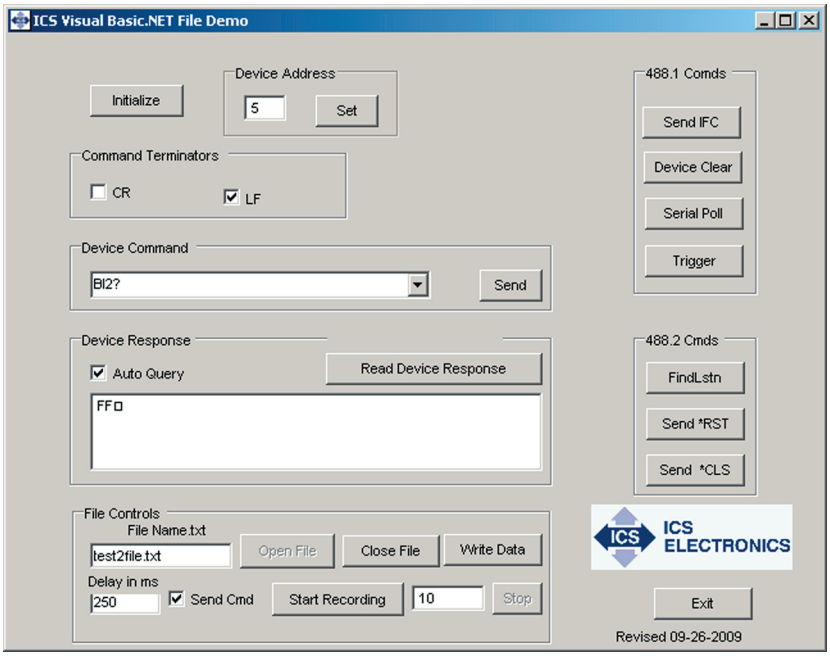
ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

VB.NET Programming Example for GPIB Controllers with File Handling Functions

INTRODUCTION

This Application Bulletin shows how to create a Visual Basic.NET program with file handling capability for GPIB Controllers that utilize an industry standard GPIB-32.DLL driver library. The user can take the different pieces of the FileDemo program and copy them into his program. This FileDemo Program was developed with Microsoft's Visual Studio 2005 and ICS Electronics GPIB-32.DLL. Note that earlier versions of Microsoft's .NET development systems are not compatible with ICS's included library files. While this program has only been tested with ICS's GPIB Controllers, there is no reason why it should not work on other GPIB Controllers that include a GPIB-32.DLL Driver Library.



Visual Basic.NET FileDemo Main Form

GENERAL CONCEPTS

This program builds on the basic VB.NET Demo program described in Application Bulletin AB48-39 by adding file handling capabilities to the main form. The Visual Basic.NET FileDemo Program was written as an interactive control program that the user can easily adapt to his or her use. The FileDemo Program has just the one form shown in the figure on the right.

The original Demo program form has text windows where the user can set the address of the GPIB device, enter a command string to send to the device, and to display device responses. Other common GPIB functions like Send IFC, Device Clear, Serial Poll and Trigger are controlled by separate buttons. The FileDemo Program also includes the IEEE-488.2 FindLstn routine and buttons for sending the *RST and *CLS commands to the GPIB device.

The new File Controls are located below the Device Response frame and let a user create or open a text file, append data lines to the file and close the file. Collecting data usually requires the user to issue a command to the GPIB device, read data from the device and then write it to the file. The Start Record and Stop buttons automate this process by repeating it on a user set time interval.

The Device Command window was changed to a combo box with a pulldown arrow. During the Form load time, the box is preloaded with three IEEE_488.2 Common Commands to reduce the typing load on the user. The FindLstn function was added to the initialization routine to make the program work easier with the user's devices.

REQUIREMENTS

The executable version of the FileDemo Program can be run on any computer that has Microsoft's Visual Studio 2005 or Microsoft's .NET Framework, version 2.0 or later, installed on it. Visual Studio 2005 or 2008 is required to develop your own Visual Basic.NET program. Any new program must include ICS's GPIB-32.VB and ICSVB.VB files to link to the GPIB-32.DLL Driver Library.

RUNNING THE DEMO PROGRAM

When the FileDemo Program loads, only the Initialize button is enabled. This forces the user to initialize the GPIB Controller and the GPIB bus. The initialization routine also finds all of the GPIB devices on the GPIB bus and sets the Device address window to the first device found. Once the GPIB Controller is initialized, the remaining controls on the form are enabled. The GPIB Controler must be initialized in your program for correct operation.

The FileDemo program is designed to only control one GPIB device at a time. The address of the current device being controlled is shown in the Device Address window. The Device Address is set to the lowest GPIB address found on the GPIB bus during initialization. If you have multiple devices, you will have to enter the GPIB address of the device you want to control into the Device Address window and press Set to select a different device.

Device commands and text data can be entered into the Device command window. The FileDemo program will append a linefeed to the command strings and output them to the GPIB device when you press the Send button. The FileDemo program will automatically read the device responses if the command string contains a '?' character and if the Auto Query box is checked. You can manually read the device response at any time by pressing the Read Device response button.

GPIB commands (IEEE-488.1 multiline commands) such as GET or Device Clear aer not ASCII character commands and cannot be entered into the Device Command window. Instead they are sent to the device by clicking on one of the buttons in the 488.1 Command frame. The 488.2 Commands frame includes the 488.2 FindLstn protocol and two buttons for sending the *RST and *CLS commands. These two commands could have been included in the Device Command combo box pulldown command list.

