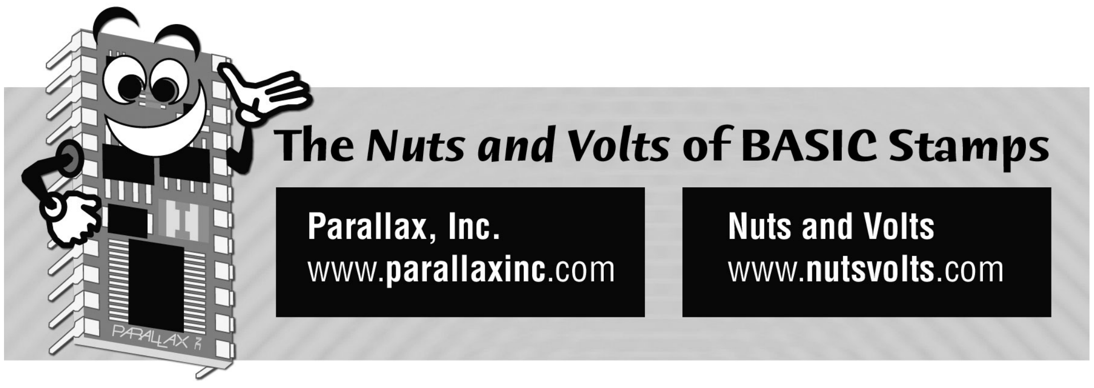


Column #63, July 2000 by Jon Williams:

Stamps in the Lab - Part 1

When Parallax introduced the BS2 I can remember thinking about what a great lab interface it would be. It's got 16 I/O lines, all kinds of neat functions and a serial interface to the PC that doesn't take away any of the I/O structure. Well, aside from a few experiments, I didn't do much in this regard. Thankfully, the clever guys at SelmaWare Solutions did. These guys have developed a really neat little program called Stamp Plot Lite that you can freely download from their site or from Parallax.

Stamp Plot Lite is a fully configurable, general-purpose graphing and datalogging utility. Stamp Plot Lite takes advantage of the BS2's DEBUG command, which is, in fact, the same as SEROUT at 9600 baud on pin 16. Stamp Plot Lite takes the information sent by DEBUG and deals with it accordingly:

- If the data is a decimal number it is graphed as the analog value.
- If the data is prefixed by "%" (by using the IBIN modifier) it is assumed to be binary data and the individual channels (up to 9) are graphed as digital (ON or OFF) values.
- If the data is prefixed by "!" it is processed as one of the various Stamp Plot Lite commands.
- If the data doesn't fit either of the above categories it is considered a general message and placed in a text box on the bottom of the screen.

The really neat part of the program is that it requires no setup. The Stamp sends all of the setup information for the graph after Stamp Plot Lite is connected. Since Stamp Plot Lite is so amazingly easy to use, I won't go into a great deal of detail here. There are a couple of notes that I do want to cover though before we get into a Stamp program that uses it.

First, Stamp Plot Lite will only graph one analog value (Note: by the time you read this article a "pro" version that graphs multiple analog channels will be available). The second note is on the "X" (horizontal) axis of the scale. This axis always represents seconds as Stamp Plot Lite will time stamp information as it arrives (the time stamp is relative to the start of the graph). If you want to graph for a specific period, you'll need to express the period in seconds to keep things straight.

Since Stamp Plot is easier to explain by putting it through its paces, let's assemble some simple hardware and give it a run.

Project Hardware

If you have a Parallax BASIC Stamp Activity Board (BSAC) and a Dallas Semiconductor DS1620 temperature sensor, you're ready to rock-and-roll. If you don't, the schematics in Figures 63.1, 63.2 and 63.3 will get you there.

Figure 63.1 is the programming port and DEBUG interface. The capacitors on the ATN line allow the Stamp to be reset by the PC but will block a steady state on the serial DSR line. This will be important later when we connect the Stamp to a terminal program. Note that if you have an older BSAC that only has one capacitor, you'll want to replace it with a two-pin header and removable jumper. The jumper will be installed for programming and removed for general serial communications.

Figure 63.2 is the button inputs and LED outputs (that will be used a little later). In case you're wondering why we've split the resistor on each I/O line into two...this configuration protects the I/O pin if it is in a HIGH (5 volts) state and the button gets pressed. The 330-ohm resistor limits the current to about 14 milliamps – a safe level for the pin.

