LAB 4
Digital-to-Analog and Analog-to-Digital Conversion

Suggested Reading

Basic Analog and Digital, Parallax Inc. Especially chapters 3, 4, 5, 7
What is a Microcontroller?, Parallax Inc., Chapter 8.
Data sheets for the following integrated circuits:
ADC0831, Analog-to-digital converter
LM358, Operational amplifier
LM35, Temperature sensor

1. Pulse Width Modulation: aka DC by AC

The 16 I/O pins on the Stamp are all digital 0 or 5 V only. However, a "cheat" allows one to simulate variable voltages between these levels which is adequate for many applications. By outputting a rectangular wave alternating between 0 and 5 V the average value can be changed by varying the relative length of time the output is at 5 V and at 0 V.

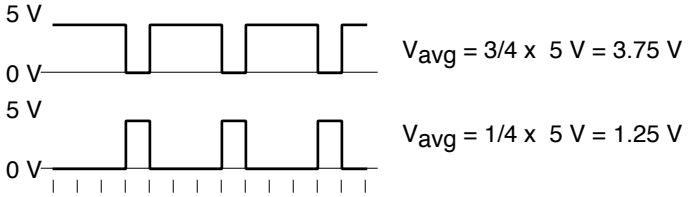

Fig. 1: Two pulse-width modulated signals giving a higher and lower average voltage.

PBASIC has two ways to do this. The PULSOUT command simply raises the level on one of the pins for a short interval. (There's no E in PULSOUT.) Its syntax is

```
PULSOUT pin#, n
```

where n determines the time that the output is high. On the Stamp 2 the time is measured in intervals of 2 μs so that the pulse width is n x 2 μs. If the PAUSE instruction follows this then one cycle of the pulse width modulated signal is complete. The PAUSE instruction measures time in 1 ms intervals on the Stamp 2. Other models of the Stamp may have different time intervals.

For example the following loop puts out a pulse 1 ms high and 3 ms low on pin 1:

```
DO
      PULSOUT 1, 500
      PAUSE 3
LOOP
```

This should correspond to the second waveform in Fig. 1.

> Try this example and observe the time intervals with a scope. Use the DMM to measure the average voltage and compare to what is predicted. Then modify the code to produce the higher voltage level in Fig. 1.

Some devices, such as DC motors, may be able to use PWM signals as is. Otherwise it needs to be filtered to a relatively ripple-free DC signal. A passive low-pass RC filter can be used with an RC time constant about 10 times longer than the cycle's period. That would be 40 ms in this case. Calculate the C value needed to be used assuming that our standard 220 Ω current-limiting resistor is attached to the output pin. If the capacitor value is large, then the capacitor may be electrolytic, Watch that you connect the polarity correctly.

Measure the signal produced by filtering the PWM signal output. Can you measure the ripple? Is it what one would expect?

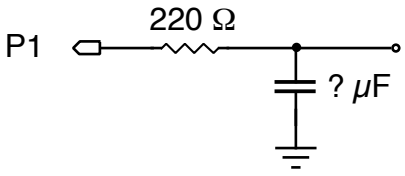

Fig 2: Use a low-pass filter whose RC time constant is 10 times the cycle length to make a relatively ripple-free DC output from a PWM signal.

There's a PWM command that outputs a PWM signal with a much faster cycle time. It's syntax is

```
PWM pin#, duty, m
```

where pin# is the pin number. Duty is the duty cycle indicated by numbers from 0 to 255 for duty cycles from 0% to 100%. (For example a 50% duty cycle would have duty=128). The number m is the length of the pulse in ms. The maximum length is only 255 ms which means that if you want a signal longer than one-quarter second, you would have to put this in a loop.

You could use a smaller filter capacitor with PWM because of its shorter cycle period. But, If it's in a loop, there would be a pause after each repeat and that will introduce ripples in the signal.

> Explore using the PWM command and see if you can filter it with a small capacitor and then see the effect of looping on the output.

