AB48-37

ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

File Transfers over the GPIB Bus

INTRODUCTION

The normal practice in a GPIB test program is to write or read short data strings to or from an instrument. However in some cases, the user may need to transfer large blocks of data to the instrument. Examples are Waveform Generators which need many bytes of instructions to specify the waveforms the user wants to generate or Data Acquisition Systems which collect large amounts of data. This Application Note describes the correct way to make file transfers with ICS's 488.2 GPIB Controllers and includes a sample Visual Basic language program.

READING DATA INTO A FILE

Data can be easily read into a file by doing the following: First create a blank file and save it in the directory with your program. This simplifies the filename in the *ibrdf* command. You can create the file ahead of time with Wordpad or Notepad or create it in your program. Save the file as a .txt. file. Then use an *ibrdf* command to read the data from the instrument into the file.

An example instruction is:

```
ibrdf(ud%, filename$)
```

where ud% is the device handle and filename$ is the name of the file that will be receiving the data. If the file is in a different directory, filename$ will need to show the full path to the file. i.e. "c:\new_directory\read_file.txt"

Follow *ibrdf* with a error checking routine that checks *ibsta* for errors to be sure that the file transfer was done correctly. If *ibsta* is < 0x8000 hex then the file transfer was okay. The variable *ibcnt* will contain the number of bytes transferred to your file.

WRITING FILE DATA TO A DEVICE

File data can be easily transferred to a device by doing the following: Create a file with the desired information. You can manually create the file with Wordpad or do it in your program. Save the file as a .txt file. Once the file is created and saved, use an *ibwrtf* instruction to transfer the file data to the instrument. EOI will be asserted by the GPIB Controller when the last byte of data is transferred to the device.

An example instruction is:

```
ibwrtf(ud%, filename$)
```

where ud% is the device handle and filename$ is the name of the file that contains the data to be transferred to the instrument. If the file is in a different directory, filename$ will need to show the full path to the file. i.e. "c:\new_directory\read_file.txt"

Follow *ibwrtf* with a error checking routine that checks *ibsta* for errors to be sure that the file transfer was done correctly. If *ibsta* is < 0x8000 hex then the file transfer was okay. The variable *ibcnt* will contain the number of bytes transferred to the device.

EXAMPLE VISUAL BASIC PROGRAM

VB_File_Xfr_Demo is a Visual Basic sample program that shows how to do simple file transfers. The VB_File_Xfr_Demo program is a simplified version of ICS's GPIBkybd program and shows a user how to do a number of basic tasks that he or she is likely to encounter in a real program. VB_File_Xfr_Demo includes two new controls for transferring files to or from a GPIB device.

In the VB_File_Xfr_Demo program, the user enters his filename in the filename text box and presses the Read File or Write File button. The files have to be created ahead of time when using the VB_File_Xfr_Demo program as the demo program does not have a file creating capability. If the file is in the same directory as the program, the user enters just the file name with its extension. If the file is located in a different directory, the user must enter the full filename.

The VB_File_Xfr_Demo.zip file from ICS's website, contains all of the Visual Basic source file, an executable file, the VB libraries and two sample files, testrd.txt and testwrt.txt. Download and expand the zipped files into a temporary directory. Double-click on VB_File_Xfr_Demo.exe to run the demo program. If you do not have Visual Basic or Visual Studio 6 installed, you will need to download the VB6 Runtime files from ICS's website before you run the program. (http://www.icselect.com/prgupdates.html)

Sample file testwrt.txt contains the IEEE_488.2 IDN query. If testwrt.txt is written to a 488.2 compatible device, the device will respond by writing its IDN message into the testrd.txt file. To do this, connect your GPIB Controller to an IEEE-488.2 compatible Instrument and run the VB_File_Xfr_Demo.exe program. Be sure the device's GPIB address is in the Device Address window at the top of the VB Form. Verify device communication by doing an *ESR? or *IDN? query and reading back the correct response. Then type testwrt.txt into the Write File test box and click the Write file button. Next type testrd.txt into the Read File text box and click the Read File button. Use Wordpad, Notepad or any text editor to view the testrd file.

Use the cmdWrtf and cmdRdf subroutines in the VB_File_Xfr_Demo program as starting points for your own file transfer program. These two subroutines are listed on the right in Figures 1 and 2.

SUMMARY

This Application Note has shown how a user can do file transfers to and from a GPIB device. Figure 1 on the right is a Visual Basic language example of a file transfer to an instrument. Figure 2 is a Visual Basic language example of a file transfer from an instrument. Both examples and the VB_File_Xfr_Demo program can be used directly with ICS's and any other GPIB Controllers that support the National Instrument 'ib' and 488.2 command sets.

```
Private Sub cmdWrtf_Click()
   txtError.Text = “”
   txtError.Visible = False
   If txtWrtFilename.Text = “” Then
      txtError.Text = “Missing source filename”
      txtError.Visible = True
      Beep
   Else
      wfilename$ = txtWrtFilename.Text
      Call ibwrtf(ud%, wfilename$)
      If (ibsta% And EERR) Then
         Call gpiberr(“ibwrtf Error”)
         txtError.Text = RetMsg$
         txtError.Visible = True
         Beep
      Else
         txtResults.Text = Str$(ibcntl) & “ bytes written to device
“ & Str$(Device)
      End If
   End If
End Sub
```

Figure 1   Write File Example

```
Private Sub cmdRdf_Click()
   txtError.Text = “”
   txtError.Visible = False
   If txtRdFilename.Text = “” Then
      txtError.Text = “Missing destination filename”
      txtError.Visible = True
      Beep
   Else
      rfilename$ = txtRdFilename.Text
      Call ibrdf(ud%, rfilename$)
      If (ibsta% And EERR) Then
         Call gpiberr(“ibrdf Error”)
         txtError.Text = RetMsg$
         txtError.Visible = True
         Beep
      Else
         txtResults.Text = Str$(ibcntl) & “ bytes read from de-
vice “ & Str$(Device)
      End If
   End If
End Sub
```

Figure 2   Read File Example