 °F would not be measurable.

When the BASIC Stamp processes the binary value, the process is reversed where the byte value is converted into a temperature using code. In performing the conversion, a

line equation can again be used where the change in output temperature will be 0-500 °F (based on the transfer function of 0.01 V/°F) for the change in input value of 0 to 255.

Temp = m(Bit Value) + b where b = 0
m = ΔOutput/ΔInput = ΔTemperature Span/ΔByte Value = 500 °F/255 = 1.96 °F
Temp = 1.96 °F x Byte Value

Each change of 1 in byte value will signify a temperature change of 1.96. Since the ADC is limited on resolution, the output will have distinct steps as you have probably noted in many experiments.

Example Program: AdcSpanOffset.bs2

√ Using the same circuit as Activity #1, enter, save and run AdcSpanOffset.bs2.

```
' -----[ Title ]-------------------------------------------------------------
' Process Control - AdcSpanOffset.bs2
' Tests the spanning and offset input range of the ADC 0831 using PWM
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Declarations ]-------------------------------------------------------

ADC_ByteValue  VAR   Byte       ' Analog to Digital Converter data
V_Offset       VAR   Byte       ' Offset voltage read from StampPlot
V_Span         VAR   Byte       ' Span voltage read from StampPlot
TempF          VAR   Word       ' Calculated temp in hundredths of degree F

ADC_CS      PIN 13              ' ADC Chip Select pin
ADC_Clk     PIN 14              ' ADC Clock pin
ADC_Dout    PIN 15              ' ADC Data output

ADC_VRef    PIN 10              ' Pin for PWM to set ADC voltage span
ADC_Vminus  PIN 11              ' Pin for PWM to set ADC Offset

' -----[ Initialization ]-----------------------------------------------------
PAUSE 1000                      ' Allow connection stabilization

' -----[ Main Routine ]-------------------------------------------------------
DO
  GOSUB ReadSP
  GOSUB SetADC
  GOSUB ReadADC
  GOSUB CalcTemp
  GOSUB UpdateSP
```

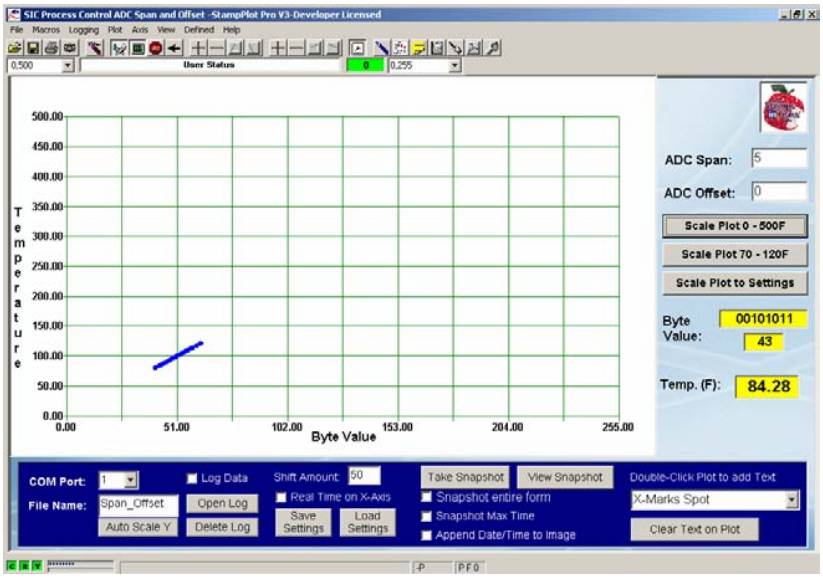
```
  GOSUB PlotPoint
  PAUSE 100
LOOP

' -----[ Subroutines ]------------------------------------------------------
ReadSP:
  DEBUG CR,"!READ [(txtADCoffset),*,10]",CR ' obtain offset volt. in tenths
  DEBUGIN DEC V_Offset
  PAUSE 50
  DEBUG "!READ [(txtADCspan),*,10]",CR      ' obtain span voltage in tenths
  DEBUGIN DEC V_Span
  PAUSE 50
  RETURN

SetADC:
  PWM ADC_Vminus, V_Offset * 255/50,100     ' Apply PWM to set offset volt.
  PWM ADC_Vref, V_Span * 255/50,100         ' Apply PWM to set span voltage
  RETURN

ReadADC:                ' Read ADC 0831
  LOW ADC_CS            ' Enable chip
  SHIFTIN ADC_Dout, ADC_Clk, MSBPOST,[ADC_ByteValue\9] ' Clock in ADC data
  HIGH ADC_CS           ' Disable ADC
  RETURN

CalcTemp:   ' y = mx + b
            ' where y=temp,
            ' m = (change in output)/(change in input) = voltage Span/255
            ' x = ADC value read, b = offset, y = temperature in hundredths
            ' temperature = (Span/255)Byte + Offset
  TempF = (V_Span * 1000)/255 * ADC_ByteValue + (V_Offset * 1000)
  RETURN

UpdateSP:
  DEBUG "!O txtByteBin = ", BIN8 ADC_ByteValue,CR, ' Update w/ binary ADC val
        "!O txtByte = ", DEC ADC_ByteValue,CR      ' Update w/ decimal ADC val
  DEBUG "!O txtTemp = [", DEC TempF,",/,100]",CR   ' Update w/ temperature/100
  RETURN

PlotPoint:
  DEBUG "!FCIR (txtByte),(txtTemp),0.3A,(WHITE)",CR  ' Plot white circle at
  PAUSE 100                                          '  byte, Temp as X,Y
  DEBUG "!FCIR ,,,(BLUE)",CR                         ' Plot again in blue
  RETURN
```