2. Making PWM sine waves: AC from DC

If one needs a sine wave output there's a variation of the PWM command, FREQOUT:

```
FREQOUT pin#, m, f1, f2
```

m is the duration in ms. The frequency of the output is f1 and a second frequency f2 can be optionally added. FREQOUT puts out a PWM signal with a varying duty cycle so that when filtered a sine wave results. There's naturally an offset voltage but that usually doesn't matter if you want to hear noise on a speaker.

A small piezoelectric speaker element (not a buzzer!) can play the tones thus generated and annoy the hell out of your classmates. These little speakers aren't very good at bass response, so try several octaves above middle C. Using 440 Hz as the first A above middle C, The A two octaves higher would be 1760 and the next higher A, 3520 Hz.
The well-tempered diatonic scale consists of 12 intervals, each successively higher note in the scale has a frequency that is $2^{1/12}$ times the previous. So $A^{\#}$ is 1760 x 1.0595 = 1964.7. You'll have to round this to the nearest integer 1965 to use it in the FREQOUT command.

> Calculate the musical scale, and practice a few scales and then play a little tune on your stamp. [Be aware that PBASIC is integer only and does not do floating point arithmetic, so you can't do the exponentiation on the fly in the stamp.]

3. Four-bit parallel Digital-to-Analog Conversion using the R-2R Ladder

The R-2R ladder shown in the following figure allows direct conversion of a 4-bit output to an analog voltage proportional to the binary value of the output. All resistors are identical so that the resistors paired

in parallel have half the resistance of the single ones. 2 kΩ or 2.2 kΩ resistors provide a convenient input impedance that doesn't load the output pins too much.

Construct the circuit on your breadboard. To keep it neat you should trim the leads on the resistors so that they fit snugly against the board when spanning several empty holes or stand up so that the leads can fit in adjacent holes.

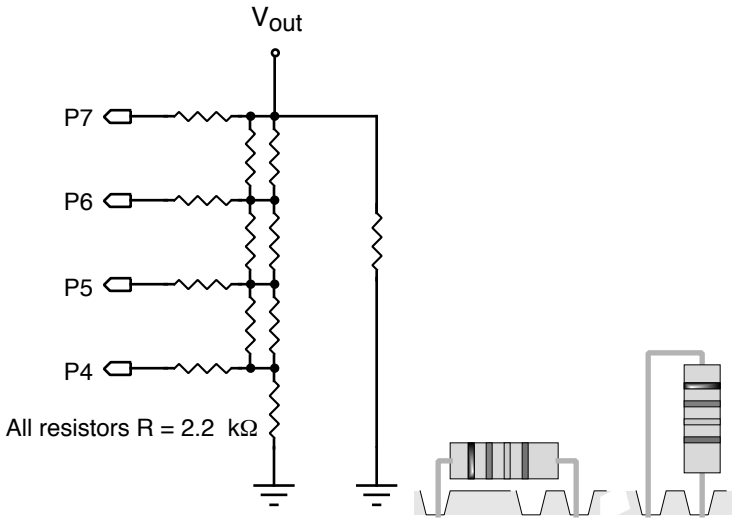

Fig 3: The R-2R ladder creates an output voltage proportional to the input nibble on pins 4-7. Neat bread-boarding technique can save time and space.

Measure the output voltage at Vout for 0001, 0010, 0100 and 1000. Analyse the circuit using Thévenin's theorem and verify that the output is consistent with theory. The principle of superposition allows you to predict the output when two or more bits are high. Write a program to count up from 0000 to 1111 repetitively. Output each nibble to port B (pins 3-7). Monitor the Vout on the scope and display the stair-step ramp-up of voltage by adjusting the time scale and triggering on the downward edge produced at the rollover from 1111 to 0000.

Find the output impedance of the R-2R ladder by connecting another 2 kΩ resistor between Vout and ground. From the decrease in the maximum output voltage (when the nibble is 1111) you can calculate the effective output impedance. You can also figure it out theoretically by using the Thévenin analysis. (I.e, imagine all voltage sources are shorts to ground and calculate the equivalent resistance to ground.)

