Session 2559

GPIB Instrument Control

Leonard Sokoloff
DeVry College of Technology

Abstract

Virtual Instrumentation is an important technology that is making a significant impact in today's industry, education and research. Virtual instrument software can be used for simulation, and with appropriate interfacing, it can also be applied to data acquisition and control. There are a number virtual instrument packages available commercially but, for the application described in this abstract, LabVIEW, a product of National Instruments Corporation has been used.

This paper describes GPIB instrument control and data processing. Among its many features, LabVIEW also supports the IEEE-488 standard. The functional capability of many commercially available instrument drivers is often much greater than one requires in a specific instrument control application. Any instrument that has a GPIB capability includes in its manual a listing of GPIB commands. These commands can be used in LabVIEW environment to build a custom instrument driver whose functionality satisfies individual needs. As needs change, these drivers can be easily modified and expanded.

An additional benefit of the custom tailored instrument driver is the inclusion in the same software package specific data processing tasks that a commercial package will not have. The VI that integrates instrument control and data processing can be modified and expanded with relative ease to include other instruments as well as additional processing tasks. For example, the waveform that an oscilloscope is displaying in process monitoring operation, can be imported into a GPIB VI over the GPIB bus. The data can then be displayed on a waveform graph or stored to a file for future reference. The data can also be applied to immediate signal processing such as displaying its frequency spectrum in order to assess process performance in real time. This paper describes this type of instrument control for three different instruments with data processing.

This paper is a result of my development effort to include virtual instrumentation experiments in the electronics laboratory of the Bachelor program at DeVry College of Technology. We are using LabVIEW and TCP/IP protocol in network applications, data acquisition applications and GPIB instrument control and signal processing application, the subject of this paper.

LabVIEW Software

LabVIEW™ (Laboratory Virtual Instrument Engineering Workbench), a product of National Instruments™, is a powerful software system that accommodates data acquisition, instrument control, data processing and data presentation. LabVIEW that can run on PC under Windows, Sun SPAR stations as well as on Apple Macintosh computers, uses graphical programming language (G language), departing from the traditional high level languages such as the C language, Basic or Pascal.

All LabVIEW graphical programs , called Virtual Instruments or simply VIs, consist of a Front Panel and a Block Diagram. Front Panel contains various controls and indicators while the Block Diagram includes a variety of functions. The functions (icons) are wired inside the Block Diagram where the wires represent the flow of data. The execution of a VI is data dependant which means that a node inside the Block Diagram will execute only if the data is available at each input terminal of that node. By contrast, the execution of a traditional program, such as the C language program, follows the order in which the instructions are written.

LabVIEW incorporates data acquisition, analysis and presentation into one system. For acquiring data and controlling instruments, LabVIEW supports IEEE-488 (GPIB) and RS-232 protocols as well as other D/A and A/D and digital I/O interface boards. The Analysis Library offers the user a comprehensive array of resources for signal processing, filtering, statistical analysis, linear algebra operations and many others. LabVIEW also supports the TCP/IP protocol for exchanging data between the server and the client. LabVIEW v.5 also supports Active X Control allowing the user to control a Web Browser object.

GPIB Interface

GPIB (General Purpose Interface Bus) interface has its origin in the HPIB (Hewlett Packard Interface Bus) developed by Hewlett Packard Corporation in the 70's for interfacing their instruments. It gained widespread popularity and soon became a defacto industry standard. In 1975 the IEEE standards group published the standard IEEE-488 for the GPIB interface. Industry uses this standard to this day as a parallel interface for various applications in instrument control.

GPIB interface is a parallel 24 conductor bus. It includes eight data lines for control messages that are often ASCCI encoded, and various management and handshake lines. Handshaking is used to transfer messages between the PC and the instrument being controlled.

Fig .1 shows a typical connection between the PC with the GPIB interface board and the software for controlling the operation of an instrument. Not every instrument has a GPIB capability. This capability is designed into the instrument by the manufacturer and a GPIB connector is provided on the instrument housing. The cable used to connect the instrument must also be GPIB compatible and meet the IEEE-488 protocol.

As shown in Fig. 1, the operation of several instruments may be controlled. Consequently each instrument must have a unique address between 0 and 30, address 0 is usually reserved for the GPIB interface board. National Instrument Corporation has an array GPIB interface boards with different capabilities. As shown in Fig. 1, AT-GPIB interface board is used in this paper. It has a maximum data transfer rate of 1 Mbps.

The NI-488.2 is the driver software for the GPIB interface. GPIB devices can be Talkers, Listeners, or Controllers. The Listeners receives the messages while the Talker sends the messages. The Controller, usually the PC where the GPIB board and NI.488.2 software are installed, manages the flow of commands or messages on the GPIB bus. The communication between the PC and the instrument is message based. ASCCI encoded message strings are transferred using handshaking.

