Chapter III
Software

3.0 Importance of VB
Evolution of Visual Basic
VB 1.0 was introduced in 1991. The approach for connecting the programming language to the graphical user interface is derived from a prototype developed by Alan Cooper called Tripod. Microsoft contracted with Cooper and his associates to develop Tripod into a programmable shell for Windows 3.0, under the code name Ruby (no relation to the Ruby programming language).

Tripod did not include a programming language at all, and Ruby contained only a rudimentary command processor sufficient for its role as a Windows shell. Microsoft decided to use the simple Program Manager shell for Windows 3.0 instead of Ruby, and combine Ruby with the Basic language to create Visual Basic. Ruby provided the "visual" part of Visual Basic—the form designer and editing tools—along with the ability to load dynamic link libraries containing additional controls (then called "gizmos"). Ruby's extensible gizmos later became the VBX interface.

Language Features
Visual Basic (VB) is an event driven programming language and associated development environment from Microsoft. The version of VB was derived heavily from BASIC and enables the rapid application development (RAD) of graphical user interface (GUI) applications, access to databases using various database connectivity methods.

A programmer can put together an application using the components provided with Visual Basic itself. In business programming, Visual Basic has one of the largest user bases. In a survey conducted, 62% of developers reported using some form of Visual Basic.

Visual Basic was designed to be easy to learn and use. The language not only allows programmers to easily create simple GUI applications, but also has the flexibility to develop fairly complex applications as well. Programming in VB is a combination of visually arranging components or controls on a form, specifying
attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code. Performance problems were experienced by earlier versions, but with faster computers and native code compilation this has become less of an issue.

3.1 Main Program

Dim RepSiglndx As Integer
Dim RepFlag As Boolean
Dim CurPoint As Integer
Dim CmdStr(1 To 3) As Integer
Dim CmdIndx As Byte

Private Sub Com_OnComm()
On Error GoTo ErrHnd

    Select Case Com.CommEvent
    Case 1004
        Commsg = "Framing Error"
    Case 1006
        Commsg = "Port OverFlow"
    Case 1008
        Commsg = "Receive Buffer OverFlow"
    Case 1011
        Commsg = "UnExpected Error"
    Case 1
        Commsg = "Send"
    Case 2
        Commsg = "Receive"

End Sub
DataStr = Empty
DataStr = Com.Input
If Len(DataStr) <> 5 Then
    Com.InBufferCount = 0
    MsgBox "Frame Receive Error", vbCritical, "Receive Error"
ElseIf Mid(DataStr, 1, 2) <> "#T" Or Mid(DataStr, 5, 1) <> "!" Then
    Com.InBufferCount = 0
    MsgBox "Frame Receive Error", vbCritical, "Receive Error"
Else
    TempVal = Val(Mid(DataStr, 3, 2))
    DrawGraph
    List1.AddItem TempVal, 0
    If List1.ListCount > 5 Then
        List1.RemoveItem 5
    End If
End If
End If
Case Else
    Commsg = Com.ComCommEvent
End Select
    'Label3.Caption = Commsg
Exit Sub

ErrHnd:
    MsgBox "Error - " & Err.Number & " -" & Err.Description, vbCritical
End Sub

Private Sub Command1_Click()
If Com.PortOpen Then
    Command3_Click
    Exit Sub
End Sub
Private Sub Command2_Click()
If Com.PortOpen = False Then
    MsgBox "No Com Port Available!", vbInformation, "ComPort"
    Exit Sub
End If
If IsNumeric(txtSetPoint) Then
    If Val(txtSetPoint) > 99 Or Val(txtSetPoint) < 0 Then
        MsgBox "Invalid Temperature Range", vbInformation, "Error Input"
    Else
        Com.Output = "#S" + Trim(txtSetPoint) + "!"
        SetLine.Y1 = 99 - Val(txtSetPoint)
        SetLine.Y2 = 99 - Val(txtSetPoint)
    End If
Else
    MsgBox "Invalid Number", vbInformation, "Error Input"
End If
End Sub

Private Sub Command3_Click()
On Error GoTo ErrHnd
If Com.PortOpen = False Then
    Com.CommPort = PortNo
    Com.PortOpen = True
    Command3.Caption = "Stop"
    prey = P.ScaleHeight
Else
Com.PortOpen = False
Command3.Caption = "Start"
End If
Exit Sub
ErrHnd:
MsgBox "Error - " & Err.Number & " - " & Err.Description, vbCritical
End Sub

Public Sub DrawGraph()

Val1 = 99 - TempVal

x = x + 1
P.Line (prex, preyHx, Val1)
prex = x
prey = Val1
If x > P.ScaleWidth Then
    x = 1
    P.Cls
    prex = 0
End If
Exit Sub
ErrHnd:
MsgBox "Error - " & Err.Number & " - " & Err.Description, vbCritical

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyF1 Then
    Text1.Visible = Not Text1.Visible
    If Text1.Visible = True Then Text1.SetFocus
End Sub
Private Sub Option1_Click(Index As Integer)
PortNo = Index + 1
If Com.PortOpen = True Then
    Command3_Click
    Command3_Click
Else
    Command3_Click
End If
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = vbKeyReturn Then
    Com.Output = Trim(Text1)
End If
End Sub

.BAS File

'Data Acquisition
Public DataStr As String
Public TempVal As Integer
Public x As Long, y As Long
Public prex As Long, prey As Long
Public PortNo As Integer

Form_Load()
This function sets up graph with ranges on X and Y axes, with Temperature (in Celsius) on Y-axis and Seconds on X-axis.
The Button Start will execute the application.

The Button Stop will stop the application from executing.

The Button Set will allow the application to monitor the temperature specified in the Textbox.

Com_OnComm()
This function monitors the data received from the USB Controller. This monitoring is displayed on the graph which is setup on Form load event

The option buttons will allow the user to select the Com port.

The .BAS file is used to declare global variables used by the application