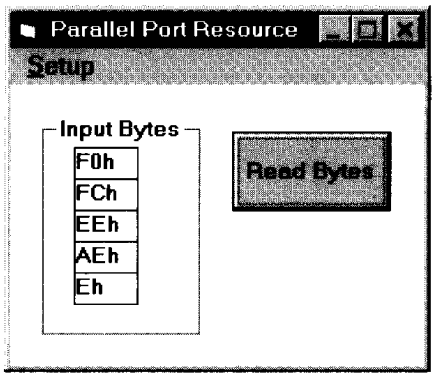

Figure 8-4: Screen for Listing 8-3's program.

Using the Data Port for Input

If you have a bidirectional data port, you can use the eight data lines as inputs. You can also use the port as an I/O port, both reading and writing to it, as long as you're careful to configure the port as input whenever outputs are connected and enabled at the data pins. In other words, when the data lines are configured as outputs, be sure to tristate, or disable, any external outputs they connect to. You can use a '374 to latch input at the Data port, as in the previous examples.

Reading Analog Signals

The parallel port is a digital interface, but you can use it to read analog signals, such as sensor outputs.

Sensor Basics

A sensor is a device that reacts to changes in a physical property or condition such as light, temperature, or pressure. Many sensors react by changing in resistance. If a voltage is applied across the sensor, the changing resistance will cause a change in the voltage across the sensor. An analog-to-digital converter (ADC) can convert the voltage to a digital value that a computer can store, display, and perform calculations on.

Simple On/Off Measurements

Sometimes all you need to detect is the presence or absence of the sensed property. Some simple sensors act like switches, with a low resistance in the presence

```
'Clock is Control bit 3.
Const Clock% = 8
'Write 1 to bits C0-C2 to allow their use as inputs.
Const SetControlBitsAsInputs% = 7
```

```
Sub cmdReadByte_Click ()
Dim LowBits%
Dim HighBits%
Dim ByteIn%
'Latch the data.
ControlPortWrite BaseAddress, SetControlBitsAsInputs + Clock
ControlPortWrite BaseAddress, SetControlBitsAsInputs
'Read the bits at C0-C2, S3-S7.
LowBits = ControlPortRead(BaseAddress) And 7
HighBits = StatusPortRead(BaseAddress) And &HF8
ByteIn = LowBits + HighBits
lblByteIn.Caption = Hex$(ByteIn) + "h"
End Sub
```

```
Sub Form_Load ()
'(partial listing)
'Initialize the Control port.
ControlPortWrite BaseAddress, SetControlBitsAsInputs
End Sub
```

Listing 8-2: Reading 8 bits using the Status and Control ports.

of the sensed property, and a high resistance in its absence. In this case, you can connect the sensor much like a manual switch, and read its state at an input bit. Sensors that you can use this way include magnetic proximity sensors, vibration sensors, and tilt switches.

Level Detecting

Another common use for sensors is to detect a specific level, or intensity, of a property. For this, you can use a comparator, a type of operational amplifier (op amp) that brings its output high or low depending on which of two inputs is greater.

Figure 8-5 shows how to use a comparator to detect a specific light level on a photocell. The circuit uses an LM339, a general-purpose quad comparator. The resistance of a Cadmium-sulfide (CdS) photocell varies with the intensity of light on it. Pin 4 is a reference voltage, and pin 5 is the input being sensed. When the sensed

input is lower than the reference, the comparator's output is low. When the sensed input is higher than the reference, the comparator's output is high.

As the light intensity on the photocell increases, the photocell's resistance decreases and pin 5's voltage rises. To detect a specific light level, adjust *R2* so that *Vout* switches from low to high when the light reaches the desired intensity. You can read the logic state of *Vout* at any input bit on the parallel port.

R4 is a pull-up resistor for the LM339's open-collector output. *R3* adds a small amount of hysteresis, which keeps the output from oscillating when the input is near the switching voltage.

You can use the same basic circuit with other sensors that vary in resistance. Replace the photocell with your sensor, and adjust *R2* for the switching level you want.

Reading an Analog-to-digital Converter

When you need to know the precise value of a sensor's output, an analog-to-digital converter (ADC) will do the job. Figure 8-6 is a circuit that enables you to read eight analog voltages. The ADC0809 converter is inexpensive, widely available, and easy to interface to the parallel port. The ADC0808 is the same chip with higher accuracy, and you may use it instead.