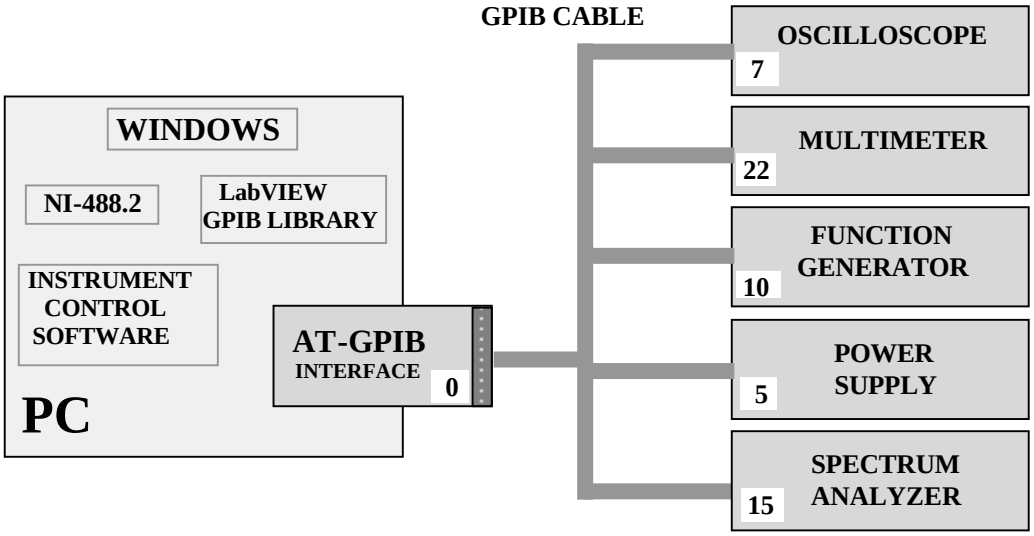
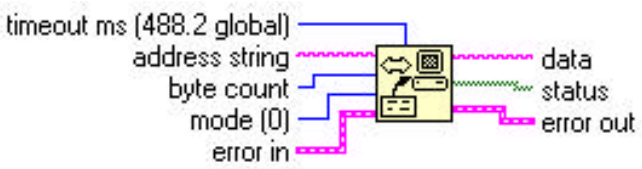


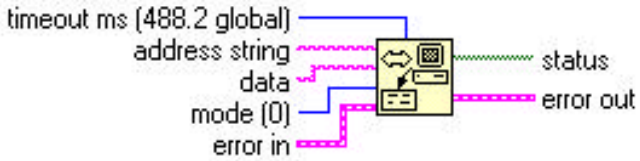
Fig. 1 GPIB Instrument Control Setup. Each Instrument and the GPIB Interface Has a Unique Address

Functions>Instrument I/O>GPIB is the location of available GPIB VIs for building an instrument control VI, the VI that is custom tailored to meet user exact needs. On a small scale this is a driver.

The most used GPIB VIs shown below are the GPIB Read and the GPIB Write VIs. The former reads a data string from the instrument with the specified address and the latter writes a control string to the instrument with the specified address.



The **GPIB Read** VI reads the specified number of bytes from an instrument with the specified address string



The **GPIB Write VI writes the data string to an instrument** with the specified address string

GPIB Instrument Commands

Four instruments are used in the experiment described in this paper: power supply, multimeter, function generator and oscilloscope. If the instrument has GPIB capability then the manual for that instrument must include a list of GPIB commands. Table 1 lists all commands for the instruments used in this paper. A typical format for an instrument control is as follows:

Command name: command value
Command name: parameter name: parameter value

For instance to select a square wave from the function generator and to set its voltage to 5 V and its frequency to 500 Hz, the following commands from Table I must be transferred to the instrument at address 10:

SOUR:FUNC: SHAP SQU
SOUR:VOLT 5.0
SOUR:FREQ 500.0

In a similar fashion we can issue commands that will control the operation of other instruments.

Instrument Control Template

The GPIB instrument control VI need not be a commercial driver. In most applications a custom driver that meets the needs of the user is more efficient and faster. Fig. 2 shows a general instrument control template that can be used with any instrument.

Table I Instrument Control Commands

INSTRUMENT	COMMAND NAME	COMMAND VALUE	ACTION
POWER SUPPLY HP E3631A **Address 5**	OUTP	ON	POWER ON
		OFF	POWER OFF
	INST:SEL	P6V	SETS OUTPUT TYPE TO +6V
		P25V	SETS OUTPUT TYPE TO +25V
		N25V	SETS OUTPUT TYPE TO –25V
	VOLT	NUMERIC	SETS OUTPUT VOLTAGE:USER ASSIGNED
	CURR	NUMERIC	SETS OUTPUT CURRENT:USER ASSIGNED
FUNCTION GENERATOR HP 33120A **Address 10**	SOUR:FUNC: SHAP	SIN	SETS OUTPUT TYPE TO SINEWAVE
		SQU	SETS OUTPUT TYPE TO SQUAREWAVE
		TRI	SETS OUTPUT TYPE TO A TRIANGULAR WAVE
		RAMP	SETS OUTPUT TYPE TO RAMP WAVE
	SOUR:FREQ	NUMERIC	SETS OUTPUT FREQUENCY VALUE
	SOUR:VOLT	NUMERIC	SETS OUTPUT VOLTAGE VALUE
MULTIMETER HP34401A **Address 22**	CONF:	VOLT:DC	SETS THE METER TO READ DC VOLTAGE
		CURR:DC	SETS THE METER TO READ DC CURRENT
		VOLT:AC	SETS THE METER TO READ AC VOLTAGE
		CURR:AC	SETS THE METER TO READ AC CURRENT
		RES	SETS THE METER TO READ RESISTANCE
		FREQ	SETS THE METER TO READ FREQUENCY
		CONT	SETS THE METER TO TEST CONTINUITY
		DIOD	SETS THE METER TO TEST DIODE
OSCILLOSCOPE Address 7	READ?	NONE	REFRESHES THE SCREEN
	WAV:DATA?	NONE	ACQUIRES THE DATA AS A STRING

