```
Option Strict On
Option Explicit On

'////////////////////////////////////////////////////////////////////////////////
'// MainForm is the Host PC's program to test the PC-PCB Interface.
'// Programmed December 9, 2011.
'////////////////////////////////////////////////////////////////////////////////

'////////////////////////////////////////////////////////////////////////////////
'// The four-bit commands which the Host PC sends to the PCB are:-
'// b'0001'=1  PCB will reply with nibble b'1110'=d'14'.
'// b'0010'=2  PCB will reply with nibble b'0011'=d'3'.
'// b'0011'=3  PCB will wait one second and then reply with nibble b'0101'=d'5'.
'// b'0100'=4  PCB will turn on the LED.
'// b'0101'=5  PCB will turn off the LED.
'// b'0110'=6  PCB will report the status of the switch SW (open=d'1'; closed=d'0').
'// To all other commands, PCB will reply with b'1100'=d'12'.
'////////////////////////////////////////////////////////////////////////////////

Public Class HostProgram
    Inherits System.Windows.Forms.Form

    ' Parallel port variables
    Private Const DataRegAddress As Int32 = &H378    ' Data register address
    Private Const StatusRegAddress As Int32 = &H379  ' Status register address
    Private Const ControlRegAddress As Int32 = &H37A ' Control register address
    Private Const ECRegAddress As Int32 = &H77A      ' Extended Control register address
    Private StatusRegData As Int32
    Private ControlRegData As Int32
    Private ECRegData As Int32

    ' Communication variables
    Private DataIn As Int32                          ' Nibble received from the PCB
    Private DataOut As Int32                         ' Nibble to send to the PCB
    Private PCBCommError As Boolean                  ' True if communication times out

    ' Communication timeout delay
    Private Const CommTO As Int32 = 2000             ' Time out after two seconds

    Public Sub New()
        InitializeComponent()
        With Me
            Name = "MainForm"
            Text = "PC-PCB Interface"
            FormBorderStyle = Windows.Forms.FormBorderStyle.Fixed3D
            Size = New Drawing.Size(340, 280)
            CenterToScreen()
            MinimizeBox = False
            MaximizeBox = False
            Controls.Add(labelDescription) : labelDescription.BringToFront()
            Controls.Add(labelNibbleOut) : labelNibbleOut.BringToFront()
            Controls.Add(textboxNibbleOut) : textboxNibbleOut.BringToFront()
            Controls.Add(buttonSend) : buttonSend.BringToFront()
            Controls.Add(labelNibbleIn) : labelNibbleIn.BringToFront()
            Controls.Add(textboxNibbleIn) : textboxNibbleIn.BringToFront()
            Controls.Add(labelStatus) : labelStatus.BringToFront()
            Controls.Add(buttonExit) : buttonExit.BringToFront()
            Visible = True
```

```
        PerformLayout()
        BringToFront()
    End With
    Initialization()
End Sub

Private Sub Initialization()
    ' Configure the Enhaced Capabilities Port as a bidirectional Standard
    ' Parallel Port by setting the top three bits of ECRRegAddress to b'001'.
    ECRegData = Inp(ECRegAddress)
    ECRegData = ECRegData And &H1F
    ECRegData = ECRegData Or &H20
    Out(ECRegAddress, ECRegData)
    ' Disable interrupts through the Standard Parallel Port by setting
    ' ControlRegAddress<4> low.
    ControlRegData = Inp(ControlRegAddress)
    ControlRegData = ControlRegData And &HEF
    Out(ControlRegAddress, ControlRegData)
    ' Put the data bus into its high-impedance state.
    ConfigureDataBusForInput()
    ' Assert low the interrupt output line to the PCB.
    AssertDsubPin1Low()
    ' Wait two seconds before testing if the PCB is ready.
    Threading.Thread.Sleep(2000)
    ' Do not proceed until the PCB is ready: D-sub pin 10 will be low.