Figure 63.3 is the interface to the Dallas DS1620 temperature sensor. Since we've covered the DS1620 on several occasions, we won't go into any detail about its workings. Refer to past articles by Scott Edwards and me for operational details of the DS1620.

Figure 63.1: Serial port connection

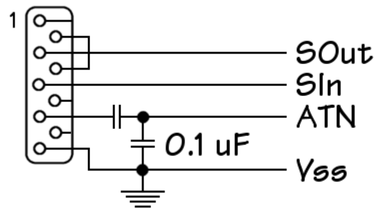


Figure 63.2: Stamp I/O pin connections

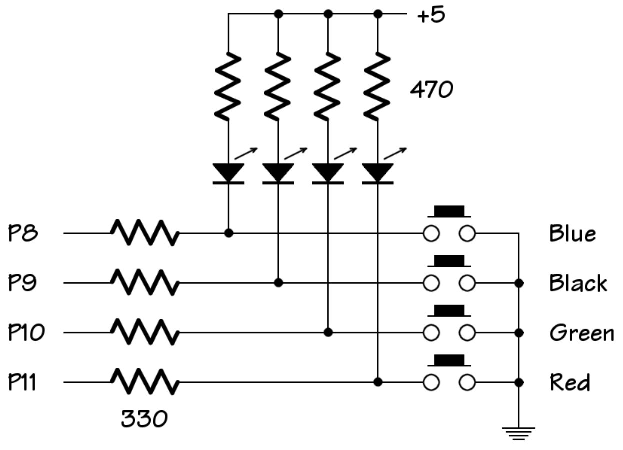
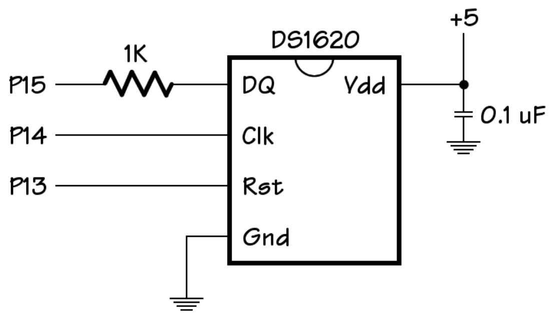


Figure 63.3: DS1620 hookups



How Well Is That A/C Working?...

For reasons I can't explain, I'm nutty about knowing the temperature. That's probably why we're using the DS1620 again. It also gives me an opportunity to see the accuracy of my air conditioner's thermostat and I suppose how well the insulation in my apartment is working.

Take a look at Program Listing 63.1 (TEMPPLOT.BS2). The purpose of this program is to measure the temperature every second for an hour and send it to Stamp Plot Lite for graphing.

Let's talk specifically about the interface to Stamp Plot Lite. We'll start in the section labeled Init_Graph. The first thing we do is issue the reset command, !RSET. This will clear the contents of the Stamp Plot Lite screen. Notice that the DEBUG command is terminated with CR (carriage return character -- same as 13). This is very important. The CR character lets Stamp Plot Lite know that the string has been entered and can be processed.

The next several commands are used to setup the graph. We'll start with the "Y" (vertical) axis. The !SPAN command sets the lower and upper limits of the graph. Since we're interested in monitoring the temperature in my living room, we've used "!SPAN 70,80" to set the lower limit of the graph to 70 degrees and the upper limit of the graph to 80 degrees.

The next command, !AMUL is very clever. This command stands for "Analog Multiplier." What Stamp Plot Lite does is multiply the analog value by this multiplier before sending it to the graph. This is great for converting raw sensor data to values that are meaningful. In our case, temperature is going to be sent to Stamp Plot Lite in tenths of degrees (if the air temperature is 73.5 degrees, the Stamp will send 735 to the PC). By using the multiplier, we will correctly display our temperature data.

Finally, we'll use !CLMM to clear any previously stored minimum and maximum values.

Now it's time to setup the "X" (horizontal) axis. !TMAX tells Stamp Plot Lite how wide the graph should be. Remember that this value is in seconds. In our case, we want to watch the temperature for an hour – 3600 seconds. !PNTS tells Stamp Plot Lite how many data points we want to collect. Since we want to fill the screen with data, well set the number of points to 3600; one data point per second for an hour. !MAXS causes the graph to stop when the maximum number of data points is reached.