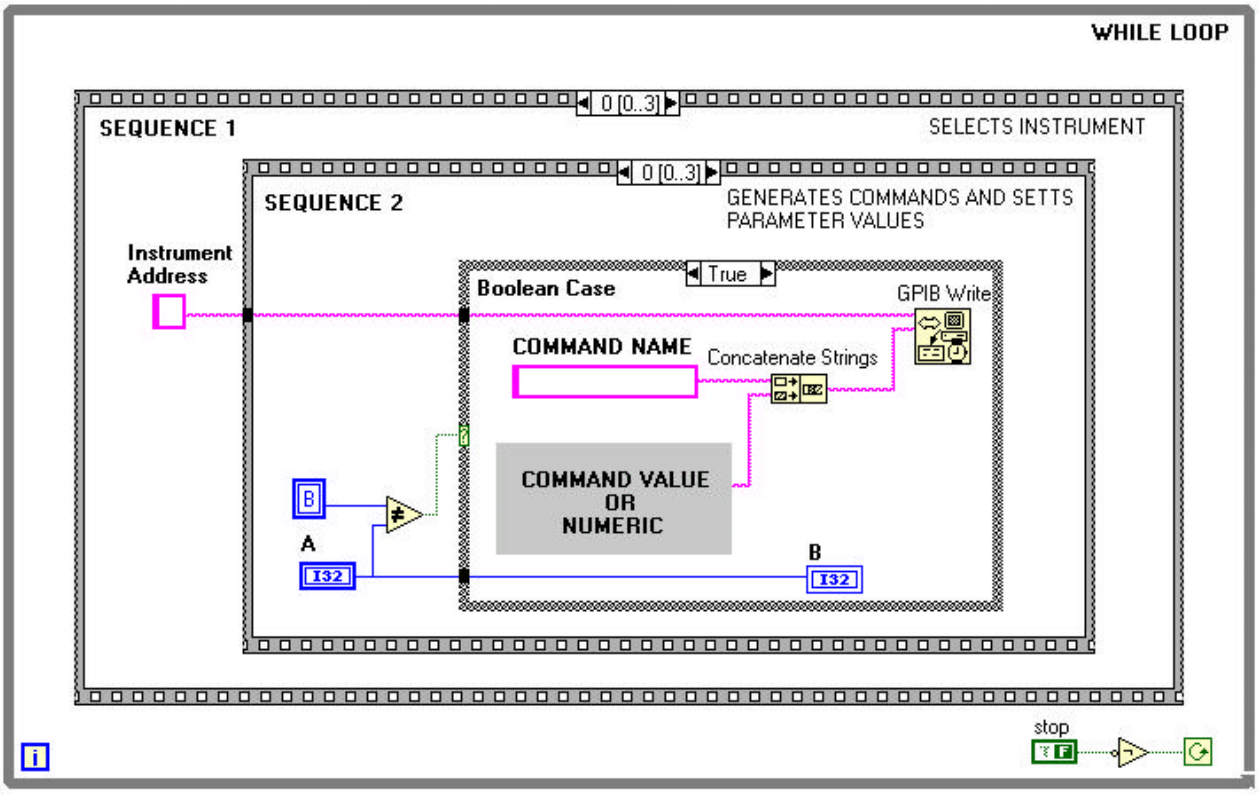


Fig. 2 GPIB Instrument Control Template

The user must know the commands for the instrument that he wants to control. If the instrument is GPIB compatible then the commands can be found in the manual for that instrument.

Fig. 2 shows a general purpose instrument control template that can be adapter to almost any GPIB instrument. It consists of two Sequence structures inside a While Loop. As long as a TRUE is applied to its condition terminal, the code inside will be repeatedly executed. The user must click on the STOP button in the Front Panel to terminate execution.

Sequence 1 structure selects the instrument to be controlled. One frame is required for each instrument.

Sequence 2 structure generates the necessary commands to turn the instrument, select one of its outputs or parameters and adjust the selected parameter to the desired value. Also included in this frame is code that detects whether the user changed any of the Front

Panel Controls. If not, the empty FALSE case will be executed and no message will be sent. However if user changed the setting of one or more of the Front Panel Controls, the TRUE case is executed and commands are generated reflecting the Front Panel changes made by the operator.

The Front Panel control designated by "A" can be Boolean, numeric, or parameter type associated with a vertical slide. Its value is stored in Front Panel digital indicator designated by "B". The value of "A" and the local variable "B" is compared by the NOT EQUAL function outside of the Boolean Case structure whose output is FALSE if the operator did not change the settings of the Front Panel Controls, and if the operator changed some settings, then the output will be TRUE.

Depending on Front Panel settings that changed, the TRUE frame will generate the appropriate commands that reflect those changes. For example, should the operator decide that the power supply output must be changed from +25V type to –25V type and the value of the output must be changed from +12 VDC to –15 VDC then two frames of Sequence 2 structure will be executed, one reflects output type change and the other, the output value change. In each Sequence 2 frame a comparison that was described will be done and the TRUE Boolean case will generate the appropriate code.

Before transmission, all commands must be in the form of strings as strings provide the most flexible format. They can represent alphabetical characters, a numeric characters or almost any character on the keyboard. The transmitted command strings are usually ASCII encoded.

GPIB Instrument Control Software

This section describes the hardware and the software used in the laboratory to run the GPIB instrument control experiment of this paper. Fig. 1 shows the hardware setup. The Front Panel and the Block Diagram of the software, GPIB Instrument Control.vi, are shown in Fig. 3. As can be seen, the Front Panel is sub-divided into sections pertaining to the instruments being controlled. There are five instrument sections: power supply, multimeter, function generator, spectrum analyzer and the oscilloscope. Each section includes controls for selecting instrument functions and for adjustment of the selected function value. For example, the user may select on the vertical slide the square wave from the function generator and adjust its amplitude and frequency on the adjacent digital controls.

In the Block Diagram of Fig.3, the software is inside the While Loop providing the user with an interactive environment. The loop will execute as long as TRUE is applied to its condition terminal. The user can then adjust Front Panel controls to set the instruments to

the desired outputs and adjust their values. To terminate execution, the user must click on the STOP button in the Front Panel.

Due to the repetitive nature of the instrument control software, Fig. 1 shows the Block Diagram details that apply to the power supply in frame 0 of the Sequence 1 structure. The four frames of Sequence 1 structure include the generation of the required commands to control the operation of the power supply.

