NASA Technical Memorandum 103849

Using a Graphical Programming Language to Write CAMAC/GPIB Instrument Drivers

Horacio Zambrana and William Johanson

May 1991

National Aeronautics and
Space Administration

NASA Technical Memorandum 103849

Using a Graphical Programming Language to Write CAMAC/GPIB Instrument Drivers

Horacio Zambrana, Eloret Institute, Sunnyvale, California
William Johanson, Ames Research Center, Moffett Field, California

May 1991

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

Using a Graphical Programming Language to Write CAMAC/GPIB Instrument Drivers

HORACIO ZAMBRANA* AND WILLIAM JOHANSON

Ames Research Center

Summary

To reduce the complexities of conventional programming, we have used graphical software in the development of instrumentation drivers. The graphical software provides a standard set of tools (graphical subroutines) which are sufficient to program the most sophisticated CAMAC/GPIB drivers. We made use of these tools and successfully developed instrumentation drivers for operating CAMAC/GPIB hardware from two different manufacturers: LeCroy and DSP. Our paper will present the use of these tools for programming a LeCroy A/D Waveform Analyzer.

Introduction

A graphical programming approach has been selected at the Ames Research Center Hypersonic Free Flight Facility to develop instrumentation drivers. These drivers control and process experimental data acquired by CAMAC (Computer Automated Measurement and Control) and GPIB (General Purpose Interface Bus) instrumentation.

Our basic data acquisition system consists of a Macintosh II computer, an NB-GPIB board, several CAMAC A/D Waveform Analyzers (digitizers), a CAMAC to GPIB interface module, and a CAMAC crate which houses both the digitizers and the interface (see fig. 1). The software package used, LabVIEW, provides programming tools for data acquisition, control, and data processing as well as the flexibility to model almost any application. The flow of control and data in LabVIEW is implemented in a graphical programming language, G. The graphical programming approach makes instrumentation programming an easy task for the programmer. Our paper describes the development of a CAMAC/GPIB driver for an A/D Waveform Analyzer (digitizer) using the software's library of GPIB subroutines.

Horacio A. Zambrana was supported by a grant from NASA to Eloret Institute (NCC2-487).

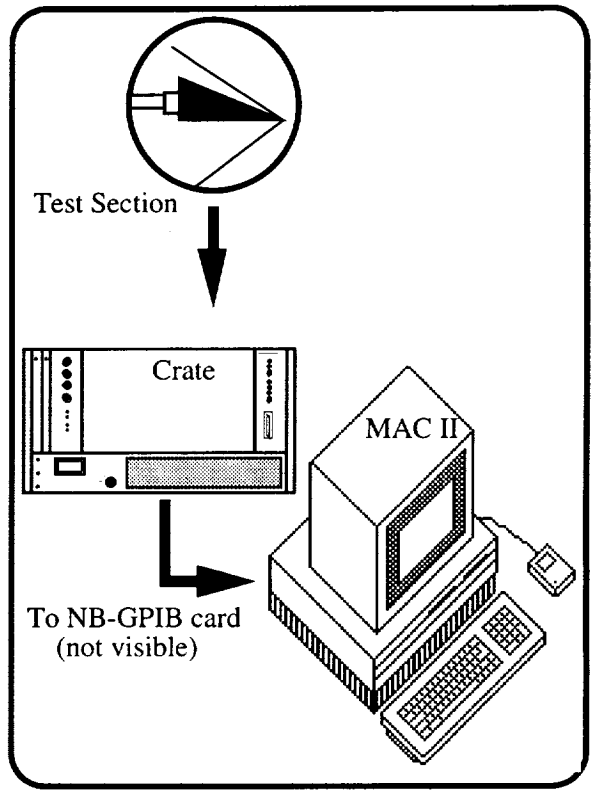


Figure 1. Data acquisition setup.

Background on CAMAC and GPIB

Understanding how data flows in a CAMAC/GPIB environment is essential to the development of the digitizer driver. In general terms, data commands will flow from the computer to the GPIB interface, and from the interface to the CAMAC digitizer. Digitized data flows from the digitizer to the interface, and from the interface to the computer. The digitizer accepts only CAMAC commands. The interface accepts both CAMAC (which are passed along to the digitizer) and GPIB commands. Three of the most relevant CAMAC commands are F(), A(), and N(). The N() command specifies the crate station number of the digitizer that is being addressed. The A() command specifies a subaddress within the digitizer

*Eloret Institute, Sunnyvale, CA 94087.