The command !TSMP ON enables time stamping. The time stamp is relative to the start of the graph and is expressed in seconds.

The next few commands are designed to give information to the user about what's being monitored, graphed and (possibly) logged. !TITL sets the Stamp Plot Lite window title. !USRS puts a message in the user status box. This is the narrow box located just above the graph. !CLRM clears the general message box (located below the graph). The last DEBUG command has no specific formatting character and is assumed to be a general message. It is written to the general message box and stamped with the time-of-day and offset from reset.

The last thing we need to do is tell the graph to start plotting. This is accomplished with the !PLOT ON command.

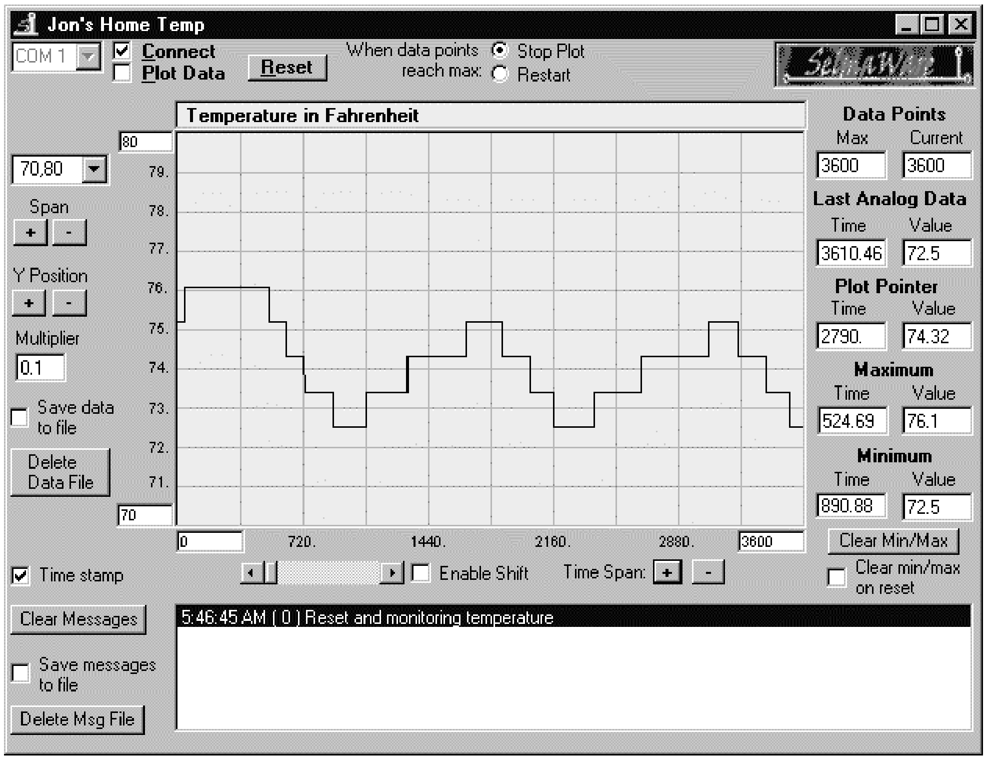
Okay, we've setup the graph so let's start sending some data. The first thing we do is grab the current air temperature from the DS1620. Remember that the data returned is expressed in half degrees using the Celsius scale. Bit8 of the temperature of is the negative indicator. Since we're measuring indoor temperatures, we're probably not ever going to see this bit set – but we'll check it, just to be safe (and if it is set, my goldfish is going to be really angry...).

The conversion to Fahrenheit probably looks familiar, if not a bit funky. The reason that there is no divisor (usually 5) in the equation is that the raw temperature reading is in half-degree increments. This makes converting to tenths pretty easy. Once we've done that, we'll send the temperature to Stamp Plot Lite with a DEBUG command that is terminated with CR.

Since it takes a little bit of time to read, convert and transmit the temperature, we'll set our PAUSE value to 990. This will give us almost exactly one second between readings.

Now we just need to start Stamp Plot Lite, click on the Connect check box and start our Stamp program (or restart it by pressing the reset button on the BSAC). Take a look at the screen shot in Figure 63.4. This shows the result of measuring the temperature in my living room for one hour. What we can see is that with a thermostat setting of 74, the temperature fluctuated between 72.5 and 76.1 degrees. It was a warm morning when I took the readings; almost 85 degrees outside.