- √ Load StampPlot macro sic_pc_adc_span_offset.spm.
- √ Verify that "ADC Span" is set to 5 and "ADC Offset" is set to 0 in StampPlot.
- √ Connect and plot. A small blinking dot will appear on the screen, marking the temperature.

- √ Read down from the dot to determine the byte count read from the ADC. The display is not a ".vs time" plot, so if the temperature doesn't change much, you will only see a few dots.
- √ Pinch the LM34, or briefly apply a heat source, to change the temperature reading.

Figure 6-5 is a plot of the byte count read from the ADC and the calculated temperature. Over a range of 0 °F to 500 °F, there appears to be very good resolution based on how close the individual points are.

Figure 6-5 ADC Byte Count vs. Calculated Temperature - Full Range

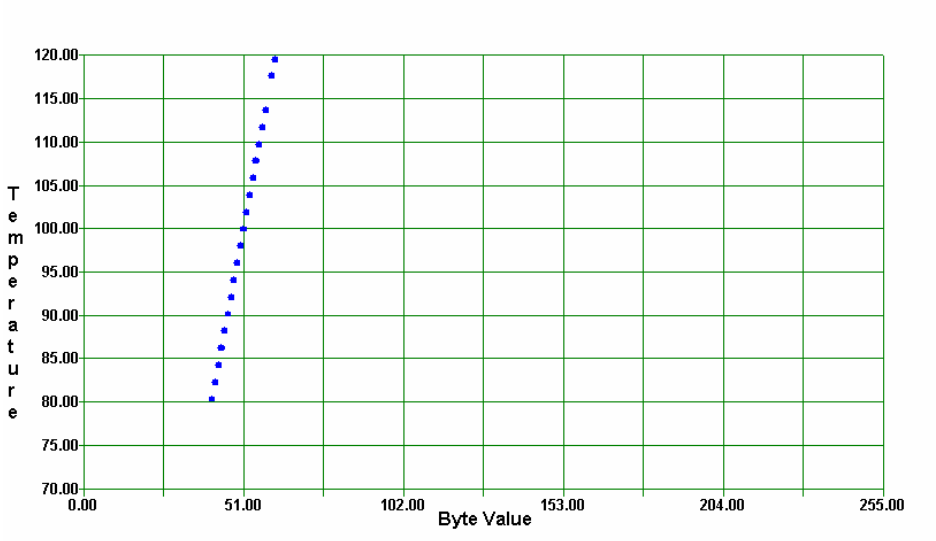


If we were monitoring and controlling temperature over 0 °F to 500 °F, this may be adequate resolution. But our temperatures of focus will be in the 70 °F to 120 °F range. How finely can the system monitor and control over this range?

- √ Click the "Scale Plot 70−120F" button on the interface.

Figure 6-6 is a capture of the plotted data for this range. Doesn't look so good now, does it? With the current resolution and span, the system can only resolve temperatures to 1.96 °F. This isn't very good for precise temperature monitoring and control.