An obvious improvement would be to decrease the output impedance so that the voltage doesn't change significantly when a significant load is connected to Vout. This can be accomplished by using an op-amp in the voltage follower configuration as shown in the next figure.

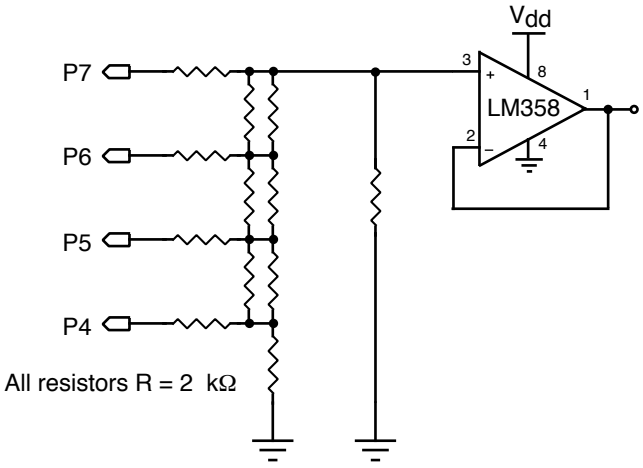

Fig. 4: A voltage follower can decrease the output impedance

Using the voltage follower, verify the low output impedance by connecting a 2kΩ (or lower resistance) load to the op-amp's output.

It might be desirable to obtain 5V output when the nibble is 1111. Because the R-2R ladder provides less than 5 V at maximum output, the follower circuit only reproduces that voltage. By adjusting the feedback ratio an amplification can be obtained but there's another problem. The op amp saturates at 1.5 V less than its supply voltage when is 5 V. Therefore only a small amount of amplification can be obtained before saturation occurs at 3.5 V. There is the possibility of using Vin instead of Vdd for the op-amp supply. Verify from the LM358 specifications that Vin is indeed less than the maximum allowed for the op amp. If it is ok, then connect pin 8 to Vin.

After connecting the supply to Vin, adjust the feedback ratio so that the output is 5 V when the nibble is 1111 and check to see if it works. Vin is not regulated, but small variations in Vin won't affect the output voltage because the negative feedback on the op amp assures that the output voltage is determined by its input only, at least in principle.

4. Analog-to-Digital Conversion

Remove the negative feedback from the op amp and connect the inverting input to an externally supplied steady voltage between 0 and 5 V. The op-amp is now functioning as a comparator and can signal whether the digital-to-analog output is greater than or less than the external voltage. If one programs the micro-controller to vary its D/A output in a systematic way it can determine the value of the external voltage within an error equal to its resolution: ±1/2 a voltage step. The following circuit illustrates the principle

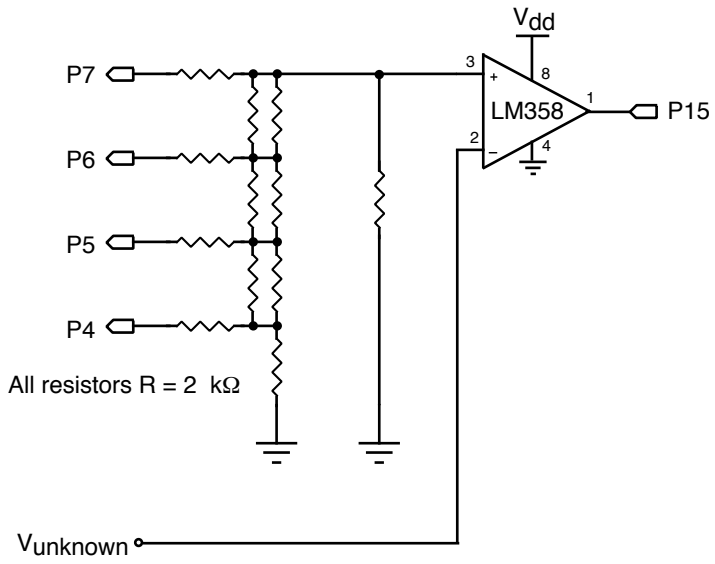

Fig. 5: Using the LM358 as a comparator the output of which is read by the micro-controller allows analog-to-digital conversion if the micro-controller also controls the digital-to-analog converter.