Figure 63.4: Stamp Plot Lite showing DS1620 temperature measurements



Connect [✓] Plot Data [ ] When data points reach max: [✓ Stop Plot] Restart

Multiplier: 0.1 Time stamp [✓]

	Time	Value
Data Points (Max / Current)	3600	3600
Last Analog Data	3610.46	72.5
Plot Pointer	2790.	74.32
Maximum	524.69	76.1
Minimum	890.88	72.5

```
5:46:45 AM ( 0 ) Reset and monitoring temperature
```

Pretty neat, isn't it? – even with a very simple program. Stamp Plot Lite offers a lot of other cool features, including the ability to save the collected data to a file. And it's free! You can't beat that.

BASIC to BASIC

After working with Stamp Plot Lite for a while I got inspired to play with ideas I had about connecting the Stamp to a PC – the guys at SelmaWare had shown me the way. Now, I'm not interested in writing a graphing program, Stamp Plot Lite is already there and works great. What I would like, however, is a program that can display information collected by the Stamp and perhaps even do more advanced processing with raw sensor data.

As a starter, we'll write a Visual Basic program that will be compatible with most of the Stamp Plot Lite commands. Once we get that working, we can move on to bigger things.

Program Listing 63.2 is the Visual Basic code for the project. Keep in mind that this project – and any VB project the does serial communications – will require the MSComm control, which is found in only the Professional and Enterprise editions. Since our focus here is on receiving and process the Stamp Plot Lite commands, that's where we'll keep our focus.

The first thing we need to do is setup the MSComm control. The code to do this is found in the Form_Load event handler so that it gets setup when the program is started. The Settings parameter is set to "9600,N,8,1" to make it compatible with the DEBUG output from the Stamp. Other key parameters that we pay attention to are RThreshold and InputLen. We set both of these to 1 so that we can process each character as it arrives. Finally, we need to set the InputMode to text (comInputModeText) since we're sending text strings from the Stamp.

Okay, now that we're setup, here's how our program works. With the comm port selected and opened, the program waits for a character to come in. When it does, the MSComm1_OnComm event gets fired. We grab the character from the serial port and check to see if it's a carriage return (character 13). If it's not, we'll add it to our own buffer. When a carriage return is received, we send our buffer to the subroutine called ProcessBuffer.

The first thing that ProcessBuffer does is check the first character for “!” or “%” since these characters have special meaning from Stamp Plot Lite. Let’s assume that the first character was and exclamation point. This indicates a Stamp Plot Lite command so we’ll handle it separately with a subroutine called DoCommand.

DoCommand starts by looking for a space character in the buffer. A space is used to separate the command from any parameters. Experienced programmers might suggest that we could have just grabbed the first five characters from the buffer since the Stamp Plot Lite commands have a fixed length. Yes, we could have, but then we’d have had to gone back and grabbed the parameters for each command that uses them. It seemed easier to look for the delimiter (space) and extract the parameter string at the beginning. If a command uses parameters, we have already have them.

Now that we have our command (and sometimes a parameter string) we can process our compatible list with Select Case structure. When a command sends a single numeric parameter (like !AMUL), the parameter is converted with the CLng function. We have to keep in mind that Visual Basic Integers are signed, so Longs are used in our program to deal with PBASIC’s unsigned Integers.

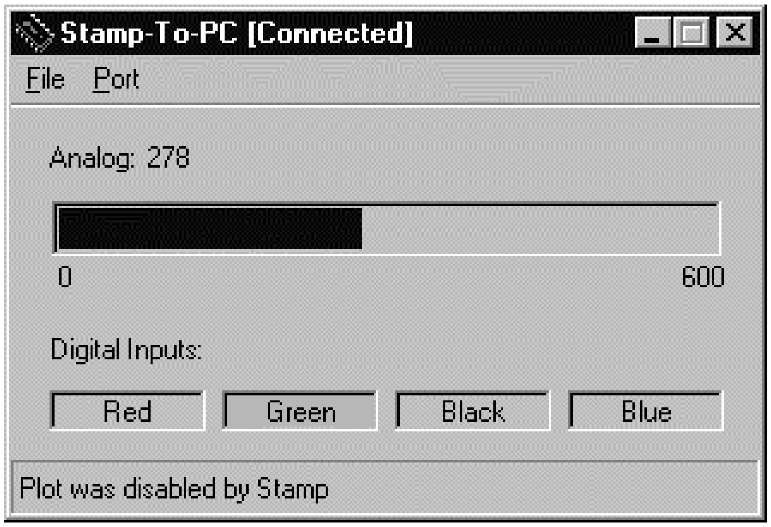
What VB doesn’t offer, however, is a binary to decimal conversion function. This is no surprise, since binary numbers are rarely necessary in PC applications. We need this function though, since our Stamp programs designed for Stamp Plot Lite will probably send binary data.