Figure 6-6 ADC Byte Count vs. Calculated Temperature - Narrow Range



For better resolution the system needs to be modified to focus on the temperatures of interest. One means to do this would be amplify the voltage before converting to a digital value.

Consider (but do not build) the schematic in Figure 6-7. What effect would it have on the resolution?

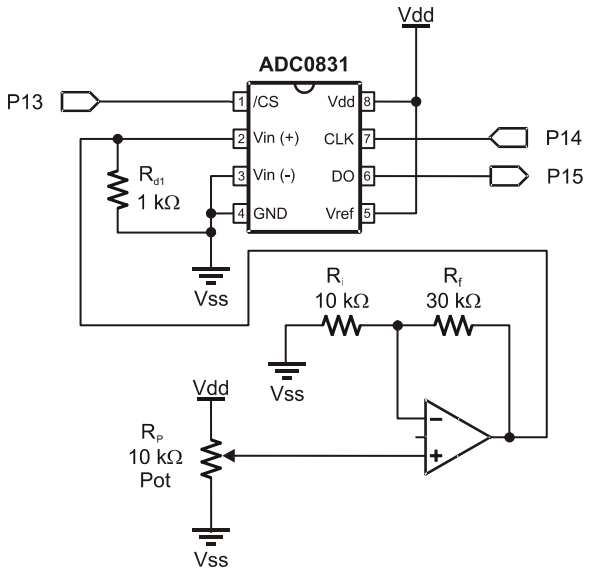


Figure 6-7
Voltage Monitoring With Gain

DO NOT BUILD

Recall that the op-amp is configured in a non-inverting configuration where the gain is:

$$R_f/R_i + 1$$

For the given values, the gain is:

$$30\ k\Omega/10\ k\Omega + 1 = 3+1 = 4$$

For the potentiometer range of 0 to 5 volts, this would provide an output of 0 to 20 V (providing that the op-amp was powered from a voltage above 20 V). The ADC is still converting the voltage range of 0 to 5 V to a byte count of 0 to 255. Consider what occurs to the slope of the graph in Figure 6-8 of sensor voltage to voltage ADC and byte value when gain is applied.

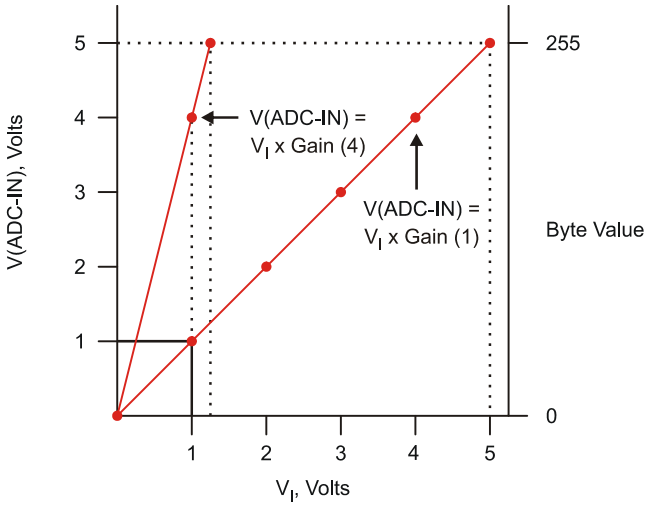


Figure 6-8
Vin vs. ADC Voltages

With no gain, or actually a gain of 1 or unity gain, the voltage to the ADC is the voltage sensed at the ADC. When the op-amp is used with a gain of 4, an input voltage of 1.25 V will provide 5 V to the ADC. Note the slope of the line:

$$m = \Delta Output/\Delta Input = (5-0)/(1.25-0) = 5/1.25 = 4$$

The slope of the line is also the gain, giving a line equation of:

$$VADC = 4(ADC\text{-}IN) + 0$$

What is the resolution of the circuit now? In comparing the applied voltage to the bit count: 1.25 V/255 = 0.0049 V/bit value. In terms of temperature:

$$125\ °F/255 = .49\ °F/bit\ value$$

...which is 4 times the resolution previously measured.

In terms of byte value, the resolution is 255/1.25 V or 204/V. Our high temperature of interest, 120 °F (1.2 V), would provide a byte value of 1.2 V x 204 = 244.8 or 245. The low temperature of interest, 70 °F (0.7 V), would provide a byte value of 0.7 x 204 = 142.8 or 143. So the temperature range of interest covers a byte value change of 245−143 = 102. The majority of our available byte values (0 to 101 and 246 to 255) are still outside the temperature range of interest.