The simplest method of searching for the unknown voltage value would be to start from zero volts and step up incrementally until the comparator output changes to 1. The external voltage, $V_{unknown}$, is somewhere between the last voltage for which the comparator output is zero, and the voltage where it just turned to one.

```
' Using the output of a 4-bit R2R ladder
'  and an op-amp as comparator
' Do A/D conversion with a ramp-up algorithm
' {$STAMP BS2}
' {$PBASIC 2.5}

Delta_V CON 205          ' Voltage step
Delta_Vo2 CON 102 ' one half of the voltage step


n var Nib    ' n   is the output nibble,

DirB=%1111          'All 4 bits on port B, pins 4-7 are output
INPUT 15            'Pin 15 is used to signal too high or too low "guess"
                    ' 1 on pin 15 indicates the "guess" is too high
                    ' 0 on pin 15 indicates the "guess" is too low

DEBUG CLS    ' Clear the screen of detritus

DO

      FOR n = 0 to 15

            OutB=n
            IF (IN15=1) Then
                  IF (n=0) Then
                        DEBUG CLS,"V= ", DEC 0," +/- ",DEC 103," mV",CR
                  ELSE
                        DEBUG CLS,"V= ",dec (n*Delta_V)-Delta_Vo2," +/- 103
```

```
mV",CR
                    ENDIF
                    EXIT
               ENDIF
        NEXT

        loop
END
```

Successive Approximation

A more efficient algorithm would be to start searching in the middle of the voltage range. In terms of the output nibble that means to start with 1000 and see if the voltage produced is too high or too low. If we have adjusted 1111 to give 5 V then 1000 is (8/15)x5V or 2.667. For example if the unknown voltage were 3.1 volts the process would be as follows

Try 1000: 2.6667 V --- comparator says too low, leave bit 3 high
Try 1100: 4.0000 V --- comparator says too high, reset bit 2 low
Try 1010: 3.3333 V --- comparator says too high, reset bit 1 low
Try 1001: 3.0000 V --- comparator says too low, leave bit 0 high

This process could continue if we had more bits in the D/A.

With only four bits, the result of the conversion would be 1001 but to be honest the value is between 3.00 and 3.33 V and one should quote the result as 3.17±0.17 V which is consistent with the actual voltage 3.1 V.

This process, known as successive approximation, can be characterized as successively finding the value of each bit, starting with the most significant. At each stage the comparator output determines whether the bit in question is 1 or 0.

As an exercise, write a routine to achieve successive approximation analog-to-digital conversion using the circuit of fig. 5. On average, is the conversion time faster for successive approximation than for the ramp-up routine? How much faster?

5. The ADC0831 Analog-to-Digital Converter Chip

There are many single-chip successive approximation A/D chips. Among them is the ADC0831. The ADC0831 is an eight-bit converter which accepts an analog input voltage between 0 and 5 V. If the entire 5 V span is digitized with eight-bit resolution then the resolution is 5/256 or about 20 mV per bin. Greater resolution can be achieved by limiting the input range. Putting a voltage on Vin– sets the lowest voltage of the input range, and the voltage on Vref is the highest voltage. For example, if one wanted a 10 mV resolution between 1 V and 3.5 V, one would connect Vin– to 1 V and Vref to 3.5 V.

> The following circuit allows you to test the converter. Connect the circuit and run the following program to convert the input voltage. There is a small defect in the display subroutine which causes the binary input value to be incorrectly converted to decimal. This comes about because all arithmetic in PBASIC is integer and the division operator (/) only return the integer part of the answer and the remainder operation (//) returns the remainder that is subsequently converted to the decimal part of the answer. If the defect annoys you see if you can find a fix for it.