If we go back to ProcessBuffer we see that if the first character of our buffer is “%” then the data is extracted and converted to a numeric value with the function Bin2Dec. For this program, we could probably get away without the conversion, but by doing it, we give ourselves the opportunity to process and manipulate the data before sending it to the display.

To see the VB program in action, we need to load our Stamp with the demo program that comes with Stamp Plot Lite (you’ll find it in the Help file). Run the Stamp, then start the VB program. If you have a newer BSAC or used the circuit in Figure 63.1, the VB program can reset the Stamp when the comm port is opened. To do this, we need to leave the “Reset Stamp on Connection” option of the Port menu checked.

If the Stamp’s ATN is opened after programming (by removing a jumper) then we’ll have to manually reset the Stamp after the VB program is started and the comm port opened. Once this happens we’ll see the analog graph parameters setup (span is set from 0 to 600) and data will start coming in. Figure 63.5 shows the VB program in action.

Figure 63.5: Visual BASIC programming in action



Plot was disabled by Stamp

This Is Cool – Let's Do More!

Okay, we will. Next month we'll extend our Stamp program by allowing it to receive requests for information. This will allow us to "ask" for the information with a simple terminal program or a custom program that we'll write in VB, which will give us more precise control over when we get information and what we actually get. Finally, we'll update our Stamp program to receive data with a command. Again, this program will be compatible with a text terminal or a custom program and opens up all kinds of possibilities.

