



Column #64, August 2000 by Jon Williams:

## Stamps In the Lab – Part 2

Last month we had some fun by keeping our PC connected to the Stamp and displaying information with a neat little program from SelmaWare called Stamp Plot Lite. Later, we went on to create a Visual Basic program that was compatible with Stamp Plot Lite output statements.

In both cases, the Stamp acted as an “information provider” and just pumped out data. There will be times though, when we want to *ask* the Stamp for a specific piece of information. We may also want to set some information in the Stamp that affects its behavior. That’s what we’re going to do this month.

### **Building a Two-Way Street**

Since we can’t learn to swim without getting wet, let’s jump right in. Take a look at the Stamp program in Program Listing 64.1 (if you don’t have a Parallax BASIC Stamp Activity Board, you can build an equivalent circuit using the schematics provided last month).

After the obligatory setup and initialization, the program waits on some serial input from the PC – in this case, coming in on the programming port (defined as Pin 16). The input,

## Column #64: Stamps in the Lab - Part 2

specifically, is going to start with a question mark and be followed by a hexadecimal number. This (hex) number is our query; it tells the Stamp what we want.

Why do it this way? Well, there are a couple of reasons. By using the HEX modifier in the SERIN function we're able to receive data in the form of ASCII text. This data could come from a Visual Basic program, but we haven't written that yet. Since we want to be sure the Stamp code is working as expected before we tackle the PC side of things, it would be nice if we could use some other tool to do our testing. With this technique we can. Since the Stamp is expecting a string input, we can use a general-purpose terminal program to see if things are working. We can even use a DEBUG terminal in the Windows version of the Stamp editor.

Let's go back to our SERIN function and discuss the details. First, SERIN waits for the "?" character to arrive, ignoring everything else until that happens. The question mark, then, is what signifies the start of a query. Once a question mark arrives, the HEX modifier causes the Stamp to look for valid hex characters (0 - 9, A - F). The arrival of any non-hex character (usually a carriage return [Enter] when using a terminal) tells the Stamp to stop accepting input (to the variable called param in our case) and continue on.

What has actually happened is that the Stamp has used the SERIN function to do a text-to-numeric conversion. Pretty cool, huh? Now that we have a command, we can use standard IF-THEN logic statements to see if it is valid for our particular program (Since there are so many possible commands and they are not numerically contiguous, using BRANCH to jump to the requested routine becomes a bit awkward). If the command isn't valid, a message and the offending input is displayed.

Could we have used "?ID" instead of "?F0" to get the Stamp's identification string? Yes, technically we could have, but it becomes very tedious in PBASIC programming. The reason is that the Stamp can't do direct string comparisons. What we would have to do is compare the input, character-by-character, to see if it's valid. And to be user-friendly, we'd have to allow uppercase and lowercase entries – doubling the number of comparisons. Since the HEX modifier of SERIN is not case-sensitive, our program will work if we enter "?F0" or "?f0" or any upper/lower combination.

For an example of character-by-character string comparisons in the Stamp, take a look at Jon Richards' networking demo at <http://www.jdrichards.com>.

The Stamp responds to a request by sending a text string by using DEBUG. Remember that DEBUG in the BS2 is the same as SEROUT on pin 16 at 9600 baud. Each of the response strings consists of a label, the equal sign, the value of that particular parameter

and finally, a carriage return. When using a terminal program, the output is easily readable. Something like this:

```
ID=Parallax BS2
```

The carriage return at the end of the output gives us a new line when using a terminal program and serves as a “end of input” when we process the input with our own program (like Stamp Plot Lite does). The equal sign will be used as a delimiter when we do our own processing. We’ll use it to distinguish the label from its value.