(such as channel selection), and F() is a function to be performed at the specified subaddress in the selected digitizer. The most commonly used GPIB commands are MLA (My Listen Address) and MTA (My Talk Address). MLA commands the interface to enter Listen mode so that it may accept CAMAC or setup commands from the computer. MTA commands the interface to enter Talk mode so that it may pass data from the digitizer to the computer.

Background on Software

Application programs (such as instrumentation drivers) created by using the graphical software are called virtual instruments (VIs). VIs have three main parts: the front panel, the block diagram, and the icon.

The front panel specifies the inputs and outputs and provides the user interface for interactive operation. The block diagram is a VI's source code, which is created by using the G language.

Figure 2 shows an example of a front panel. This particular example obtains the square of a number. The user scrolls the scale selector to choose a number. The box on the right shows the square of the selected number. Figure 3, the block diagram, shows where the programming takes place. The scale selector is represented by the darkened border terminal, EXT. The output is represented by the undarkened border terminal, EXT. EXT stands for extended precision floating point decimal (fig. 4 shows other block diagram terminal types).

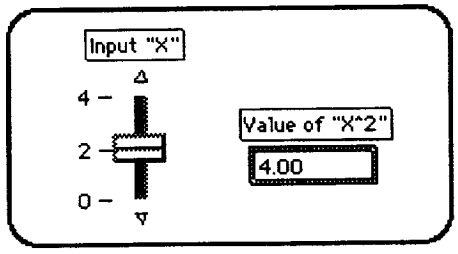


Figure 2. Front panel.

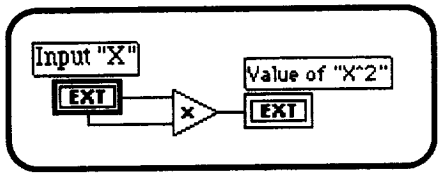


Figure 3. Block diagram.

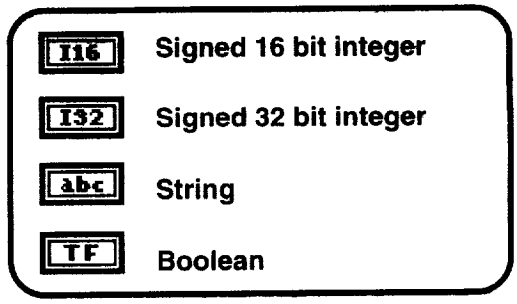


Figure 4 . Block diagram terminals.

Darkened terminals indicate an input from the front panel. In the same manner, undarkened terminals represent outputs. One of many operators available for programming is the multiplication operator. This is represented by the triangular box with the multiplication sign. The source code is generated by wiring the input to the multiplication operator, and wiring the result to the output. The style of the wire is set by the data type of the source terminal to which it is attached (see fig. 5).

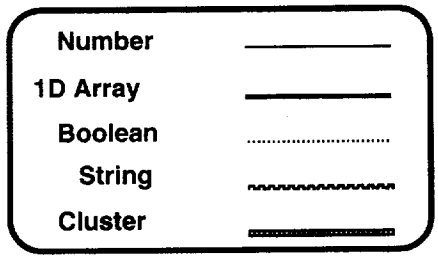


Figure 5. Wire styles.

The icon is a representation of the VI (see fig. 6). The icon represents the VI in a manner analogous to a subroutine call statement when the VI is used as a subVI in another VI's block diagram (ref. 1)

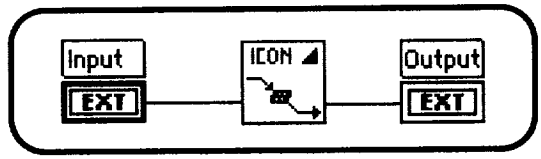


Figure 6. Icon example.

There are various graphical structures which perform the conventional *if then, do, while do loops,* etc. We will briefly discuss the three types used in our discussion. The first type is the *Sequence structure* shown in figure 7. The *Sequence structure* consists of one or more frames that execute sequentially.