What line equation would best serve the needs of the monitoring and control system? Figure 6-9 is the ideal line needed to convert the temperature range of interest to a digital value for measurement. The temperature range of 70 °F (0.7 V) to 120 °F (1.2 V) is amplified and covers 0 to 5 volts into the ADC for byte values of 0 to 255.

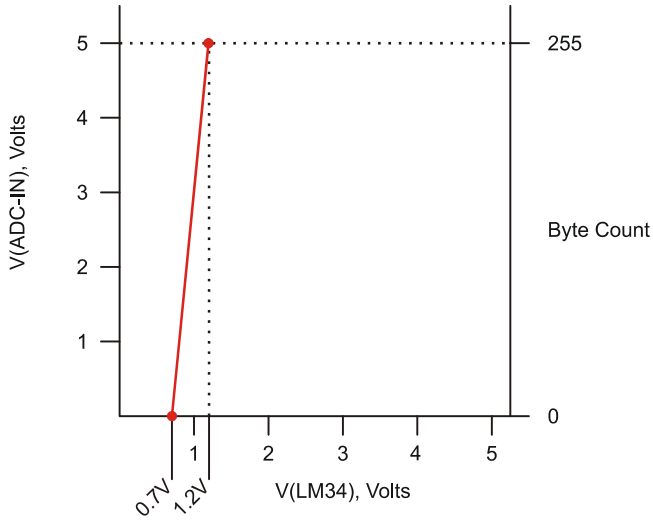


Figure 6-9
Ideal Conversion for
70 °F-120 °F

Let's analyze the line and develop the line equation.

$$y = mx + b$$
$$m = \Delta Output/\Delta Input = (5-0\ V)/(1.2-0.7\ V) = 5\ V/0.5\ V = 10$$

Since the line does not cross at (0,0), b can be calculated given any single known point, such as 0.7 V and 0 V or 1.2 V and 5 V.

$5\ V = 10(1.2\ V) + b$
$5\ V - 12\ V = b$
$b = -7\ V$
$y = 10(x) - 7\ V$ or $VADC = 10(VIN) - 7\ V$

From this equation we can see that a gain of 10 is required and 7 must be subtracted from the product. There are op-amp configurations that can add or subtract voltages, called summing amplifiers. Figure 6-10 illustrates an op-amp circuit which performs the equation of $V_O = 10(V_I) - 7\ V$.

This circuit utilizes a two-stage op-amp configuration to perform the equation or transfer function. Stage 1 provides the gain using an inverting amplifier. The Stage 2 amplifier is configured as a summing amplifier.

Stage 1:

$$V_{S1} = (-R_{f1}/R_{i1})V_I$$
$$= (-100\ k\Omega/10\ k\Omega)V_I$$
$$V_{S1} = -10\ V_I$$

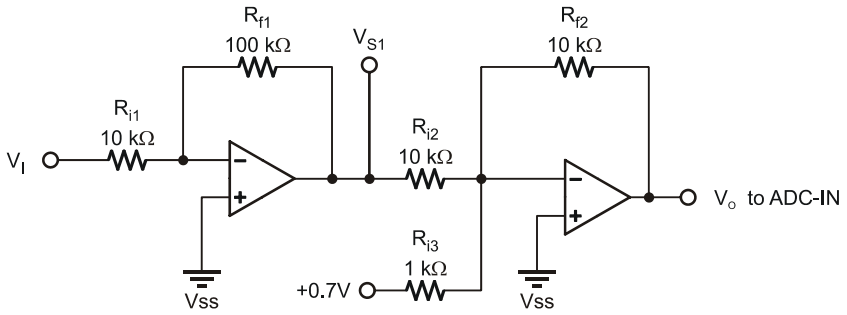
Stage 2:

$$V_O = V_{S1}(-R_{f2}/R_{i2}) + V_{offset}\ (-R_{f2}/R_{i3})$$
$$= V_{S1}\ (-10\ k\Omega/10\ k\Omega) + 0.7\ (-10\ k\Omega/1\ k\Omega)$$
$$= V_{S1}\ (-1) + 0.7\ (-10)$$
$$V_O = -V_{S1} - 7$$

