ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

USING THE 488-PCI and PCM CARDS IN A NON_CONTROLLER APPLICATION

INTRODUCTION

This application note describes how ICS's Model 488-PCI and 488-PCM cards can be used in a non-controller application. Such an application lets a PC mimic a printer or another GPIB device or lets it snoop on the GPIB bus and listen to the data being sent to a particular device.

THE PROBLEM

Normally GPIB Controller Cards in PCs are used as the System Controller. Therefore most example programs are written that way and show how to use the GPIB Controller Cards to read and write data, to serial poll a device or to otherwise control GPIB devices. The steps that have to be taken to use the GPIB Cards as a device are simple but not obvious. The goal of this application note is to show one way the GPIB Cards can be programed as a device and used to transfer data strings over the GPIB bus.

THE EXAMPLE SOLUTION

The Application Note describes a way the 488-PCI or 488-PCM GPIB Controller Cards can be programmed as a device and used to transfer data strings over the GPIB Bus. The Non-controller example program is written in Visual Basic but can be easily adapted to C or C++. The program makes use of the NI ib type commands to control the GPIB Card.

To test the program, connect two PCs together. The ICS's GPIBkybd program on the PC that will be the GPIB Controller. Run the Non-Controller Example program on the PC that is to be the device.

When the Non-Controller Example program first runs, the Form_Load routine initializes the GPIB card as a Non-Controller and assigns it the default GPIB address of 4. The user can change the address by entering another GPIB address value from 1 to 30 in the Device Address window and clicking the SET button. Use the FindLstn button on the Controller PC to verify your address setting.

Data is received from the GPIB bus by pressing the READ button. The program then waits for the GPIB Controller to address the GPIB Card and write data to it. The data is displayed in the Input Message window. Use the GPIBkybd program on the GPIB Controller to send the data string.

Data or a response message is sent to the GPIB bus by entering the information in the Response Message window. Click the appropriate check boxes to select the message terminators. Press the Send button to send the data. On the GPIB Controller use the GPIBkybd Read Response button to read the data from the Non-controller PC. If the AutoSend box is checked, the Dev_example program will automatically send the Response Message each time it receives a message from the GPIB bus.

THE DEVICE EXAMPLE PROGRAM

The Visual Basic code for the Device Example program is listed in Figure 1. The complete program can be downloaded from ICS's website at http://www.icselect.com. The user can use it as a starting point for his own application.

The major program components are Form_Load, cmdRead and cmdSend. Input data can be taken from InBuf which is sized for up to 1000 character. Output data should be placed in the OutMsg string. Some program features such as GPIB address change or adding the carriage return termination may not be needed and can be omitted in the user's program. Other features like notification when the GPIB Card is addressed to talk or listen would be nice to include in the user's application.

```
‘************************************************************************************
‘   Visual Basic GPIBkybd2 Program                                         07-24-02
‘   Copyright 2002 ICS Electronics div Systems West, Inc.
‘
‘   Program makes the GPIB board a device to receive and send strings
‘   Program controls the GPIB-32.dll with NI type ib calls
‘************************************************************************************
Dim CmdStr As String
Dim OutFlag
Public NL
Public Msg_Format$
Public Cmd$
Public FirstTimeFlg
Public Curaddr%
Public Device                    ‘ppss address form
Public GPIB_BD%
Public Er%
Public vtmo                      ‘current Timeout Setting
Const winPictureBox = 2016002
Const winCommandButton = 2007557
Dim Outbuf As String * 1000
Dim Inbuf As String * 1000
Dim GPIB_Intfc As String

Private Sub cmdExit_Click()
   Call ibonl(GPIB_BD%, 0)          ‘remove the device.
   End
End Sub

Private Sub cmdRead_Click()
    txtError.Visible = False
    txtInputMsg.Text = “”
    Call ibrd(GPIB_BD%, Inbuf)
    txtInputMsg.Text = Inbuf
    txtInputMsg.Text = RTrim$(txtInputMsg.Text)
    txtInputMsg.Text = txtInputMsg.Text + NL + Str$(ibcntl) + “ bytes received”
    If ckAutoSend.value = 1 Then
       Call cmdSend_Click
   End If
End Sub

Private Sub cmdSend_Click()
   txtError.Visible = False
   Outmsg$ = txtRespMsg.Text
   L = InStr(Outmsg$, Chr$(10))
   If L > 1 Then                                ‘if linefeed found
      Outmsg$ = Left$(Outmsg$, L - 1)        ‘reduce to original string
   End If
   If ckCR.value = 1 Then
      Outmsg$ = Outmsg$ + Chr$(13)
   End If
   If ckLF.value = 1 Then
      Outmsg$ = Outmsg$ + Chr$(10)
   End If
   Call ibwrt(GPIB_BD%, Outmsg$)
   txtRespMsg.Text = txtRespMsg.Text + NL + Str$(ibcntl) + “ bytes sent”
End Sub
```