FRONT PANEL

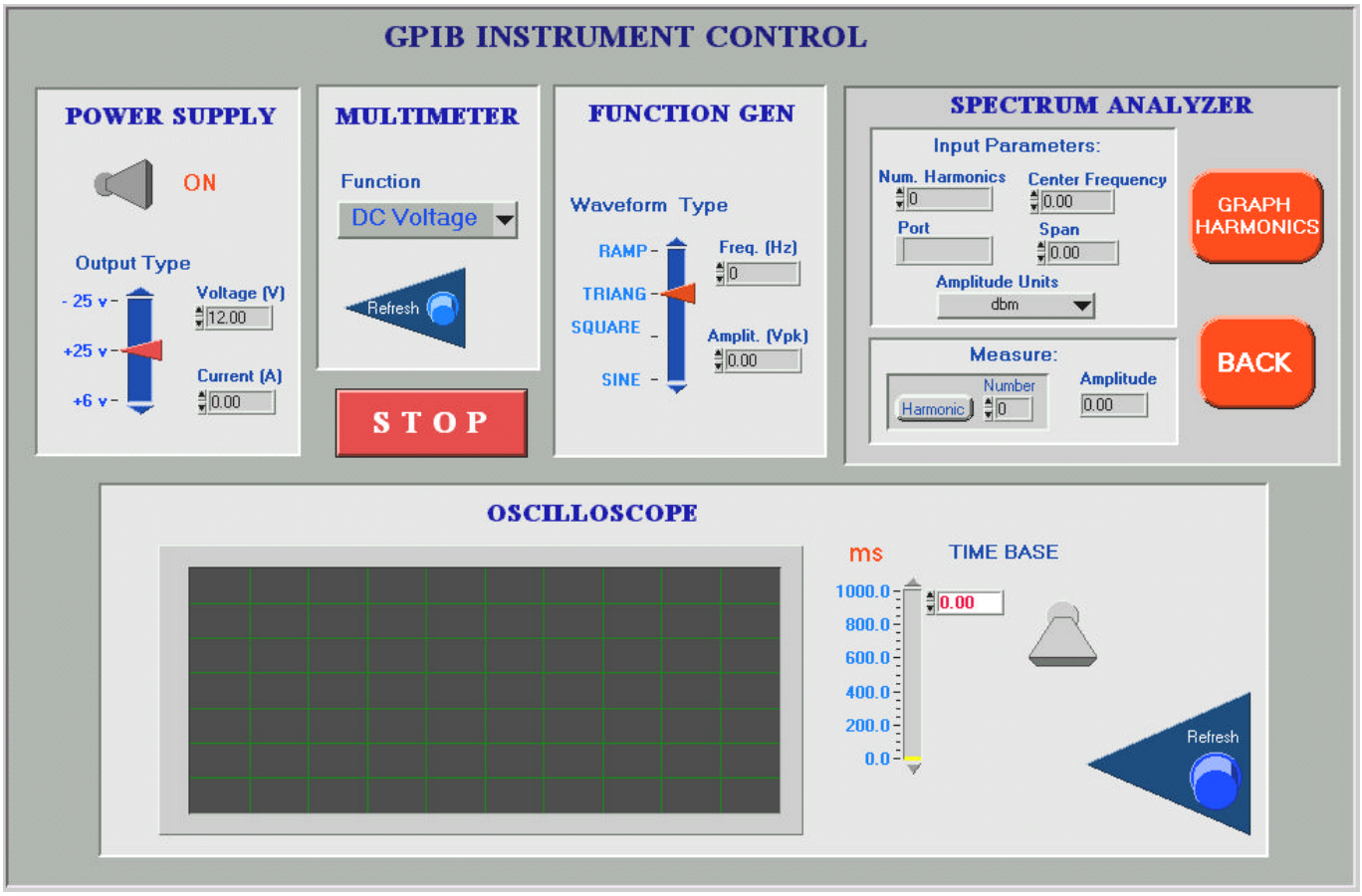


Fig. 3 The Front Panel and the Block Diagram of GPIB Instrument Control.vi

Frame 0 of Sequence 2 checks the state of the power switch. If the user changed the state, Case 2 generates the command OUTP followed by ON or OFF from Case 1 and sends it to the power supply at address 5.

Frame 1 of Sequence 2 checks the state of the power supply output type (6V, +25V, –25V). If the operator changed the state, Case 4 generates the command INST:SEL followed by P6 or P25 or N25 from Case 3 and sends it to the power supply at address 5.

Frame 2 of Sequence 2 checks the state of the output voltage. If the user changed the voltage value then Case 5 generates the command VOLT followed by the new value of voltage from the Front Panel digital control Voltage and sends this command to the power supply at address 5. The decimal form of voltage is converted to a string by To Fractional function.

Frame 3 of Sequence 2 checks the of the current setting. If the user changed the setting, then Case 6 generates the command CURR followed by the new value of current from the digital control Current and sends this command to the power supply at address 5.