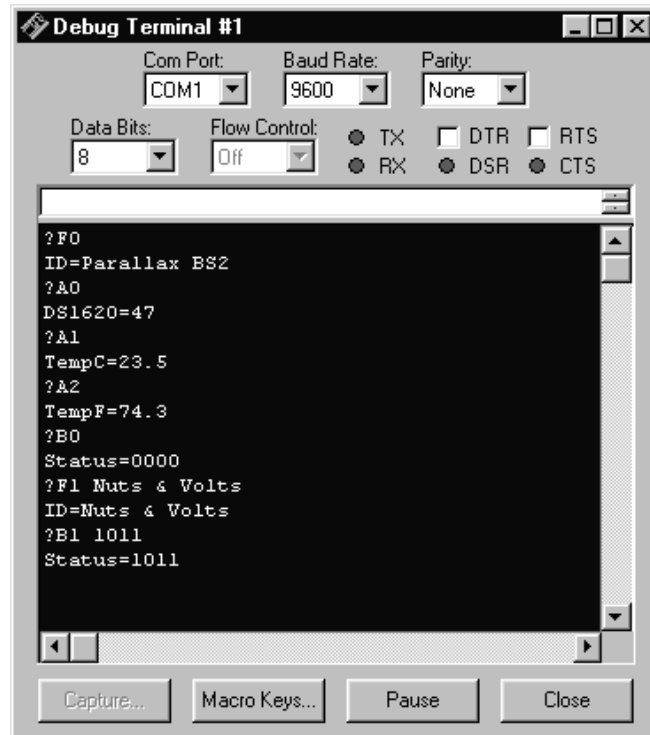
Most of our queries are requests for information. Two of them, however, can modify information that is stored in the Stamp. Let’s look at those.

The first one is “?F1” which will allow us to write a string value to the Stamp’s EEPROM (in a location called ID). When F1 is received as a command value, the program jumps to the subroutine called SetID. On entry to SetID, the EE pointer called addr is initialized, then the Stamp waits for a character to arrive. Notice that no modifier is used here. Since terminal programs and the Stamp represent characters using ASCII codes, we don’t have to do anything special. When a character does arrive, it is written to the EE and the address pointer is incremented. If the last character was a carriage return (13), the program outputs the new string (using the code at ShowID), otherwise it loops back and waits for another character.

The second modifying query is “?B1” which allows us to set the status of four LEDs. Take a look at the subroutine called SetLEDs. This time, we’re going to use the BIN modifier of SERIN. What this means is that we valid inputs are going to be “1” (LED is on) and “0” (LED is off). Since the hardware uses active low outputs, we’re forced to invert the data. That’s easily handled with the complement operator (~).

Figure 64.1 shows an actual online session with this Stamp program using the DEBUG terminal. If you’re not using the Windows Stamp editor, you can use any generic terminal program instead (make sure that Local Echo is disabled). Each command is terminated with a carriage return (Enter on the PC keyboard). For “?F1” and “?B1” a space was used as the separator between the command and the data. And don’t forget the SERIN DEC modifier. If you develop a program (like this one) where you can enter data from a terminal, using the DEC modifier will make it easy.

Figure 64.1: BASIC Stamp debug session

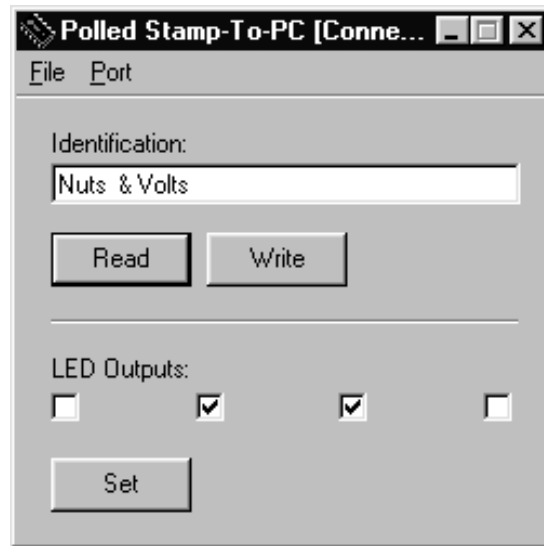


### More BASIC to BASIC

Now that we have a Stamp project that will respond to requests we can use Visual Basic to create a program that handles or even automates them for us. This is a pretty simple process; in fact, we only have to make a very small addition to the code that we created last month.