Combining the results of the two stages provides the final voltage into the ADC.

$$V_O = -V_{S1} - 7$$
$$= -(-10\ V_I) - 7$$
$$V_O = 10\ V_I - 7$$

Figure 6-10 Amplify and Offset Signal with Op-Amps *(DO NOT BUILD)*



Testing it out with a voltage of 1.2 V at V_I representing 120 °F:

Stage 1: $-10(1.2\text{ V}) = -12\text{ V}$

Stage 2: $-(-12\text{ V}) - 7\text{ V} = 5\text{ V}$

Of course, accomplishing this requires supply voltages in excess of +/− 12 V for the op-amps. Another option is to offset the input prior to amplifying the voltage.

$$V_O = 10(V_I - 0.7)$$

Consider the differential amplifier in Figure 6-11. Recall that this configuration amplifies the difference between the two input voltages.

The formula for this configuration is:

$$V_O = (V_{I1} - V_{I2})(R_f/R_i)$$

where V_{I1} is the voltage from the LM34 and V_{I2} is the offset voltage of 0.7 V. For the voltage of 1.2V from the sensor, relating to a temperature of 120 °F:

$$V_{ADC} = (1.2\text{ V} - 0.7\text{ V})(100\text{ k}\Omega/10\text{ k}\Omega) = (0.5\text{ V})(10) = 5\text{ V}$$

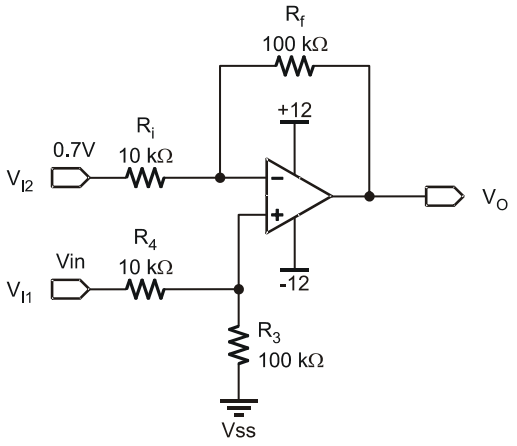


Figure 6-11
Offset and Gain using Differential Configuration

DO NOT BUILD

6

How would the 0.7 V be set? One way would be to use a potentiometer that is adjusted to 0.7 V as the input for V_{I2}. In both configurations we are setting the offset and span for the voltage range of interest.

Fortunately, the ADC0831 converter can directly perform the spanning and offsetting. Consider the pin-out of this device as shown in Figure 6-12.

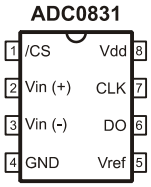


Figure 6-12
ADC0831 Pin Out

Up until now Vin(-) has been connected to Vss (0 V) and V_{ref} has been connected to Vdd (5 V). These two inputs set the offset and span voltages over which to convert. If the ADC0831's Vin(-) is set to 0.7 V and Vref is set to 0.5 V, the ADC will convert the range of 0.7 V to 1.2 V to a byte value of 0 to 255 respectively.

One method of using potentiometers to set these two pins is shown in Figure 6-13.

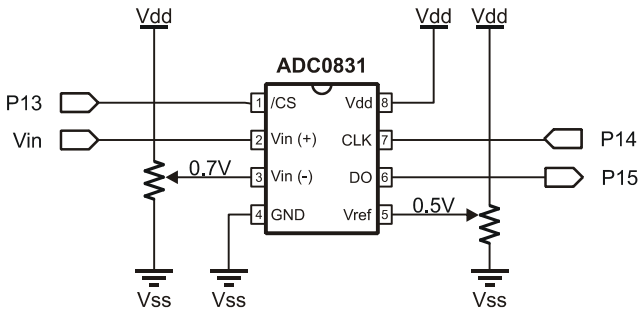


Figure 6-13
Span and Offset of ADC0831 Using Potentiometers

DO NOT BUILD

Instead of using potentiometers, which must be manually adjusted for different ranges, can you think of a simple way to have the BASIC Stamp set the offset and span voltage of the ADC directly?