    Do While True
        If (ReadDsubPin10() = False) Then
            Exit Do
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
    MainProgram()
End Sub

Private Sub MainProgram()
    Do While True
        ' Check to see if the PCB is requesting an interrupt.
        If (ReadDsubPin10() = True) Then
            DataIn = ReceiveNibble()
            If (DataIn < 0) Then
                labelStatus.Text = "Reception failed."
            Else
                textboxNibbleIn.Text = Trim(Str(DataIn))
                labelStatus.Text = "Reception succeeded."
            End If
            Me.Refresh()
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
End Sub

'////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////
'// Controls for MainForm.

Private labelDescription As New Windows.Forms.Label With _
```

```
    {.Size = New Drawing.Size(300, 150), _
     .Location = New Drawing.Point(25, 10), _
     .Text = "Description of commands:" & vbCrLf & _
             "  1 = PCB should reply with '14'." & vbCrLf & _
             "  2 = PCB should reply with '3'." & vbCrLf & _
             "  3 = PCB should wait one second, then reply with '5'." & vbCrLf & _
             "  4 = PCB should turn on the LED." & vbCrLf & _
             "  5 = PCB should turn off the LED." & vbCrLf & _
             "  6 = PCB should report the status of the switch ('1'=open).", _
    .TextAlign = ContentAlignment.TopLeft}

Private labelNibbleOut As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(150, 20), _
     .Location = New Drawing.Point(25, 120), _
     .Text = "Nibble to send to PCB", .TextAlign = ContentAlignment.MiddleLeft}

Private WithEvents textboxNibbleOut As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(50, 20), _
     .Location = New Drawing.Point(180, 120), _
     .Text = "", .TextAlign = HorizontalAlignment.Left}

Private WithEvents buttonSend As New Windows.Forms.Button With _
    {.Size = New Drawing.Size(75, 30), _
    .Location = New Drawing.Point(235, 115), _
    .Text = "Send nibble", .TextAlign = ContentAlignment.MiddleCenter}

Private labelNibbleIn As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(150, 20), _
     .Location = New Drawing.Point(25, 150), _
     .Text = "Nibble received from PCB", .TextAlign = ContentAlignment.MiddleLeft}

Private textboxNibbleIn As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(50, 20), _
     .Location = New Drawing.Point(180, 150), _
     .Text = "", .TextAlign = HorizontalAlignment.Left}

Private labelStatus As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(300, 20), _
     .Location = New Drawing.Point(25, 180), _
     .Text = "", .TextAlign = ContentAlignment.MiddleLeft}

Private WithEvents buttonExit As New Windows.Forms.Button With _
    {.Size = New Drawing.Size(75, 30), _
    .Location = New Drawing.Point(235, 205), _
    .Text = "Exit", .TextAlign = ContentAlignment.MiddleCenter}

'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
'// Handlers for controls for MainForm.

Private Sub textboxNibbleOut_Change() Handles textboxNibbleOut.TextChanged
    ' Clear the display from the previous command when the user types.
    textboxNibbleIn.Text = ""
    labelStatus.Text = ""
    Me.Refresh()
End Sub

Private Sub textboxNibbleOut_Click() Handles textboxNibbleOut.MouseClick
```

```
    ' Clear this textbox if the user clicks on it.
    textboxNibbleOut.Text = ""
    Me.Refresh()
End Sub

Private Sub buttonSend_Click() Handles buttonSend.MouseClick
    ' Validate the user's entry in the textboxNibbleOut field.
    DataOut = CInt(Val(textboxNibbleOut.Text))
    If ((DataOut < &H0) Or (DataOut > &HF)) Then
        labelStatus.Text = "Error: Command must be between 0 and 15."
        Me.Refresh()
        Exit Sub
    End If
    ' Send the user's entry to the PCB.
    Dim Result As String
    Result = SendNibble(DataOut)
    ' Display the status of the transmission.
    labelStatus.Text = "Transmission status: " & Result & "."