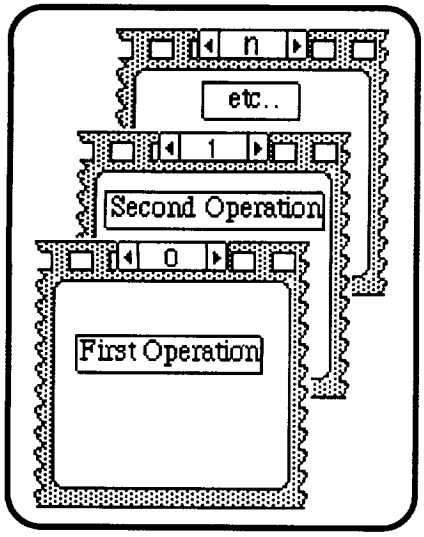


Figure 7. Sequence structure.

The second type is the True/False *Case structure* shown in figure 8. Depending upon the boolean input, the operation in the True or the False frame will be executed.

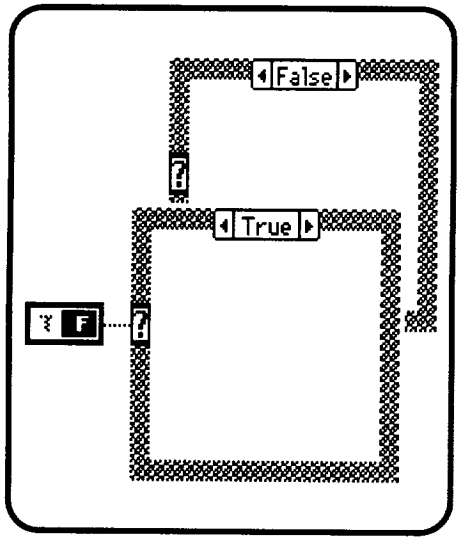


Figure 8. Case structure.

The third type is the *While Loop* shown in figure 9. A *While Loop* executes its subdiagram repeatedly until a false boolean value is passed to the conditional terminal.

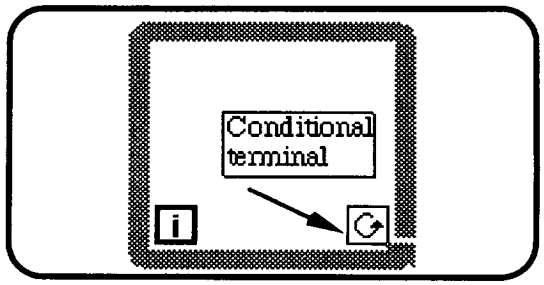


Figure 9. While Loop.

GPIB VIs

Several VIs are supplied with the software to perform specific GPIB functions. The most relevant GPIB VIs are: *GPIB Write*, *GPIB Read*, *GPIB Misc.*, and *GPIB Serial Poll.*

The *GPIB Write* VI is used to send a data string from the computer to the interface. The two principal inputs to the GPIB Write are the data string, which is a string of bytes representing CAMAC and/or GPIB commands, and the address string of the interface. Both strings are encoded in binary. The *GPIB Write* VI icon and the two principal inputs are shown in figure 10.

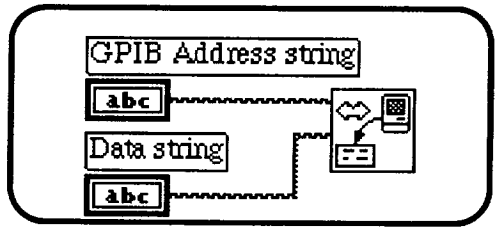


Figure 10. GPIB Write VI example.

The *GPIB Read* VI is used for reading out data from the digitizer through the interface to the computer. The VI can also be used to read status data from the interface. To retrieve the data by using the *GPIB Read* VI, both the GPIB address string and the number of data bytes to be read must be specified. An example using *GPIB Read* is shown in figure 11.

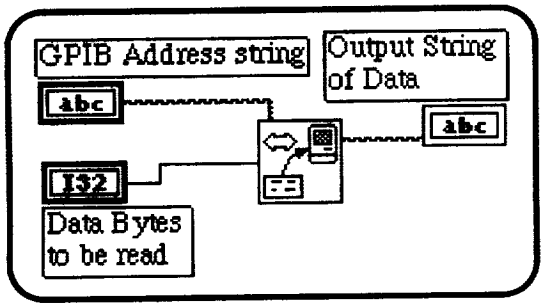


Figure 11. GPIB Read example.