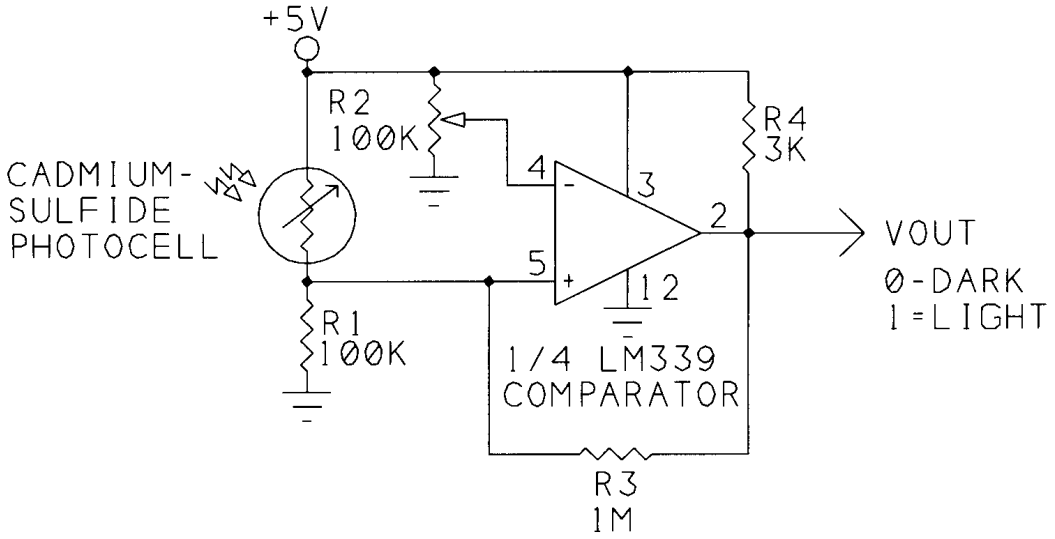

Figure 8-5: A comparator can detect a specific voltage.