Figure 1   Device Example Program

```
Private Sub cmdSet_Click()                    ‘sets board GPIB address from 1 to 30
 NewAddr% = Val(txtAddr.Text)
    If NewAddr% <> Curaddr% Then
        If (NewAddr% >= 1) And (NewAddr% <= 30) Then
            pad% = Val(txtAddr.Text)
            sad% = 0
            nisad% = 0
            dev% = pad%
             Call ibpad(GPIB_BD%, NewAddr%)
             Curaddr% = NewAddr%
        Else
            Beep
            txtResults.Text = “Device Address outside of 1 to 30 range, please reenter”
        End If
        Device = Val(txtAddr.Text)                              ‘Device%=ppss format
        txtError.Text = “Board GPIB Address => “ + Str$(Device)
    Else
        Beep
        txtError.Visible = True
        txtError.Text = “New Device address same as current address,  please reenter”
    End If
End Sub

Private Sub Form_Load()
    NL = Chr(13) + Chr(10)
    SPACE80S = Space$(100)                          ‘100 spaces
    Rev$ = “Revised 07-24-2002”
    txtRev.Text = Rev$
    txtError.Text = “”                                ‘clear label and text box
    txtError.Visible = False
    txtInputMsg.Text = “”
    OutFlag = 0
    Buf_length = 1000
    BD% = 0                                           ‘define initial values
    dev% = 4
    bddev% = 0
    addr% = 4
    Device = 4
    GPIB_Intfc = “GPIB0”
    txtAddr.Text = “4”
    cmdExit.Enabled = True
    vtmo = T3s                                        ‘defaut to 3 second timeout

    ckCR.value = 0
    ckLF.value = 1
    ckAutoSend.value = 0
    txtError.Enabled = False
    ErrFlag = 0

    FirstTimeFlg = 0                                  ‘GPIB card initialization
    Call ibfind(GPIB_Intfc, GPIB_BD%)                          ‘get board handle
    If (ibsta% And EERR) Then                              ‘set GPIB address
        Call gpiberr(“ibfind error”)
        txtError.Text = RetMsg$
        txtError.Visible = True
        GoTo Formexit
    End If
```

Figure 1   Device Example Program Listing continued

```
    Call ibrsc(GPIB_BD%, 0)                          ‘make a non controller
    If (ibsta% And EERR) Then                        ‘set GPIB address
        Call gpiberr(“ibrsc error”)
        txtError.Text = RetMsg$
        txtError.Visible = True
        GoTo Formexit
    End If

    Curaddr% = Val(txtAddr.Text)
    Call ibpad(GPIB_BD%, Curaddr%)
    If (ibsta% And EERR) Then                        ‘set GPIB address
        Call gpiberr(“ibpad error”)
        txtError.Text = RetMsg$
        txtError.Visible = True
        GoTo Formexit
    End If

    cmdSet.Enabled = True
    cmdSend.Enabled = True
    cmdRead.Enabled = True
    FirstTimeFlg = 1
Formexit:
End Sub
```

Figure 1   Device Example Program Listing continued