Until then, happy Stamping.

```
' Nuts & Volts - Stamp Applications
' July 2000 (Program Listing 63.1)

' ========================================================================
' Program... TEMPPLOT.BS2
' Author.... Jon Williams
' Started... 29 MAY 2000
' Updated... 29 MAY 2000
' ========================================================================


' -----[ Program Description ]--------------------------------------------
'
' Plot temperatures measured by a DS1620 using Stamp Plot Lite from
' SelmaWare Solutions (www.selmaware.com)


' -----[ Revision History ]-----------------------------------------------
'


' -----[ I/O Definitions ]------------------------------------------------
'
Rst     CON     13                      ' DS1620.3
Clk     CON     14                      ' DS1620.2
DQ      CON     15                      ' DS1620.1


' -----[ Constants ]------------------------------------------------------
'

' DS1620 commands
'
RTmp    CON     $AA                     ' read temperature
WTHi    CON     $01                     ' write TH (high temp)
WTLo    CON     $02                     ' write TL (low temp)
RTHi    CON     $A1                     ' read TH
RTLo    CON     $A2                     ' read TL
StartC  CON     $EE                     ' start conversion
StopC   CON     $22                     ' stop conversion
WCfg    CON     $0C                     ' write config register
RCfg    CON     $AC                     ' read config register


' -----[ Variables ]------------------------------------------------------
'
tmpIn   VAR     Word                    ' raw data from DS1620
halfBit VAR     tmpIn.Bit0              ' 0.5 degree C indicator
sign    VAR     tmpIn.Bit8              ' 1 = negative temperature
tempF   VAR     Word                    ' degrees F in tenths
```

```
' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
Init_DS1620:
       HIGH Rst                      ' alert the DS1620
       ' use with CPU; free run mode
       SHIFTOUT DQ,Clk,LSBFIRST,[WCfg, %10]
       LOW Rst
       PAUSE 10                      ' allow DS1620 EE write
       HIGH Rst
       ' start temp conversion
       SHIFTOUT DQ,Clk,LSBFIRST,[StartC]
       LOW Rst

Init_Graph:
       DEBUG "!RSET", CR             ' clear graph
       DEBUG "!SPAN 70,80", CR       ' disply 70 - 80 degrees
       DEBUG "!AMUL 0.1", CR         ' convert from tenths
       DEBUG "!CLMM"                 ' clear min/max values
       DEBUG "!TMAX 3600", CR        ' 1 hour scale
       DEBUG "!PNTS 3600", CR        ' graph every second
       DEBUG "!MAXS", CR             ' stop when graph full
       DEBUG "!TSMP ON", CR          ' enable time stamping

       DEBUG "!TITL Jon's Home Temp", CR    ' set window title
       DEBUG "!USRS Temperature in Fahrenheit", CR ' graph legend
       DEBUG "!CLRM", CR                    ' clear messages
       DEBUG "Reset and monitoring temperature", CR' message box

       DEBUG "!PLOT ON", CR          ' enable plotting


' -----[ Main ]------------------------------------------------------------
'
Main:  GOSUB GetTemp                 ' get the raw temperature
       IF sign = 0 THEN NotNeg       ' if positive, okay
       tmpIn = 0                     '  - otherwise make zero
NotNeg: tempF = (tmpIn * 9) + 320    ' convert to 10ths F
       DEBUG DEC tempF, CR           ' send to Stamp Plot Lite

       PAUSE 990                     ' wait about 1 second

       GOTO Main                     ' do it again


' -----[ Subroutines ]-----------------------------------------------------
```

```
'

GetTemp:
       HIGH Rst                      ' alert the DS1620
       SHIFTOUT DQ,Clk,LSBFIRST,[RTmp]      ' read temperature
       SHIFTIN DQ,Clk,LSBPRE,[tmpIn\9]      ' get the temperature
       LOW Rst
       RETURN
```

```
' Nuts & Volts - Stamp Applications
' July 2000 (Listing 63.2)

' ========================================================================
' Program... StampToPC.vbp
' Author.... Jon Williams
' Started... 26 MAY 2000
' Updated... 26 MAY 2000
' ========================================================================

Option Explicit

Dim showData As Boolean                 ' okay to show incoming data
Dim rxBuffer As String                  ' buffer for incoming characters
Dim multiplier As Single                ' analog multiplier

Private Sub Form_Load()

  ' center form
  Me.Left = (Screen.Width - Me.Width) / 2
  Me.Top = (Screen.Height - Me.Height) / 2
  Me.Caption = App.Title

  ' setup comm object
  With MSComm1
    .CommPort = 1
    .Settings = "9600,N,8,1"            ' setup for DEBUG
    .DTREnable = mnuPortResetStamp.Checked
    .RThreshold = 1                     ' process one char at a time
    .InputLen = 1
    .InputMode = comInputModeText       ' input will be strings
  End With

  multiplier = 1#                       ' analog multiplier
  SetSpan ("0,100")                     ' set span of progress bar
  ClearForm