Essentially, we just have to build a command string for each of our functions and send it to the Stamp. What we have to keep in mind though is that the MSComm control in VB does not have the pacing parameter that the Stamp's SEROUT function has and we're going to be forced to code around this. We need some pacing between characters because we're using SERIN modifiers and in the case of the ID string, we're writing the characters to the Stamp's EEPROM as they come in. These processes take time.

Figure 64.2: Visual BASIC intercepts debug commands



Program Listing 64.2 is the VB code for a small demo program (see Figure 64.2) that takes advantage of a couple of the features of our Stamp project. This program will allow us to get and set the identification string and set the BSAC LEDs.

Remember that Visual Basic is event-driven, that is, nothing happens until something happens. In our case, things get started when we connect to the Stamp (Connect on the Port menu) and the click on one of the buttons. Let's go through one.

When we click on the button labeled "Read" a subroutine called `cmdIDRead_Click` is executed. This subroutine calls `SendStr` with the data "?F0" (read the ID). If we jump down to `SendStr` we see that it sends each character of the passed string with a five-millisecond delay in between them. Once all of the characters have been sent, it completes the transmission with a carriage return (13).

One of the details that we have to deal with when using the Stamp's programming port for general serial communications is that it echoes back everything that it receives. What this means is that every character we send to the Stamp through the programming port comes back in the on the receive line. When a character arrives, the `MSComm1_OnComm` subroutine gets called.

## Column #64: Stamps in the Lab - Part 2

MSComm1\_OnComm collects all incoming characters in a string called rxBuffer until it sees a carriage return. When that happens, it passes the buffer to the subroutine (cleverly called) ProcessBuffer for handling.

ProcessBuffer looks at the first character of the incoming string. If it's a "?" we know that the string is the command we just sent and it can be discarded. If the first character is something else, we'll check it for "=" and parse the label and its parameter. A Select-Case structure lets us iterate through available labels and deal with the parameter. In the case of "ID" we send the parameter to the Identification text box. Prove to yourself that this works by clearing the text box and then clicking on "Read."

This is a simple demo and doesn't do a lot – but is a very good shell for any application of this sort that you might want to develop. And you can always automate the polling by adding a timer to the project and using it to send a command. You can also use the PC's clock to create scheduled polling events. It's up to you now. With this basic communications infrastructure in place, there's no reason not to partner your PC and Stamps on sophisticated projects.

### Can We Do Both?

Last month our Stamp program did its processing served up information without asking. This month we've gone to the other extreme: the Stamp sits and waits for a command before doing anything. Can we find a happy middle ground?

Yes we can -- with caution. We might, for example, create a device that normally runs on its own but has the facility to display and set parameters. In this case, we could use an input to indicate Run or Review mode. When the input (switch) is set to Review, the program would jump to a section that waits for an input query (like Main in this month's program).

When we do this, we'll want to add the timeout parameter to the SERIN function. On a timeout, the program would jump to the Run/Review switch check. If back in Run, the program would go back to its normal operations, otherwise it would continue waiting for some input from the PC.

One of the neat things about using the event-driven serial input of VB is that we don't have to wait around for a specific number of characters to arrive. If you're using the Stamp's programming port of serial communications, be careful not to mix polled commands to the Stamp with spontaneous serial data.

Remember what we just covered: when using the Stamp programming port, the PC's serial transmit pin gets tied back to its serial receive pin. Our VB program deals with this by checking the first character of the buffer for the query character. If we allow the Stamp to spontaneously transmit in a normally polled situation, however, that transmission could take place right in the middle of our polling sequence, corrupting the buffer. And since the Stamp can't receive and transmit serial data at the same time, the incoming data would be missed. Again, be cautious and try not to mix spontaneous and polled communications.