    Me.Refresh()
End Sub

Private Sub buttonExit_Click() Handles buttonExit.MouseClick
    Application.Exit()
End Sub

'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
'// Communication procedures.

Private Function SendNibble(ByVal dataout As Int32) As String
    ' Sends an integer in the range 0 - 15, representing four bits of data, to
    ' the PCB.  Returns "SUCCESS" if the transmission is successful or a string
    ' describing the error if it is not.
    ' Delay 1ms to ensure PCB is finished with previous communication cycle, if any.
    Threading.Thread.Sleep(1)
    ' Dispose of any prior instance of a PCB communication timeout timer.
    If Not (PCBCommTimer Is Nothing) Then
        PCBCommTimer.Dispose()
    End If
    ' Initialize a new instance of the PCB communication timeout timer.
    PCBCommError = False
    PCBCommTimer = New Timers.Timer
    PCBCommTimer.Interval = CType(CommTO, Double)
    ' Start the PCB communication timeout timer.
    PCBCommTimer.Start()
    ' Wait for PCB ready: D-sub pin 10 will be low.
    Do While True
        If (PCBCommError = True) Then
            ' The PCB never got ready to accept an interrupt.  Therefore, assert
            ' low the interrupt output line, dispose of this instance of the
            ' timer and return an error string.
            AssertDsubPin1Low()
            PCBCommTimer.Dispose()
            SendNibble = "PCB is not ready with pin 10 low."
            Exit Function
        End If
        If (ReadDsubPin10() = False) Then
            ' The PCB is ready to accept an interrupt if D-sub pin 10 is low.
```

```
            Exit Do
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
    ' Interrupt the PCB by setting D-sub pin 1 high.
    SetDsubPin1High()
    ' Wait until the PCB acknowledges the interrupt by setting D-sub pin 10 high.
    Do While True
        If (PCBCommError = True) Then
            ' Communication with the PCB has timed out.  Therefore, assert low
            ' the interrupt output line, dispose of this instance of the timer and
            ' return an error string.
            AssertDsubPin1Low()
            PCBCommTimer.Dispose()
            SendNibble = "PCB did not set pin 10 high."
            Exit Function
        End If
        If (ReadDsubPin10() = True) Then
            ' The PCB has set the Interrupt Acknowledge line high.
            Exit Do
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
    ' Configure the data bus for output.
    ConfigureDataBusForOutput()
    ' Present the low-order nibble of dataout on the data bus.
    Out(DataRegAddress, dataout And &HF)
    ' Delay 1ms for circuits to settle.
    Threading.Thread.Sleep(1)
    ' Assert the Interrupt Request line low.
    AssertDsubPin1Low()
    ' Wait until the PCB asserts the Interrupt Acknowledge line low.
    Do While True
        If (PCBCommError = True) Then
            ' Communication with the PCB has timed out.  Therefore, assert
            ' low the interrupt output line, dispose of this instance of
            ' the timer and return an error string.
            AssertDsubPin1Low()
            PCBCommTimer.Dispose()
            SendNibble = "PCB did not assert pin 10 low."
            Exit Function
        End If
        If (ReadDsubPin10() = False) Then
            ' The PCB has asserted the Interrupt Acknowledge line low.
            Exit Do
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
    ' Dispose of the timer.
    PCBCommTimer.Dispose()
    ' Re-configure the data bus for input.
    ConfigureDataBusForInput()
    ' Report successful communication.
    SendNibble = "SUCCESS"
End Function
```

```
Private Function ReceiveNibble() As Int32
    ' Returns an integer in the range 0 - 15, representing the four bits of data
    ' sent by the PCB.  Returns -1 if the communication fails.  This function
    ' called when the Host PC recognizes that D-sub pin 10 has gone high.
    ' Dispose of any prior instance of a PCB communication timeout timer.
    If Not (PCBCommTimer Is Nothing) Then
        PCBCommTimer.Dispose()
    End If
    ' Initialize a new instance of the PCB communication timeout timer.