  showData = False                      ' wait for reset

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)

  If MSComm1.PortOpen Then MSComm1.PortOpen = False

End Sub

Private Sub mnuFileExt_Click()

  Unload Me

End Sub

Private Sub mnuPortComX_Click(Index As Integer)

  ' deselect last port
  mnuPortComX(MSComm1.CommPort).Checked = False
  ' select new
  MSComm1.CommPort = Index
  mnuPortComX(Index).Checked = True

End Sub

Private Sub mnuPortConnect_Click()

  Dim x As Byte

  If Not (MSComm1.PortOpen) Then
    ' open the port
    On Error GoTo PortError
    MSComm1.PortOpen = True
    ' update the title bar
    Me.Caption = App.Title & " [Connected]"
    ' update port menu
    For x = 1 To 4
      mnuPortComX(x).Enabled = False
    Next
    mnuPortConnect.Caption = "&Disconnect"
  Else
    ' close the port
    MSComm1.PortOpen = False
    ' update the title bar
    Me.Caption = App.Title
    ' update port menu
    For x = 1 To 4
      mnuPortComX(x).Enabled = True
    Next
    mnuPortConnect.Caption = "&Connect"
  End If
  Exit Sub

PortError:
  MsgBox "Could not open Com" & Trim(Str(MSComm1.CommPort)) & ". " & _
         vbCr & "Please select another port.", _
         vbExclamation + vbOKOnly, App.Title

  On Error GoTo 0

End Sub

Private Sub mnuPortResetStamp_Click()

  mnuPortResetStamp.Checked = Not (mnuPortResetStamp.Checked)
  MSComm1.DTREnable = mnuPortResetStamp.Checked

End Sub

Private Sub MSComm1_OnComm()

  Dim newChar As String

  Select Case MSComm1.CommEvent
    Case comEvReceive
      newChar = MSComm1.Input
      If newChar = Chr(13) Then
        ProcessBuffer (rxBuffer)
        rxBuffer = ""
      Else
        rxBuffer = rxBuffer & newChar
      End If

    ' process other events here

  End Select

End Sub

Private Sub ProcessBuffer(ByVal strBuffer As String)

  Dim leadChar As String
  Dim param As String

  ' get leading character
  leadChar = Mid(strBuffer, 1, 1)

  Select Case leadChar
    Case "!"
      ' command string
      DoCommand (strBuffer)
    Case "%"
      ' binary data
      param = Trim(Mid(strBuffer, 2))
      If showData Then ShowDigital (Bin2Dec(param))
    Case Else
      If IsNumeric(strBuffer) Then
        ' buffer has analog data
        If showData Then ShowAnalog (CLng(strBuffer))
      Else
        ' buffer contains message
        sbarMessage.SimpleText = Trim(strBuffer)
      End If
  End Select

End Sub

Private Function DoCommand(ByVal theCommand As String)

  Dim delimPos As Integer
  Dim cmd As String
  Dim param As String

  ' remove any leading or trailing spaces
  theCommand = Trim(theCommand)

  delimPos = InStr(1, theCommand, " ")
  If delimPos = 0 Then
    ' no parameter(s)
    cmd = UCase(theCommand)
  Else
    ' command has parameter(s)
    ' - get command
    cmd = UCase(Mid(theCommand, 1, delimPos - 1))
    ' extract parameters from command string
    param = Mid(theCommand, delimPos + 1)
  End If

  ' process the command
  Select Case cmd
    Case "!RSET"
      ClearForm
      showData = True
    Case "!CLRM"
      If showData Then sbarMessage.SimpleText = ""
    Case "!USRS"
      If showData Then sbarMessage.SimpleText = param
    Case "!AMIN"
      pbarAnalog.Min = CLng(param)
    Case "!AMAX"
      pbarAnalog.Max = CLng(param)
    Case "!AMUL"
      multiplier = CSng(param)
    Case "!SPAN"
      SetSpan (param)
  End Select
End Function

Private Function Bin2Dec(ByVal binValue As String) As Long

  Dim temp As Long
  Dim binLen As Integer
  Dim x As Integer

  temp = 0
  binLen = Len(binValue)
  For x = 1 To binLen
    ' add bit value if "1"
    If Mid(binValue, x, 1) = "1" Then
      temp = temp + 2 ^ (binLen - x)
    End If
  Next

  Bin2Dec = temp

End Function

Private Sub SetSpan(ByVal span As String)

  Dim comma As Integer

  comma = InStr(1, span, ",")
  If comma = 0 Then Exit Sub            ' improper format - exit

  ' update progress bar
  pbarAnalog.Min = CLng(Mid(span, 1, comma - 1))
  pbarAnalog.Max = CLng(Mid(span, comma + 1))

  ' update legends
  lblSpanMin.Caption = Str(pbarAnalog.Min)
  lblSpanMax.Caption = Str(pbarAnalog.Max)

End Sub

Private Sub ShowAnalog(ByVal aValue As Long)

  aValue = CLng(CSng(aValue) * multiplier)

  ' show value
  lblAnalogValue.Caption = Trim(Str(aValue))

  ' check limits and show on progress bar
  If aValue > pbarAnalog.Max Then aValue = pbarAnalog.Max
  If aValue < pbarAnalog.Min Then aValue = pbarAnalog.Min
  pbarAnalog.Value = aValue
```

```
End Sub

Private Sub ShowDigital(ByVal digValue As Long)

  Dim mask As Long
  Dim led As Byte

  For led = 0 To 3
    If (digValue And (2 ^ led)) > 0 Then
      ' channel off - extinguish
      lblDigitalInput(led).BackColor = &H8000000F
    Else
      ' channel on - light
      lblDigitalInput(led).BackColor = vbGreen
    End If
  Next

End Sub

Private Sub ClearForm()

  ShowAnalog (0)
  ShowDigital (&HFFFF)                   ' all off (active low)
  sbarMessage.SimpleText = ""

End Sub
```