### **Comm Control For Everyone**

I erred last month when I stated that you had to have the Professional or Enterprise editions of Visual Basic to the MSComm control. There is a work-around for those of you that are using the Learning edition or use a language other than VB.

Serial communications guru, Richard Grier, has created a free control that you can use with VB (and other programming languages that support ActiveX controls) called XMComm. XMComm uses the MSComm control internally and has all its capabilities. On top of that, it also supports XModem communications for sending files between two computers. Richard has written an excellent book called "Visual Basic Programmer's Guide to Serial Communications." This book, along with Jan Axelson's "Serial Port Complete" should be on your desk if you intend to do any serious serial-communications programming.

### **What's Next?**

What I do know is that I don't know for sure except that I'm toying with some Stamp-based robotics ideas. I've got my ideas, but I'd really rather hear yours. Drop me a note and we'll see what we can come up with. Until then, happy Stamping.

Richard Grier

"Visual Basic Programmer's Guide to Serial Communications"

<http://www.mabry.com>

XMComm control

[http://ourworld.compuserve.com/homepages/richard\\_grier](http://ourworld.compuserve.com/homepages/richard_grier)

Jan Axelson

"Serial Port Complete"

<http://www.lvr.com>

## Column #64: Stamps in the Lab - Part 2

```
' Nuts & Volts - Stamp Applications
' August 2000 (Program Listing 64.1)

' =====
' Program... POLLSTMP.BS2
' Author... Jon Williams
' Started... 25 MAY 2000
' Updated... 29 JUN 2000
' =====

' -----[ Program Description ]-----
'
' This program waits for a text command from the PC and responds with a
' string that contains the name of the data and its current value.
'
' If the ATN line is opened, this program will work with any general-
' purpose terminal program (i.e., HyperTerm). To read the temperature you
' would type the characters '?', 'A', '1', followed by a carriage return.

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

' -----[ I/O Definitions ]-----
'
SIOPin CON    16          ' serial I/O on programming port

Rst     CON    13          ' DS1620.3
Clk     CON    14          ' DS1620.2
DQ      CON    15          ' DS1620.1

' -----[ Constants ]-----
'
Baud96 CON    84          ' 9600-8-N-1 (matches DEBUG)

CmdID   CON    $F0        ' get string ID
CmdSet  CON    $F1        ' set string ID
CmdTmp  CON    $A0        ' get DS1620 - display raw count
CmdTmpC CON    $A1        ' get DS1620 - display in C
CmdTmpF CON    $A2        ' get DS1620 - display in F
CmdStat CON    $B0        ' get digital output status
CmdLEds CON    $B1        ' set LED outputs

' DS1620 commands
'
RTmp    CON    $AA        ' read temperature
WTHi    CON    $01        ' write TH (high temp register)
WTLo    CON    $02        ' write TL (low temp register)
RTHi    CON    $A1        ' read TH
```

Column #64: Stamps in the Lab - Part 2

```

RTLo    CON    $A2                ' read TL
StartC  CON    $EE                ' start conversion
StopC   CON    $22                ' stop conversion
WCfg    CON    $0C                ' write configuration register
RCfg    CON    $AC                ' read configuration register

' -----[ Variables ]-----
'
cmd      VAR    Byte              ' command from PC/terminal
addr    VAR    Byte              ' EE address pointer
eeDat   VAR    Byte              ' EE data
param   VAR    Word              ' parameter from PC
char    VAR    param.LOWBYTE     '
tmpIn   VAR    Word              ' raw data from DS1620
halfBit VAR    tmpIn.Bit0        ' 0.5 degree C indicator
sign    VAR    tmpIn.Bit8        ' 1 = negative temperature
tempC   VAR    Word              ' degrees C in tenths
tempF   VAR    Word              ' degrees F in tenths
potVal  VAR    Word              ' reading from BSAC pot
buttons VAR    Nib               ' BSAC input buttons

' -----[ EEPROM Data ]-----
'
ID      DATA "Parallax BS2", CR  ' CR-terminated string

' -----[ Initialization ]-----
'
      OUTC = %1111                ' turn off LEDs (active low)
      DIRC = %1111                ' make port C all outputs

      HIGH Rst                    ' alert the DS1620
      ' use with CPU; free run mode
      SHIFTOUT DQ,Clk,LSBFIRST,[WCfg, %10]
      LOW Rst
      PAUSE 10                    ' pause for DS1620 EE write cycle
      HIGH Rst
      ' start temp conversion
      SHIFTOUT DQ,Clk,LSBFIRST,[StartC]
      LOW Rst

' -----[ Main ]-----
'
Main:  ' wait for a command
      cmd = 0
      SERIN SIOpin,Baud96,[WAIT ("?"),HEX cmd]

      ' check for valid command