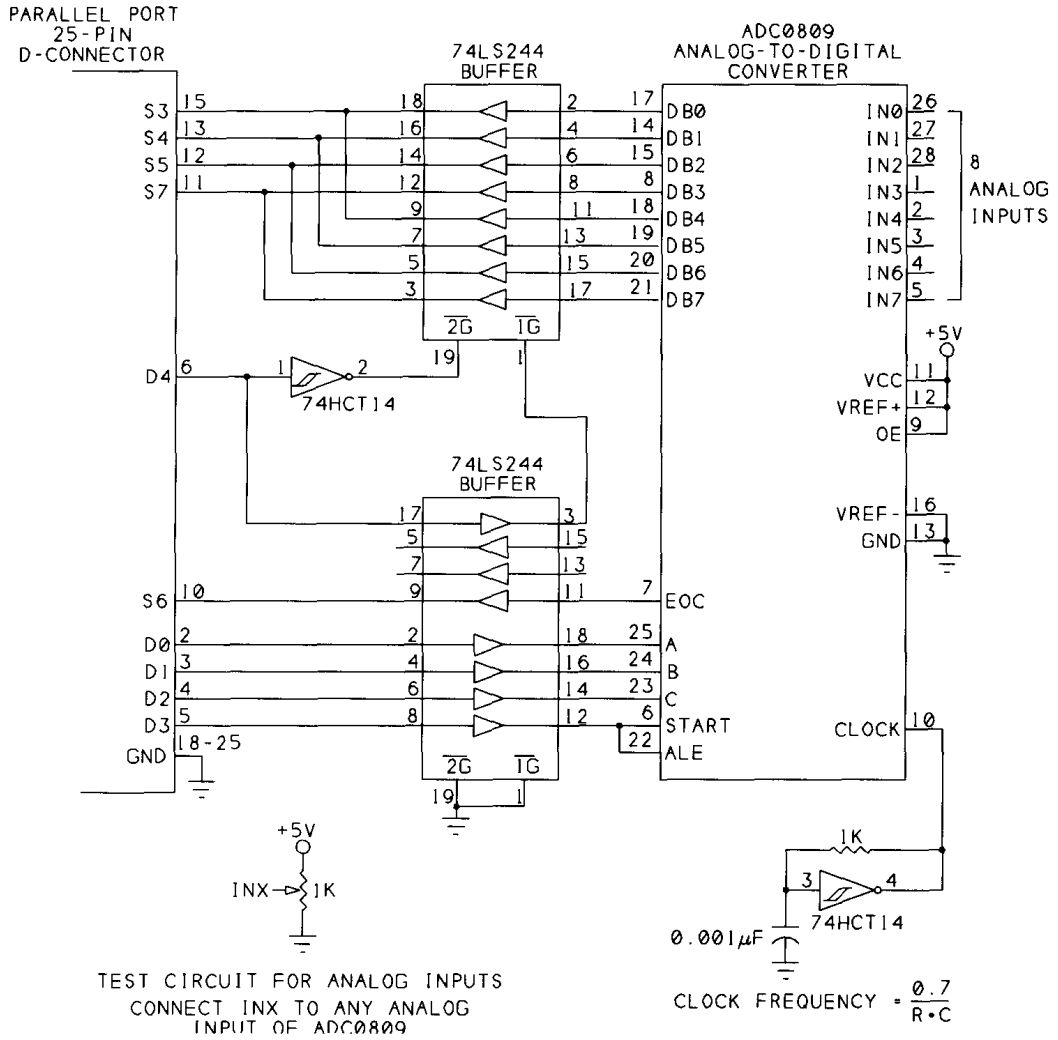

Figure 8-6: The ADC0809 analog-to-digital converter provides a simple way to read 8 analog channels at the parallel port.

The ADC0809 has eight analog inputs (*IN0-IN7*), which may range from 0 to +5V. To read the value of an analog input, you select a channel by writing a value from 0 to 7 to inputs *A-C*, then bringing *Start* and *Ale* high, then low, to begin the conversion. When the conversion is complete, *Eoc* goes high and the digital outputs hold a value that represents the analog voltage read.

The chip requires a clock signal to control the conversion. A 74HCT14 Schmitt-trigger inverter offers a simple way to create the clock. The frequency can range from 10 kilohertz to 1280 kilohertz. If you prefer, you can use a 555 timer for the clock, although the maximum frequency of the 555 is 500 kilohertz. Conversion time for the ADC is 100 microseconds with a 640-kilohertz clock.

```
Dim DataIn%(0 To 7)
Dim DataByte%(0 To 4)
```

```
Sub cmdReadBytes_Click ()
Dim BitNumber%
'The Control port selects a bit number to read.
'The Status port holds the data to be read.
For BitNumber = 0 To 7
    ControlPortWrite BaseAddress, BitNumber
    DataIn(BitNumber) = StatusPortRead(BaseAddress)
Next BitNumber
GetBytesFromDataIn
DisplayResults
End Sub
```

```
Sub DisplayResults ()
Dim ByteNumber%
For ByteNumber = 0 To 4
lblByteIn(ByteNumber).Caption = Hex$(DataByte(ByteNumber)) & "h"
Next ByteNumber
End Sub
```

Listing 8-3: Reading 40 inputs. (Sheet 1 of 2)

```
Sub GetBytesFromDataIn ()
'Bits 3-7 of the 8 bytes contain data.
'To make 5 data bytes from these bits,
'each data byte contains one bit from each byte read.
'For example, data byte 0 contains 8 "bit 3s,"
'one from each byte read.
Dim ByteNumber%
Dim BitNumber%
Dim BitToAdd%
For ByteNumber = 0 To 4
    DataByte(ByteNumber) = 0
    'BitRead gets the selected bit value (ByteNumber + 3)
    'from the selected byte read (DataIn(BitNumber)).
    'To get the bit value for the created data byte,
    'multiply times 2^BitNumber.
    'Add each bit value to the created byte.
    For BitNumber = 0 To 7
        BitToAdd = (BitRead(DataIn(BitNumber), ByteNumber + 3)) _
         * 2 ^ BitNumber
        DataByte(ByteNumber) = DataByte(ByteNumber) + BitToAdd
    Next BitNumber
Next ByteNumber
End Sub
```

Listing 8-3: Reading 40 inputs. (Sheet 2 of 2)

Inputs *Vref+* and *Vref-* are references for the analog inputs. When an analog input equals *Vref-*, the digital output is zero. When the input equals *Vref+*, the digital output is 255. You can connect the reference inputs to the +5V supply and ground, or if you need a more stable reference or a narrower range, you can connect other voltage sources to the references.