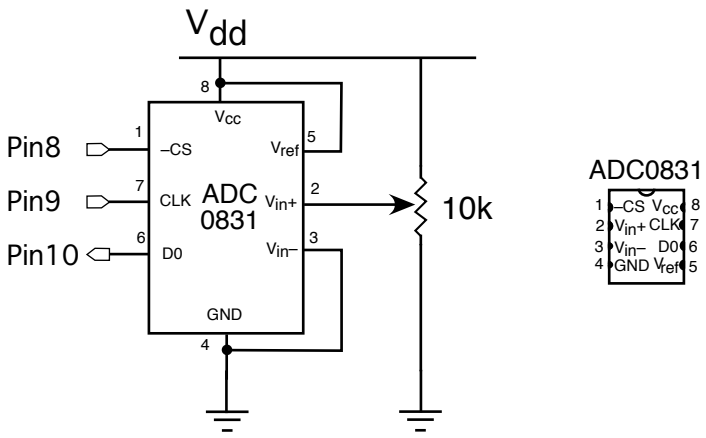

Fig 6: The ADC0831 analog-to-digital converter connected to a variable resistor in order to test its operation. The pin diagram is shown at right.

[NOTE: The following program was tried and used on an ADC0834. It needs modification for use with the ADC0831.]

```
' Read a value from the ADC034
' {$STAMP BS2}
' {$PBASIC 2.5}

CS PIN 8
D1 PIN 9
CLK PIN 10
D0 Pin 11


adcBits VAR WORD
v     VAR word
adcMode Var Nib

'OUTPUT D1
adcMode = %0100

DEBUG CLS

DO
      GOSUB ADC_Data
      GOSUB Display
LOOP

END


ADC_Data:
      HIGH CS
      LOW CS
      LOW CLK
      PULSOUT CLK, 210
      SHIFTOUT D1, CLK, MSBFIRST, [adcMode\4]
      SHIFTIN D0, CLK, MSBPOST, [adcBits\9]
RETURN

Display:
```

```
      DEBUG DEC adcBits, CR
      v = adcBits
      DEBUG "V = ", DEC v/51 ,".",DEC 196*(v//51),CR
RETURN
```

6. Temperature Measurements with the LM35

An interesting application for the ADC is to digitize the output of a temperature sensor. One easy-to-use temperature sensor is the LM35. The LM35 puts out a voltage proportional to temperature. There are several models of this, but the most common one measures temperatures between 0 and 100°C by producing an output voltage 10 mV times the temperature in degrees celcius. Thus a temperature of 20°C would be 20 mV. The device has three terminals 5V, ground and output as shown in the diagram.

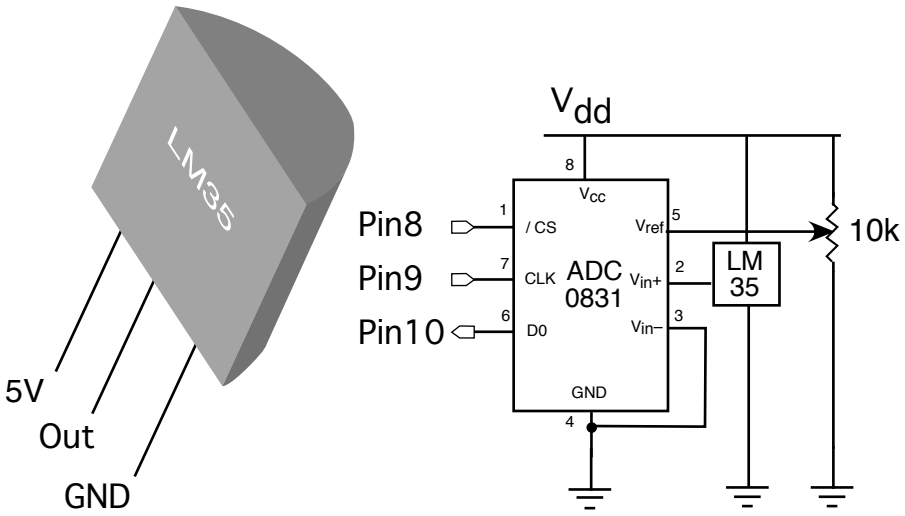

Fig 7: The LM35 temperature sensor is a three-terminal device. The output acts like a voltage source whose voltage is proportional to temperature in degrees celsius: V_{out} - (10 mV/°C)T. The circuit shows how to connect to an A/D converter with a potentiometer to set the upper limit of the ADC range.