    PCBCommError = False
    PCBCommTimer = New Timers.Timer
    PCBCommTimer.Interval = CType(CommTO, Double)
    ' Start the PCB communication timeout timer.
    PCBCommTimer.Start()
    ' Configure the data bus for input.
    ConfigureDataBusForInput()
    ' Acknowledge the interrupt by setting D-sub pin 1 high.
    SetDsubPin1High()
    ' Wait until the PCB asserts D-sub pin 10 low.
    Do While True
        If (PCBCommError = True) Then
            ' Communication with the PCB has timed out.  Therefore, assert
            ' low the interrupt output line, dispose of this instance of
            ' the timer and return an error code.
            AssertDsubPin1Low()
            PCBCommTimer.Dispose()
            ReceiveNibble = -1
            Exit Function
        End If
        If (ReadDsubPin10() = False) Then
            ' The PCB has asserted the Interrupt Request line low.
            Exit Do
        End If
        ' Give other event-handlers a chance to work.
        Application.DoEvents()
    Loop
    ' Read the data now present on the data bus, but keep only the low-order nibble.
    ReceiveNibble = Inp(DataRegAddress) And &HF
    ' Conclude the interrupt by asserting D-sub pin 1 low.
    AssertDsubPin1Low()
    ' Dispose of the timer.
    PCBCommTimer.Dispose()
End Function

Private Sub ConfigureDataBusForInput()
    ' Set ControlReg<5> = 1 to use DataReg for input.
    ControlRegData = ControlRegData Or &H20
    Out(ControlRegAddress, ControlRegData)
    ' Delay 1ms for circuit to settle.
    Threading.Thread.Sleep(1)
End Sub

Private Sub ConfigureDataBusForOutput()
    ' Set ControlReg<5> = 0 to use DataReg for output.
    ControlRegData = ControlRegData And &HDF
    Out(ControlRegAddress, ControlRegData)
    ' Delay 1ms for circuit to settle.
```

```
        Threading.Thread.Sleep(1)
    End Sub

    Private Function ReadDsubPin10() As Boolean
        ' Returns True if the PCB is sending an Interrupt Request, which
        ' it does by setting D-sub pin 10 high.
        ' D-sub pin 10 is StatusReg<6>.
        If ((Inp(StatusRegAddress) And &H40) = &H40) Then
            ReadDsubPin10 = True
        Else
            ReadDsubPin10 = False
        End If
    End Function

    Private Sub SetDsubPin1High()
        ' Sends an Interrupt Acknowledge to the PCB: set D-sub pin 1 high.
        ' D-sub pin 1 is ControlReg<0> (inverted).
        ControlRegData = ControlRegData And &HFE
        Out(ControlRegAddress, ControlRegData)
    End Sub

    Private Sub AssertDsubPin1Low()
        ' Ends an Interrupt Acknowledge to the PCB: assert D-sub pin 1 low.
        ' D-sub pin 1 is ControlReg<0> (inverted).
        ControlRegData = ControlRegData Or &H1
        Out(ControlRegAddress, ControlRegData)
    End Sub

    '////////////////////////////////////////////////////////////////////////////
    '////////////////////////////////////////////////////////////////////////////
    '// Background timer for communication with PCB

    Private WithEvents PCBCommTimer As New System.Timers.Timer

    Private Sub PCBCommTimer_Elapsed( _
        ByVal sender As Object, _
        ByVal e As System.Timers.ElapsedEventArgs) Handles PCBCommTimer.Elapsed
        PCBCommError = True
    End Sub

    '////////////////////////////////////////////////////////////////////////////
    '////////////////////////////////////////////////////////////////////////////
    '// Functions from parallel port library inpout32.dll

    Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" _
        (ByVal PortAddress As Int32, ByVal Value As Int32)

    Private Declare Function Inp Lib "inpout32.dll" Alias "Inp32" _
        (ByVal PortAddress As Int32) As Int32

End Class
```