Listing 8-4 reads all eight channels and displays the results. It reads the data in two nibbles at *S3–S5* and $\overline{S7}$. Outputs *D0–D2* select the channel to convert, *D3* starts the conversion, and *D4* selects the nibble to read. Optional input *S6* allows you to monitor the state of the ADC's end-of-conversion (*Eoc*) output.

A 74LS244 drives the Status bits. When *D4* is low, you can read the ADC's *DB0–DB3* outputs at the Status port. When *D4* is high, you can read *DB4–DB7*.

A second 74LS244 interfaces the other signals to the ADC. Bringing *D3* high latches the channel address from *D0–D2*, and bringing *D3* low starts a conversion.

Bit *S6* goes high when the ADC has completed its conversion. You can monitor *S6* for a logic high that signals that the conversion is complete, or you can use the

```
Const Start% = 8
Const HighNibbleSelect% = &H10
Dim DataIn%(0 To 7)
Dim ChannelNumber%
Dim LowNibble%
Dim HighNibble%
```

```
Sub cmdReadPorts_Click ()
Dim EOC%
For ChannelNumber = 0 To 7
    'Select the channel.
    DataPortWrite BaseAddress, ChannelNumber
    'Pulse Start to begin a conversion.
    DataPortWrite BaseAddress, ChannelNumber + Start
    DataPortWrite BaseAddress, ChannelNumber
    'Wait for EOC
    Do
        DoEvents
        LowNibble = StatusPortRead(BaseAddress)
        EOC = BitRead(LowNibble, 6)
    Loop Until EOC = 1
    'Read the byte in 2 nibbles.
    DataPortWrite BaseAddress, ChannelNumber + HighNibbleSelect
    HighNibble = StatusPortRead(BaseAddress)
    DataIn(ChannelNumber) = MakeByteFromNibbles()
Next ChannelNumber
DisplayResult
End Sub
```

```
Sub DisplayResult ()
For ChannelNumber = 0 To 7
    lblADC(ChannelNumber).Caption = _
    Hex$(DataIn(ChannelNumber)) & "h"
Next ChannelNumber
End Sub
```

Listing 8-4: Reading 8 channels from an ADC. (Sheet 1 of 2)

```
Function MakeByteFromNibbles% ()
Dim S0%, S1%, S2%, S3%, S4%, S5%, S6%, S7%
S0 = (LowNibble And 8) \ 8
S1 = (LowNibble And &H10) \ 8
S2 = (LowNibble And &H20) \ 8
S3 = (LowNibble And &H80) \ &H10
S4 = (HighNibble And 8) * 2
S5 = (HighNibble And &H10) * 2
S6 = (HighNibble And &H20) * 2
S7 = HighNibble And &H80
MakeByteFromNibbles = S0 + S1 + S2 + S3 + S4 + S5 + S6 + S7
End Function
```

Listing 8-4: Reading 8 channels from an ADC. (Sheet 2 of 2)

rising edge at *S6* to trigger an interrupt, or you can ignore *S6* and just be sure to wait long enough for the conversion to complete before reading the result.

The circuit uses *S6* as end-of-convert because it's the parallel port's interrupt pin. If you don't use interrupts, you can wire the ADC's data outputs to S4–*S7* for an easier (and faster) conversion from nibbles to byte.

At each analog input, you can connect any component whose outputs ranges from 0 to +5V.

Sensor Interfaces

If the output range of your sensor voltages is much less than 5V, you can increase the resolution of the conversions by adjusting the reference voltages to a range that is slightly wider than the range you want to measure.

To illustrate, consider a sensor whose output ranges from 0 to 0.5V. The 8-bit output of the converter represents a number from 0 to 255. If *Vref+ is* 5V and *Vref-* is 0V, each count equals 5/255, or 19.6 millivolts. A 0.2V analog input results in a count of 10, while a 0.5V input results in a count of 26. If your input goes no higher than 0.5V, your count will never go higher than 26, and the measured values will be accurate only to within 20 millivolts, or 1/255 of full-scale.

If you lower *Vref+* to 0.5V, each count now equals 0.5/255, or 0.002V. A 0.2-volt input gives a count of 102, a 0.5-volt input gives a count of 255, and the measured values can be accurate to within 2 millivolts.

If you decrease the range, you also increase the converter's sensitivity to noise. With a 5V range, a 20-millivolt noise spike will cause at most a 1-bit error in the