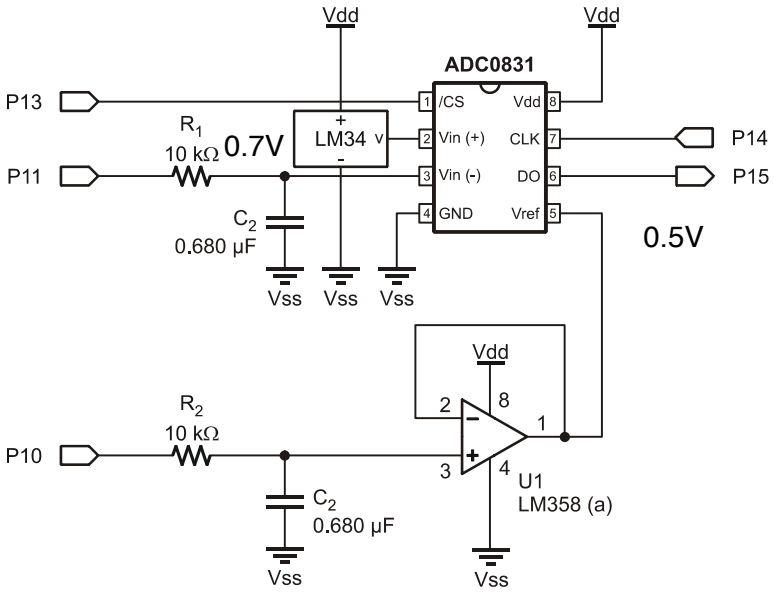
Recall how filtered PWM was used in Chapter 5 to set the voltage of the fan. The same principle may be applied here as shown in Figure 6-14. Due to the low impedance of the Vref input, the filtered PWM must be buffered prior to being applied. The ADC0831's Vin(-) is higher impedance and does not require buffering.

Additional Parts Required:

(1) LM358 Op-Amp
(2) 0.68 µF Capacitors
(2) 10 kΩ Resistors

- √ Construct the circuit in Figure 6-14.
- √ In StampPlot, set "ADC Span" to 0.5 and "ADC Offset" to 0.7.
- √ Connect and plot.
- √ Heat the LM34 again.
- √ Click the "Scale Plot 70-120F" button on StampPlot to cover the selected range.

Figure 6-14 Span and Offset of ADC Using PWM - *Schematic*



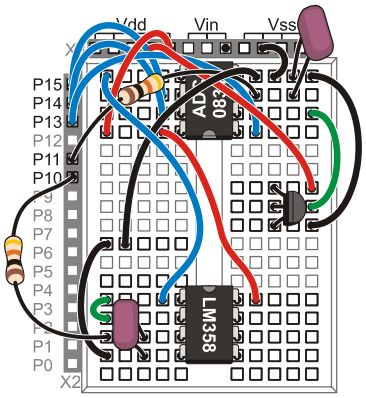


Figure 6-15
Span and Offset of ADC Using PWM

Wiring Diagram

6

Note the new resolution as shown in Figure 6-16 compared to previous tests shown in Figure 6-4. Gaps between points are temperatures that were not sampled. Note that temperature resolves to .196 °F/bit by observing the change in voltage.

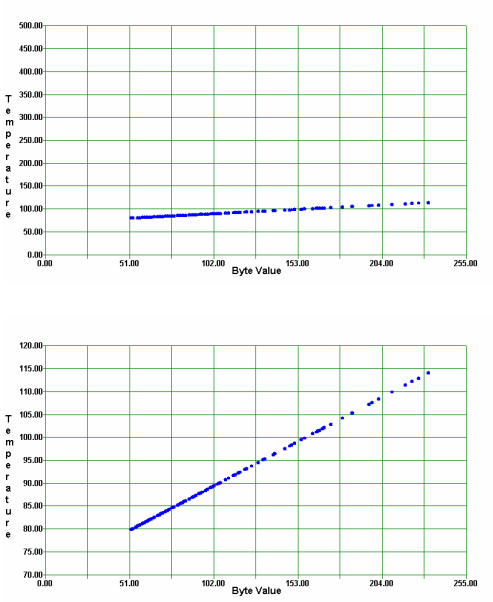


Figure 6-16
ADC Measurement
0 to 500 °F and
70 to 120 °F

The PWM style of setting offset and span may not be 100% accurate but is adequate for our testing. Offset values below 0.5 V should be avoided for better accuracy. Combined offset and span voltages used to set the upper voltage limit (Vspan + Voffset) should be 3.5 V or less since the op-amp is only powered from 5 V.