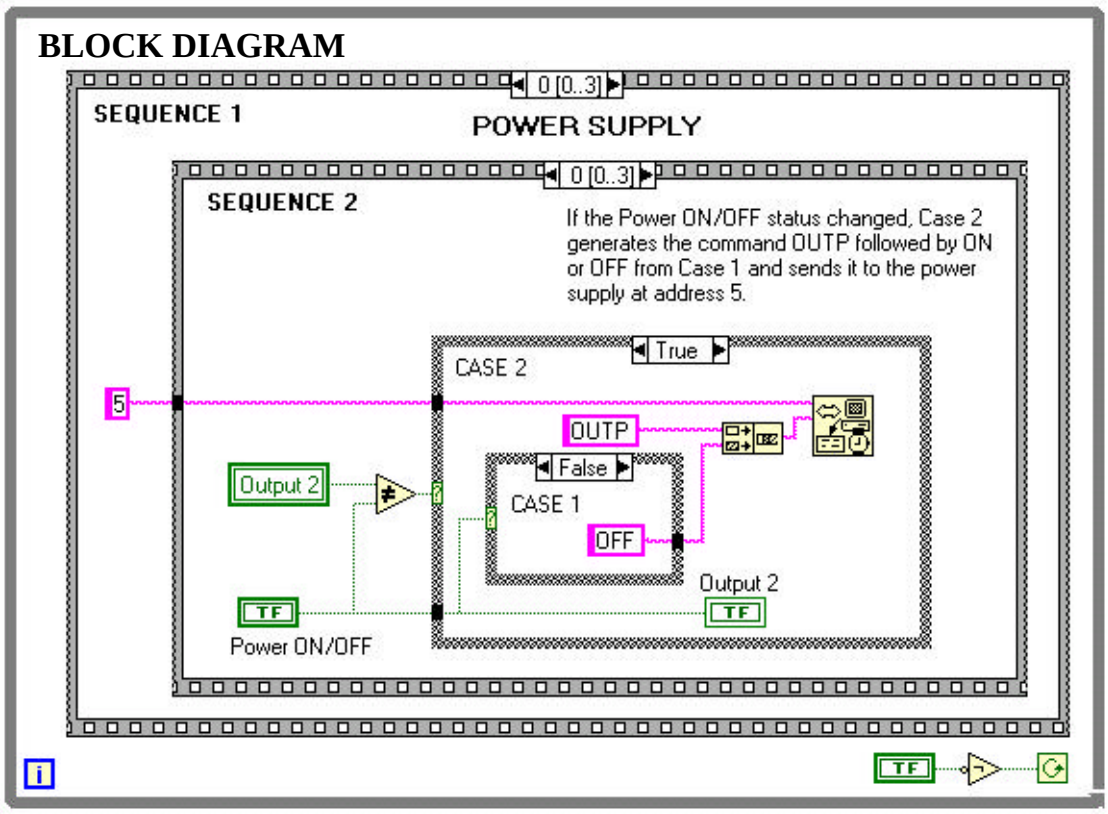


Fig. 3 The Front Panel and the Block Diagram of GPIB Instrument Control.vi (continued)

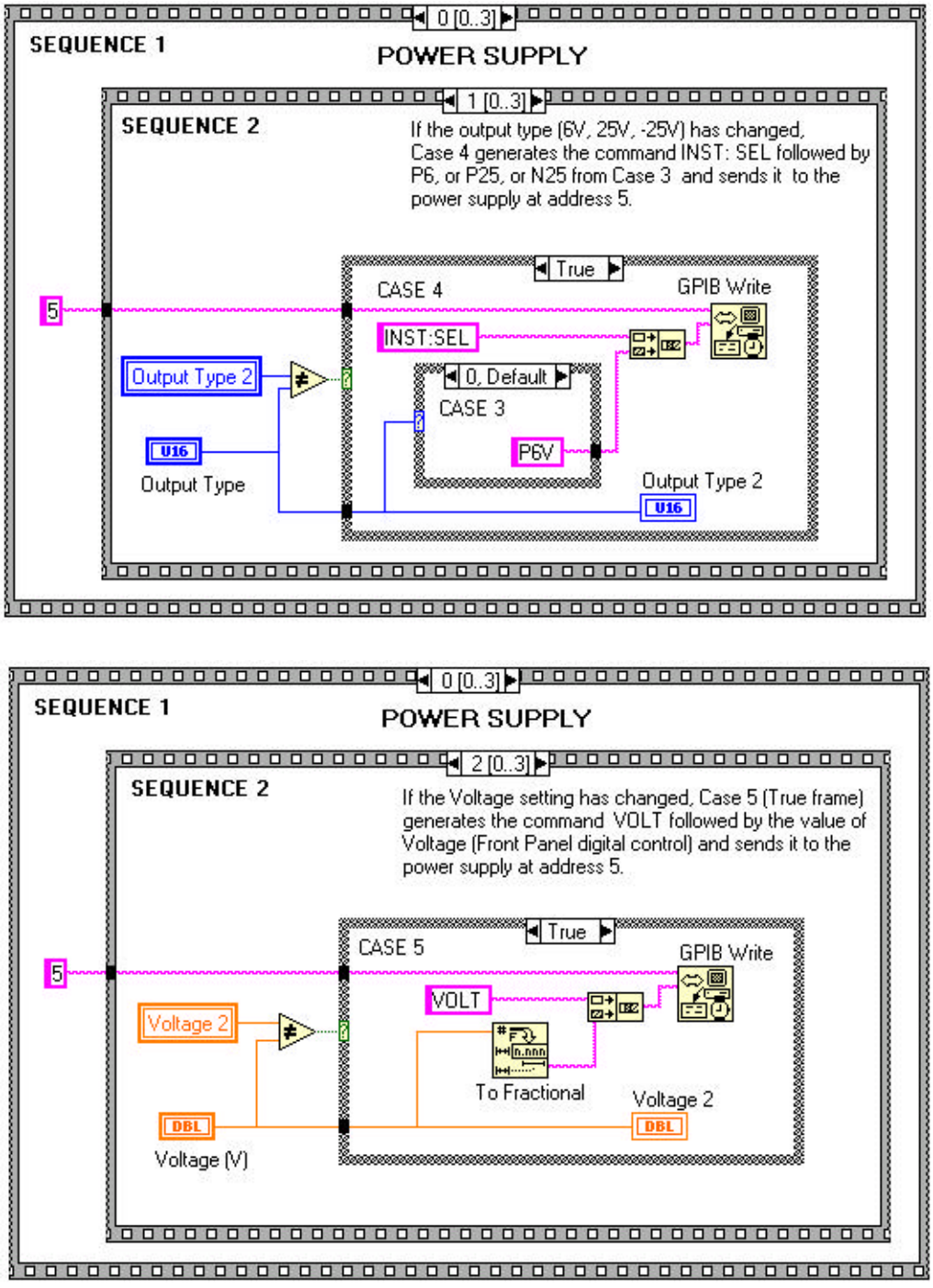


Fig. 3 The Front Panel and the Block Diagram of GPIB Instrument Control.vi (continued)