Connect the LM35 as shown in Fig. 7. Adjust V_{ref} to be 0.51 V and modify the display subroutine to print out the temperature in degrees C.

7. A Temperature Logger

It's sometimes useful to have a small device that takes data and stores it for later retrieval. It's easy to turn your stamp into a temperature logger which can be placed somewhere to record temperatures at a predetermined interval. The micro-controller doesn't need to be attached to any other computer or source of power other than its 9V battery. Later it can be brought back to the lab and the data downloaded into another computer for analysis and long-term storage. The following program uses the WRITE instruction to put data in permanent EEPROM. It records temperatures every 10 s for 1000 s (16.7 min) and then stops. Another program must be used to get the data, but downloading the retrieval program does not disturb the recorded data.

[NOTE: The following program was tried and used on and ADC0834. It needs modification for use with the ADC0831.]

```
' Read 100 Temps from the LM 35 through the ADC034 every 10 s and store in
EEPROM
```

```
' Adjust Vref to 0.51 so that the temperature is the adcBits/5
' This gives a maximum of 51°C which isn't too unrealistics for room temps.
' {$STAMP BS2}
' {$PBASIC 2.5}

CS PIN 8
D1 PIN 9
CLK PIN 10
D0 Pin 11


adcBits VAR WORD
v     VAR word
adcMode Var Nib
LOC VAR BYTE 'Temperature data storage location

LOC_Start   CON 0
NumberOfPoints CON 100
TimeInterval CON 10     'in seconds


adcMode = %1000

DEBUG CLS
LOC = LOC_Start
DO
      GOSUB ADC_Data
      WRITE LOC, adcBits
      LOC=LOC+1
      sleep TimeInterval            ' sleep does not consume much power
LOOP while LOC<NumberOfPoints

END


ADC_Data:
      HIGH CS
      LOW CS
      LOW CLK
      PULSOUT CLK, 210
      SHIFTOUT D1, CLK, MSBFIRST, [adcMode\4]

      SHIFTIN D0, CLK, MSBPOST, [adcBits\8]
RETURN




' Read 100 data points stored in the EEPROM in locations 0 to 99
' Print them out in tab-delimited format.
' {$STAMP BS2}
' {$PBASIC 2.5}

LOC VAR BYTE 'Data storage location

LOC_Start   CON 0
NumberOfPoints CON 100
```

```
DEBUG CLS
LOC = LOC_Start
DO
'     GOSUB ADC_Data
      READ LOC, adcBits
      GOSUB Display
      LOC=LOC+1
LOOP while LOC<NumberOfPoints

END

Display:
      DEBUG DEC LOC,9,DEC adcBits,CR 'writes LOC tab DATA
RETURN
```

The data will be printed in the debug output screen. You can copy and paste it into a spreadsheet or other text document as you wish. The two columns represent the data number and the actual data. There's a tab between the columns which most spreadsheet programs accept as a column separator. It's also easy to format in a table.

Mini-Project Ideas:
There are obvious improvements that could be made to the temperature logger. One improvement would be to put an LED on the device to signal when data are being taken or to signal when all the data logging is completed and they are ready for download. If you do include an LED in the device, think how to do it so that the power-saving feature of the SLEEP command is not defeated.

Consider fitting an input switch (4-bit nibble switch or rotary hexadecimal encoder) whose value would change the operation of the logger:
a) select different time intervals
b) choose more or fewer data points
c) switch between data collection and data retrieval modes in the same program.

Your own ideas:
_____
_____
_____