Program Discussion

The **`ReadSP`** subroutine reads the values of offset and span from StampPlot. These values are used in calculations and in setting the ADC. Since the BASIC Stamp does not use floating-point math (values with decimal points), the values are multiplied by 10 so that 0.5 is stored as 5, or tenths of a volt.

```
DEBUG CR,"!READ [(txtADCoffset),*,10]",CR
DEBUGIN DEC V_Offset
```

The **SetADC** subroutine uses the span and offset values to control the Vref and Vin(-) pins respectively. The settings are scaled accordingly to apply 0-5 V to these pins using PWM. Note that it is again using our transfer function line equation, y = mx + b where b= 0, though slightly rearranged.

```
PWM ADC_Vminus, V_Offset * 255/50,100
```

If 255 were divided by 50 first, the slope would have been 5 and not 5.1. By multiplying first, while ensuring 65535 is not exceeded, we have better resolution and accuracy.

PWM value = (voltage desired in 10ths) x (byte value span)/(max. voltage in 10ths)
PWM value = (voltage in tenths) x 255/50

The **ReadADC** subroutine reads the ADC and stores the byte collected in **ADC_ByteValue**. After enabling the IC using the Chip Select (CS) pin, the data from the ADC is shifted in using a clock pulse on the clock (CLK) line and data is collected from the Data Out (DO) pin. The **\9** means that 9 bits are collected, though the first is not used and is discarded.

```
LOW ADC_CS
SHIFTIN ADC_Dout, ADC_Clk, MSBPOST,[ADC_ByteValue\9]
HIGH ADC_CS
```

In the **CalcTemp** section, the values of **V_Span** and **V_Offset** are multiplied by 1000. Working with the original ADC values, a span of 0.5 would have related to 50 °F. If we used 50 in our equation for 50/255 this would equal 0.196, but since the BASIC Stamp does not perform floating-point math, this would be 0! By multiplying by 1000, the result is 196 providing a reading of 8007 for a temperature of 80.07. Again, a precaution is that 65535 is not exceeded for any intermediary calculation, such as with a span of 5.0 V.

```
TempF = (V_Span * 1000)/255 * ADC_ByteValue + (V_Offset * 1000)
```

The **UpdateSP** subroutine updates the **txtByte** and **txtByteBin** text boxes with the byte value in decimal and binary respectively, as read from the ADC0831, and also updates **txtTemp** with the current temperature divided by 100.

```
DEBUG "!O txtByteBin = ", BIN8 ADC_ByteValue,CR,
      "!O txtByte = ", DEC ADC_ByteValue,CR

DEBUG "!O txtTemp = [", DEC TempF,",/,100]",CR
```

In the **PlotPoint** subroutine, StampPlot's Filled Circle (**!FCIR**) instruction is used to plot a point in white based on the X coordinate of **ADC_ByteValue** and the Y coordinate of the temperature with a size of 0.3 absolute. After a 100 ms pause, the point is plotted using the same parameters in blue to give the effect of a blinking point.

```
DEBUG "!FCIR (txtByte),(txtTemp),0.3A,(WHITE)",CR
PAUSE 100
DEBUG "!FCIR ,,,(BLUE)",CR
```

Challenge 6-2: Spanning and Scaling for an Air Conditioner System

Instead of eventually controlling an incubator, assume the system under control is an air conditioning system for an equipment room. The temperature needs to be monitored and controlled over a range of 50 °F to 90 °F.

1. What values of **V_Span** and **V_Offset** would be appropriate?
2. What would be the resolution in degrees Fahrenheit for this range of temperature?
3. Draw a graph of byte value vs. temperature.
4. What would be the equation for this line?
5. At 72 °F, what would be:
    a. The output of the LM34?
    b. The byte value of the ADC0831?

ACTIVITY #3: MANUAL CONTROL OF INCUBATOR

Have you ever ridden in a vehicle with someone who controls the cabin temperature by cycling the heater or air conditioner on and off? Too hot – turn on the AC. Too cold – turn off the AC. Every several miles they switch again to keep the cabin comfortable. Manual control of a system with a small allowable range can be time consuming (and a little frustrating to other passengers).

In this activity you will construct the incubator circuit used in the remainder of this text. We will begin by manually controlling the incubator and testing the response. Our incubator is a closed system containing a resistor acting as a heater and the LM34 for temperature monitoring.