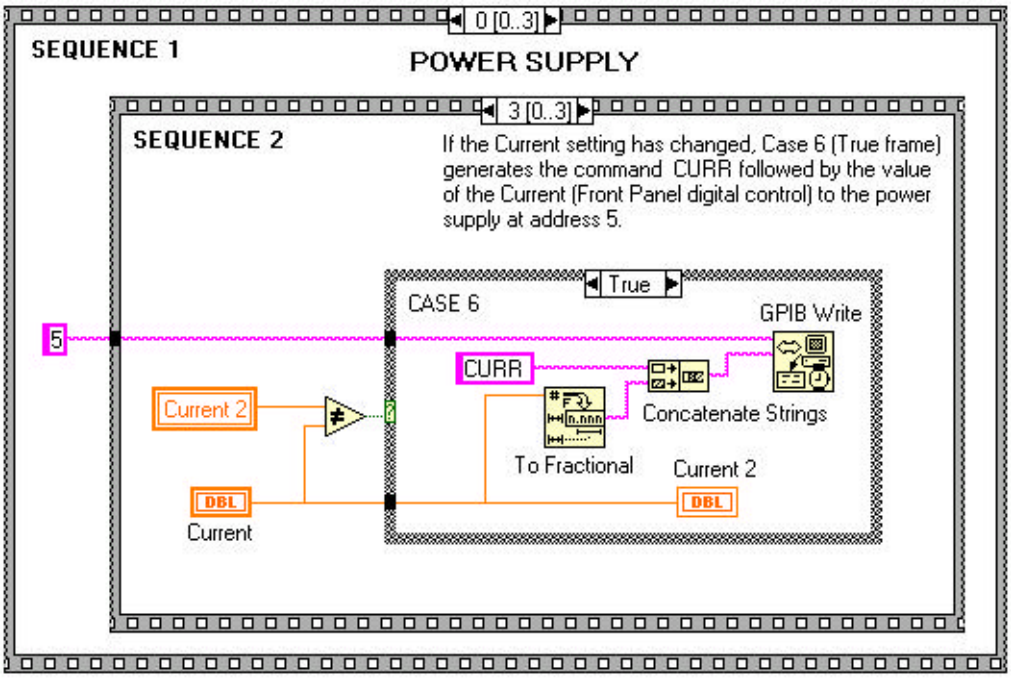


Fig. 3 The Front Panel and the Block Diagram of GPIB Instrument Control.vi (continued)

The last part of the GPIB instrument control experiment includes waveform analysis using the spectrum analyzer. The function generator equipment is connected to the spectrum analyzer and the software described here sets the function generator output to the desired waveform, imports the frequency spectrum data and plots it on a waveform graph in the Front Panel for additional analysis The software has the ability to determine harmonic distortion on a modulated input wave.

GPIB LabVIEW Application

As noted earlier, custom designed GPIB software is most efficient and most flexible. It is efficient because it does not include redundant features that a commercial driver has. It is flexible, because it meets the immediate needs of the user and it can be quickly modified should the requirements change. In addition specific data processing procedures can be implemented into the same software package.

This experiment illustrates the custom design aspects of the GPIB instrument control and data processing software. The hardware configuration is shown in Fig. 4. Bodetest.vi controls the frequency and the amplitude of the Function Generator sinusoidal signal being applied to the Butterworth filter test circuit. The AC voltmeter of the Multimeter unit measures the filter output voltage and returns data to the Bodetest.vi for further processing.

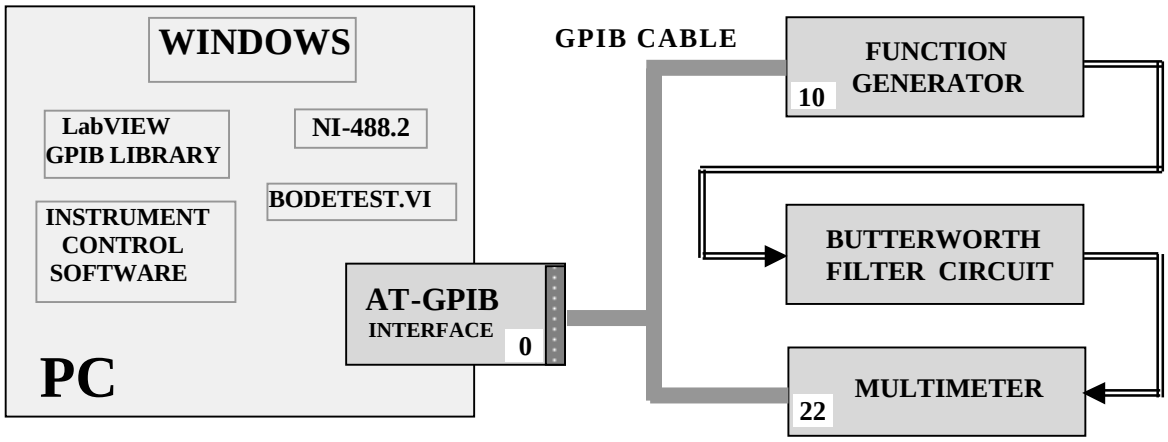


Fig. 4 GPIB Test Setup for a Butterworth Filter

GPIB Application Software: Bodetest.vi

Bodetest.vi is a LabVIEW program that incorporates the GPIB features of instrument control, the processing and display of collected data. The objective of the experiment is to test the frequency response of a Butterworth filter. The collected data is converted to db format and stored in an array. Bodetest.vi increment the frequency by the difference between the *Stop* and *Start* frequencies divided by the *Number of Points*. These parameters are Front Panel Controls as shown in Fig. 5. All frequency points are also converted to db format and stored in an array. The Front Panel in Fig. 5 shows a typical run of the Butterworth circuit data. For comparison the theory based response is also shown.