```

## Column #64: Stamps in the Lab - Part 2

```
IF cmd = CmdID THEN ShowID
IF cmd = CmdSet THEN SetID
IF cmd = CmdTmp THEN ShowTemp
IF cmd = CmdTmpC THEN ShowTempC
IF cmd = CmdTmpF THEN ShowTempF
IF cmd = CmdStat THEN ShowStat
IF cmd = CmdLEDs THEN SetLEDs

BadCommand:
  DEBUG "Invalid Command: ",HEX2 cmd,CR
  GOTO Main

' -----[ Subroutines ]-----
'
ShowID: DEBUG "ID="                ' label output
        addr = ID                  ' point to first character of ID
GetEE:  READ addr, eeDat           ' read a character from EEPROM
        DEBUG eeDat                ' print the character
        addr = addr + 1            ' point to next character
        IF eeDat <> CR THEN GetEE  ' if not CR, read another
        GOTO Main

SetID:  addr = ID                  ' point to ID location
GetC:   SERIN SIOpin,Baud96,[char] ' get character from PC
        WRITE addr, char           ' write character to EEPROM
        addr = addr + 1            ' point to next location
        IF char <> CR THEN GetC    ' if not CR, wait for another
        GOTO ShowID               ' confirm new ID

ShowTemp:
  GOSUB GetTemp
  DEBUG "DS1620=",DEC tmpIn,CR    ' send raw temp to PC
  GOTO Main

ShowTempC:
  ' will only return temps above freezing (0 C)
  GOSUB GetTemp
  IF sign = 0 THEN NoNegC
  tmpIn = 0
NoNegC: ' convert raw count to 10ths C
        tempC = tmpIn * 5
        DEBUG "TempC=",DEC (tempC/10),".",DEC (tempC//10),CR
        GOTO Main

ShowTempF:
```

```

        ' will only return temps above freezing (32 F)
        GOSUB GetTemp
        IF sign = 0 THEN NoNegF
        tmpIn = 0
NoNegF: ' convert raw count to 10ths F
        tempF = (tmpIn * 9) + 320
        DEBUG "TempF=",DEC (tempF/10),".",DEC (tempF//10),CR
        GOTO Main

ShowStat:
        ' show LED status
        DEBUG "Status=", BIN4 ~OUTC, CR
        GOTO Main

SetLEDs:
        ' wait for output bits
        ' - as binary string
        '
        SERIN SIOpin,Baud96,[BIN param]
        OUTC = ~param.LOWNIB           ' set the outputs
        GOTO ShowStat                 ' confirm new outputs

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