DEMO PROGRAM CODE

The complete FileDemo Program can be downloaded as a ZIP file from ICS's website. The source for the FileDemo Program is contained in the VB_FileDemo.vb file which can be opened in Visual Studio or in any text editor.

Visual Basic.NET is more rigorous that the older Visual Basic 6 because it requires that you preface all references to the library with a class name. However, it also provides automatic indenting and error checking as you write your program that is a big help when you are writing large functions.

The Form1 Load function only initializes a few variables and then disables all controls except Initialize and Exit. Exit must never be disabled.

The Initialize function outputs an IFC to clear the GPIB bus, gets the Controller's GPIB address, asserts REN, sets the timeout to 3 seconds and calls the FindLstn function to set the Device Address. Ibsta is checked for an error after each GPIB call. All GPIB programs should explicitly initialize the GPIB Controller and the GPIB bus as a matter of good programming practise.

The cmdSet function is more complicated in the FileDemo Program than you need in a real program because of checks to verify that the user did not enter the GPIB Controller's address and to handle both primary and secondary GPIB addresses. cmdSet also checks for presence of a GPIB device at the selected address when the Set button is clicked. If you use fixed GPIB addresses, you can assign a GPIB address to a descriptive variable, e.g. dvm = 04 at the beginning of your program.

The 488.1 commands are done by calling the appropriate function from the GPIB-32.VB library. Note that some instruments that use SCPI commands need to be enabled before they will respond to a Trigger command.

The cmdFindLstn function is the most complicated function and is included in the FileDemo Program because it is very useful as a way to verify that all of the GPIB devices in your test system are present. The FileDemo Program creates a list of all possible GPIB addresses to test before calling FindLstn and returns a list of all active devices on the GPIB bus. Once you have the Active Device List, you can use an *IDN? query to create a table of GPIB devices vs their addresses. You can also verify that all of the GPIB devices that you expect are connected and are set to their correct GPIB addresses. This is very helpful for eliminating problems when equipment has been taken away for calibration, not turned on or returned with the wrong GPIB address.

The cmdSend function sets the desired termination condition and then calls the 488.2 Send function to output the command string. If the command string was a query (contains a '?') and if AutoQuery is selected, cmdRead is called. cmdRead inputs the device response string and places it in the txtResults text box.

The File Handling controls let the user enter a file name (*filename.txt*) into the File Name Window. cmdOpen opens the file and the .NET streamwriter function (*sw*) for appending lines to it. If the file is not found in the current directory, a new one is created. cmdWrtLine simply writes the current contents of the Device Response window as a new line to the file. cmdClose closes the streamwriter function and clears the FileOpenFlg. cmdExit also checks the FileOpenFlg and closes the file if it is still open.

The Start Record and Stop buttons automate the recording process by invoking the .NET Timer function. A .NET Timer is added to the form as Timer1. When Timer1 runs, it checks the RecordFlg when the time interval has expired. If the flag is set, it then checks the ckSendCmd checkbox to see if it should send the command in the Device Command window to the GPIB device. (Note that this will not read data if the Device Command does not contain a '?' character as it depends upon the AutoQuery function to call cmdRead). If the ckSendCmd checkbox is not checked, the Timer1 routine just reads data from the device. The data is then written to the file and the count in the RecordCnt window is incremented.

cmdStartRecord checks to see it the Time Delay window value is valid. If the value is > 100 ms, cmdStartRecord sets the Timer1 interval to the Time Delay window value, enables the Timer, sets the RecordFlg, enables the Stop button and disables the Start Record button. This gives the user a visible indication that the recording process is enabled. cmdStopRecord resets the RecordFlg, enables the Start Record button and disables the Stop button.

SUMMARY

This application note has described a Visual Basic.NET program that demonstrates how to use the most common GPIB commands, how to open a file and how to save the test data to the file. The user can build on this example by adding insrument setup commands and expand the recorded data. Multiple data items can be written on the same line as long as they are separated by a comma or a tab for easy conversion into an Excel file. The user should be able to use the FileDemo Program as a framework to build a complete test application.

PROGRMMING HINT

When writing a test program, it is often handy to create a subroutine that combines cmdSend and cmdReceive as a single subroutine to reduce your coding effort. The subroutine should output the command string, check for a question mark, and if one is present read the device response. It also can do any necessary string cleanup such as removing trailing spaces and linefeeds and return the cleaned response string to the calling program. An example of its use is:

```
CmdStr$ = "*ESR?"
Call Sout(dvm, Cmdstr$)
If Val (Rdg$) <> 0 then.....
```

Otherwise this would be:

```
CmdStr$ = "*ESR?"
eotmode = GPIB32Funcs.NLend
Call Send(Bd, dvm, CmdStr$, eotmode)
If (GPIB32Funcs.ibsta And GPIB32Funcs.EERR) Then
    Call gpiberr("Send error")
    txtError.Text = RetMsg
    txtError.Visible = True
End If

Rdg$ = Space (80)
Call Receive(Bd, dvm, Rdg$, StopEnd)
If (GPIB32Funcs.ibsta And GPIB32Funcs.EERR) Then
    Call gpiberr("Receive Error")
    txtError.Text = RetMsg
    txtError.Visible = True
Else
    txtError.Visible = False
    txtResults.Text = RTrim(Instring)
    Rdg$ = RTrim(Instring)
End If

If Val(Rdg$) <> 0 then.....
```