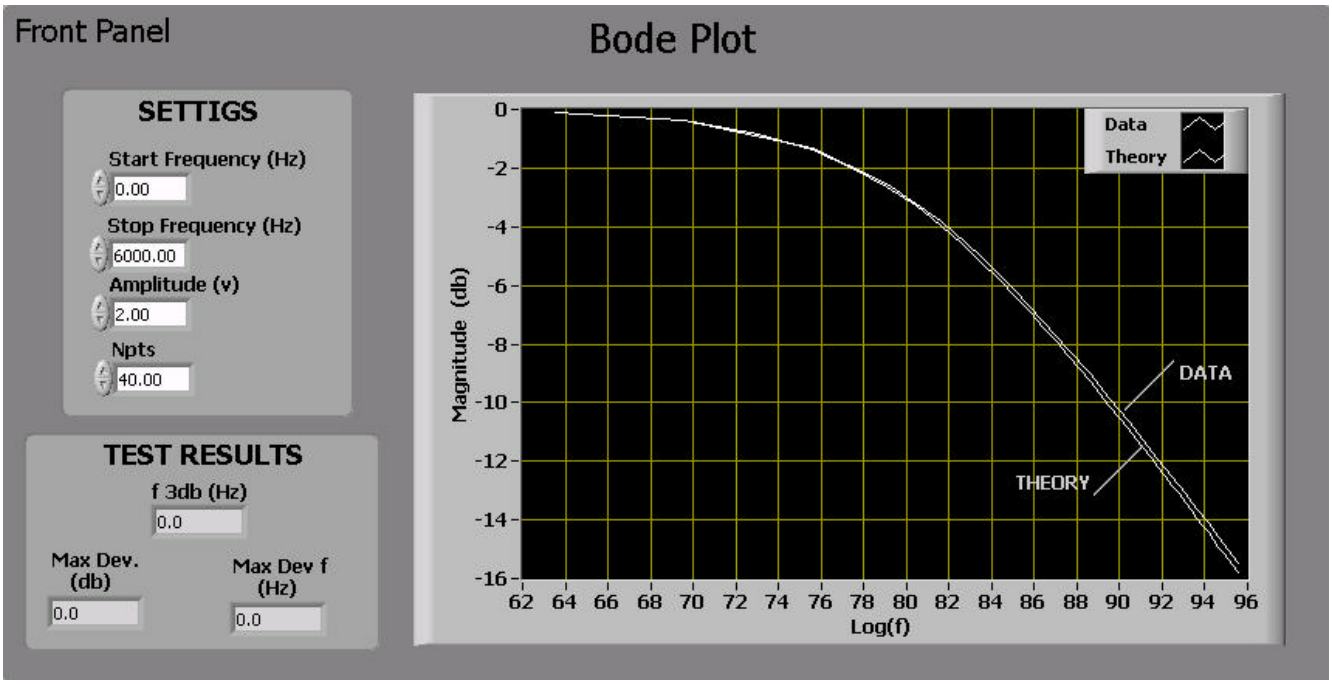


Fig. 5 The Front Panel and the Block Diagram of Bodetest.vi

The partial Block Diagram in Fig. 5 is a nested combination of Sequence Structures.

Sequence Structure 1 has four frames**.** Frame 0 sets the multimeter to read AC voltage and Function Generator to output a sinewave. Frame 1 sets the amplitude of the sinewave to the value entered by the user in the Front Panel digital control. Frame 2 reads the AC voltage from the multimeter, increments and writes the new frequency to the Function Generator, processes and displays data. Frame 3 formats the X-scale of the X-Y graph that displays the Bode magnitude plot. This is done by the *Attribute Node* as it was called in LabVIEW v.5.x and *Property Node* in version 6i. The Property Node is useful in formatting the Front Panel objects. In this case the Property Node sets the minimum and maximum frequency values for the X-axis of the X-Y graph that displays the Bode magnitude plot. The scale is updated automatically, should the user change the values of the *Stop Frequency* or the *Start Frequency* or both.

Sequence Structure 2 also contains four frames. It is inside the For Loop of Frame 2 of Sequence Structure 1. Its N terminal is set to the number of data points to be acquired by the digital control *Npts* in the Front Panel. Hence the number of iterations performed by the loop is same as the number of data points in the Bode plot. Frame 0 determines the Δf, the frequency difference between the data points. Frame 1 writes the new frequency to the Function Generator. Frame 2 sets the display on the Multimeter screen. Frame 3

reads the AC voltage from the Multimeter. After the last iteration of the For Loop the X-array (Frequency) and the Y-array (magnitude) are applied for the display on the X-Y graph. Inside Frame 2 of the Sequence 1 structure and outside of the Sequence 2 structure additional data processing is done. The VI searches the real data array for the 3 db frequency, determines the largest deviation between the real data and the theory based data and the frequency at which largest deviation occurs as shown in Fig. 5.