```

## Column #64: Stamps in the Lab - Part 2

```
' Nuts & Volts - Stamp Applications
' August 2000 (Program Listing 64.2)

' =====
' Program... PolledStamp.VBP
' Author... Jon Williams
' Started... 25 MAY 2000
' Updated... 2 JUL 2000
' =====

Option Explicit

Dim rxBuffer As String      ' response from Stamp
Dim roundRobin As Byte     ' polling control
Dim okayToClose As Boolean  ' okay to quit program?

Private Declare Function timeGetTime Lib "winmm.dll" () As Long

Private Sub cmdIDRead_Click()

    SendStr ("?F0")

End Sub

Private Sub cmdIDWrite_Click()

    SendStr ("?F1 " & Trim(txtIDstring.Text))

End Sub

Private Sub cmdSetLEDs_Click()

    Dim x As Integer
    Dim ledStr As String

    ledStr = ""                ' clear status string
    For x = 0 To 3            ' build binary string of status
        If chkLED(x).Value = 1 Then
            ledStr = ledStr & "1"
        Else
            ledStr = ledStr & "0"
        End If
    Next

    SendStr ("?B1 " & ledStr)    ' send command and status
```

```

End Sub

Private Sub Form_Load()

    Dim x As Integer

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

    ' clear ID text
    txtIDstring.Text = ""
    cmdIDRead.Enabled = False
    cmdIDWrite.Enabled = False

    ' uncheck LEDs
    For x = 0 To 3
        chkLED(x).Value = 0
    Next
    cmdSetLEDs.Enabled = False

    ' 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                     ' grab one char at a time
        .InputMode = comInputModeText    ' input will be strings
        .SThreshold = 0                   ' don't wait to send
    End With

    okayToClose = True

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

    Cancel = Not (okayToClose)

End Sub

Private Sub Form_Unload(Cancel As Integer)

    If MSComm1.PortOpen Then MSComm1.PortOpen = False

End Sub

```

## Column #64: Stamps in the Lab - Part 2

```
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 okayToClose And (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"  
        ' enable form controls  
        cmdIDRead.Enabled = True  
        cmdIDWrite.Enabled = True  
        cmdSetLEDS.Enabled = True  
    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"  
        ' disable form controls  
        cmdIDRead.Enabled = False  
    End If  
End Sub
```

```

    cmdIDWrite.Enabled = False
    cmdSetLEDs.Enabled = False
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 buffer As String)

Dim leadChar As String
Dim delimPos As Integer
Dim label As String
Dim param As String

```

## Column #64: Stamps in the Lab - Part 2

```
' get leading character
leadChar = Mid(buffer, 1, 1)

If leadChar = "?" Then
  ' echoed query - ignore
Else
  ' process the response
  delimPos = InStr(1, buffer, "=")
  If delimPos > 0 Then
    ' extract label and parameter
    label = UCase(Trim(Mid(buffer, 1, delimPos - 1)))
    param = Trim(Mid(buffer, delimPos + 1))
    ' process known responses
    Select Case label
      Case "ID"
        txtIDstring.Text = param

      Case "DS1620"
        ' process raw temperature

      Case "TEMPC"
        ' display celcius temp

      Case "TEMPF"
        ' display fahrenheit temp

      Case "STATUS"
        ' confirm LED status

      Case Else
        ' unknown label

    End Select
  Else
    ' buffer has no delimiter
    ' (error message)
  End If
End If

End Sub

Public Sub Delay(milliseconds As Single)
  Dim timeOut As Single

  timeOut = milliseconds + timeGetTime()
  Do Until timeGetTime() >= timeOut
    DoEvents
  Loop
End Sub
```

```

Private Sub SendStr(ByVal txBuf As String)

    Dim x As Integer

    ' can't quit while transmitting
    okayToClose = False

    For x = 1 To Len(txBuf)
        MSComm1.Output = Mid(txBuf, x, 1)
        ' give Stamp time to receive and process the character
        Delay (5)
    Next
    ' add CR to end of command
    MSComm1.Output = Chr(13)

    okayToClose = True

End Sub

Private Function Dec2Bin(ByVal decValue As Long) As String

    Dim tmpBin As String
    Dim testBit As Long

    tmpBin = ""
    testBit = 1

    Do
        If (testBit And decValue) > 0 Then
            tmpBin = "1" & tmpBin
            decValue = decValue - testBit
        Else
            tmpBin = "0" & tmpBin
        End If
        testBit = testBit * 2
    Loop While (decValue > 0)

    Dec2Bin = tmpBin

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)

```

## Column #64: Stamps in the Lab - Part 2

```
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
```