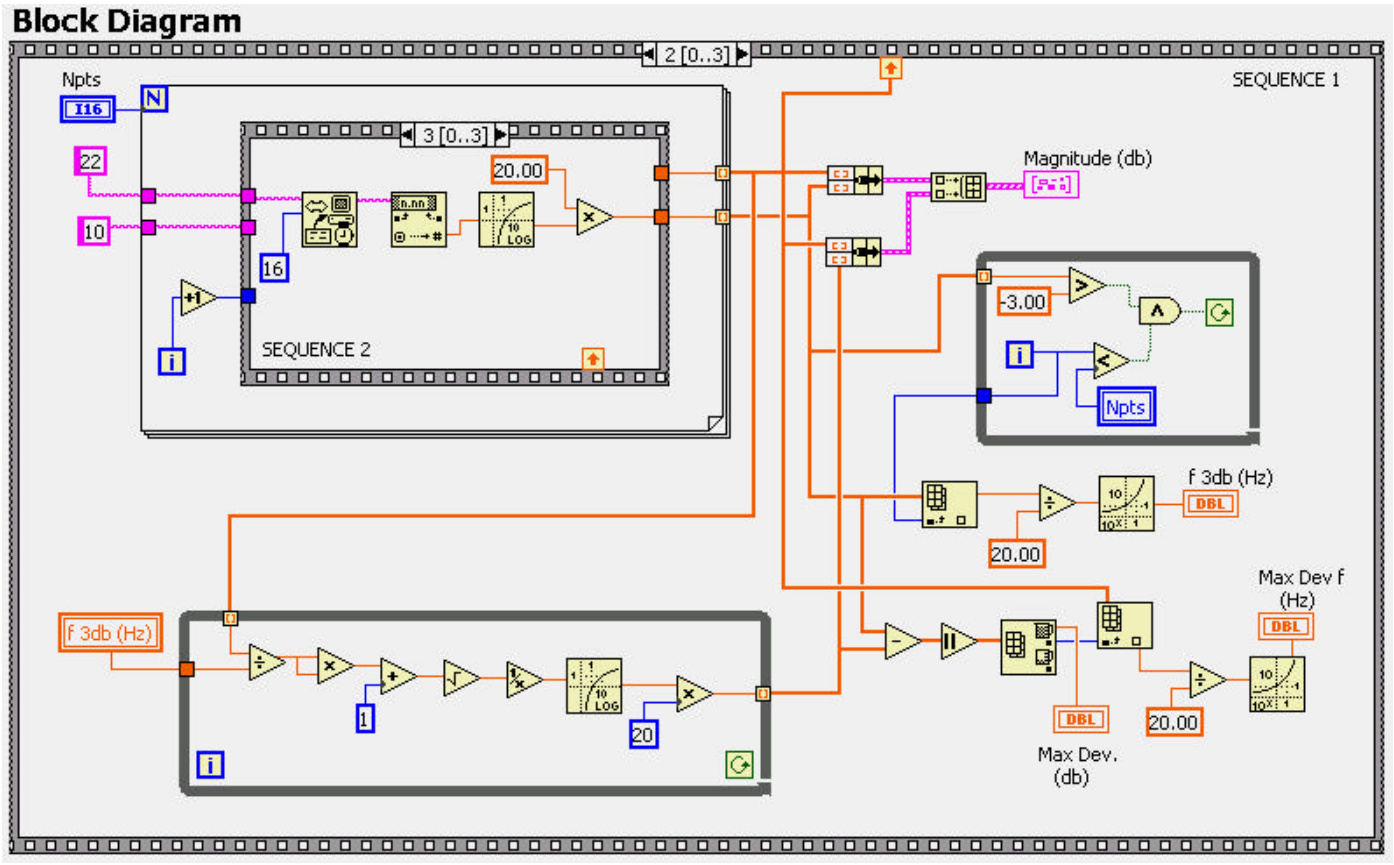


Fig. 5 The Front Panel and the Block Diagram of Bodetest.vi (continued)

Test Results and Conclusions

LabVIEW GPIB software described in this paper includes the GPIB Instrument Control.vi and Bodetest.vi. The former details the methodology for controlling any instrument with GPIB capability. The latter demonstrates a specific application that includes instrument control, acquisition and processing of data. Test results for the Bode magnitude plot agree with theory as can be seen from a typical run shown in the Front Panel of Fig. 5.

In conclusion, this paper presents a practical approach to the development of instrument control software in the LabVIEW environment using the GPIB interface. There are several advantages to the custom designed instrument control software. It meets the immediate needs of the user. The user can also incorporate specific data processing tasks. Bodetest.vi, for example, collects the data into an array, searches for the 3 db frequency, determines the maximum deviation between the real and theory based data and displays the data on an X-Y graph. Multiple instruments may be controlled by the same VI. Modification of the software to include new requirements can be done quickly and easily.

Bibliography

Basic Concepts of LabVIEW 4 by L. Sokoloff, Prentice Hall, 1997.
LabVIEW Graphical Programming by G. W. Johnson, McGraw Hill, 1994.
LabVIEW GPIB Reference Manual, National Instruments.
GPIB Course from National Instruments
Microprocessors and Microcomputers: hardware and Software, R. Tocci, Prentice Hall
Microprocessor Architecture, Programming, and Applications, R. Gaonkar, Merrill, 1989
Electronic Communication Systems, Schweber, Prentice Hall, 1991
Advanced Communication Systems, W. Tomasi, Prentice Hall, 1987
Learning with LabVIEW 6i, R. Bishop, Prentice Hall, 2001

Biography

Leonard Sokoloff was born in Russia, immigrated to the United States in 1950 and was awarded BSEE degree from Stevens Institute of Technology (1959), the MS Applied Science degree from Adelphi University (1964) and the PhDEE (candidate) from Stevens Institute of Technology. Worked in industry as semiconductor application and circuit design engineer (1959 – 1970). For the past 31 years with DeVry College of Technology, currently as senior professor, teaching associate level and bachelor level courses in advanced mathematics, electronic communication and controls. Currently involved in curriculum development projects using Virtual Instrumentation, PLC and PLD and GPIB instrument control.