The *GPIB Misc.* VI performs the operation indicated by the GPIB command string. The two most common applications of this VI are for setting the interface in talk mode (MTA command) or for setting the interface in Listen mode (MLA command). A simple example using *GPIB Misc.* is shown in figure 12.

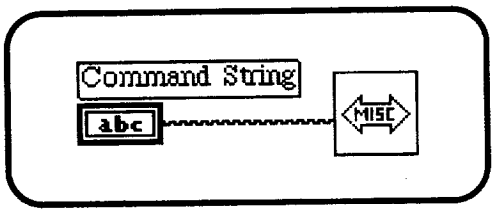


Figure 12. GPIB Misc VI example.

Status bytes are generated by the digitizer and sent to the interface. These bytes convey information such as whether the data read is valid or invalid. *GPIB Serial Poll* reads the status bytes from the interface to the computer (see figure 13).

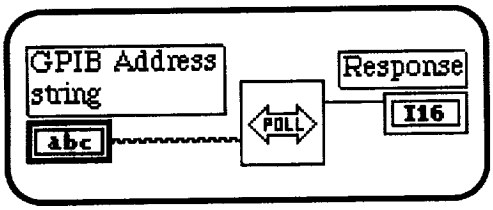


Figure 13. GPIB Serial Poll example.

The Waveform Analyzer VI

The A/D Waveform Analyzer used is a 10-bit transient digitizer designed in accordance with CAMAC standards (IEEE Std. 583). It can process signals from four different sources simultaneously. Although most of the setup commands are selected on the actual digitizer, the initialization and data readout are CAMAC controlled through the interface.

To send commands to the digitizer through the interface, the following steps are required: One, the interface must be addressed to Listen; two, the desired command information (F(), A(), N()) must be loaded in; and three, a CAMAC cycle must be executed to send the command to the digitizer. A CAMAC cycle is executed by addressing the interface to Talk (ref. 2). These procedures must be repeated every time a new command is sent to the digitizer. We created the *SEND FAN* VI to perform these necessary procedures. The *SEND FAN* VI inputs are F(), A(), N() (numerical inputs) and GPIB Address (string input) of the interface. The front panel of *SEND FAN* is shown in figure 14.

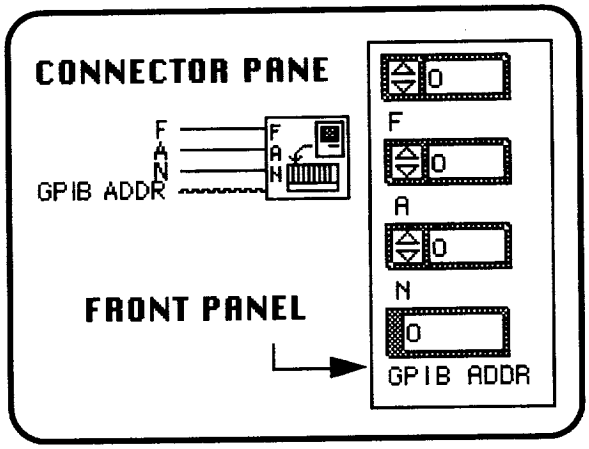


Figure 14. SEND FAN VI.

The block diagram of *SEND FAN* is shown in figure 15. Before F(), A(), and N() are fed into the sequence structure, they are grouped into a numeric array and converted into a binary string. The first operation, sequence-0, executes the *GPIB Write* VI. This addresses the interface to Listen and loads the command information. Sequence-1 executes a CAMAC cycle using the *My Talk Address* SubVI and the *GPIB Misc* VI. The use of *SEND FAN* VI makes the programming of the *Waveform Analyzer* VI an easier task.

The front panel of the *Waveform Analyzer* VI consists of three numerical inputs, one boolean input, one string input, and one graphical output (see figure 16).

The three numerical inputs are: N, the channel of the digitizer to be read out, and the number of data samples to be acquired. The boolean input is a switch for selecting either an internal or an external stop trigger. The string input specifies the GPIB address of the interface. The graph is for displaying the digitized data.

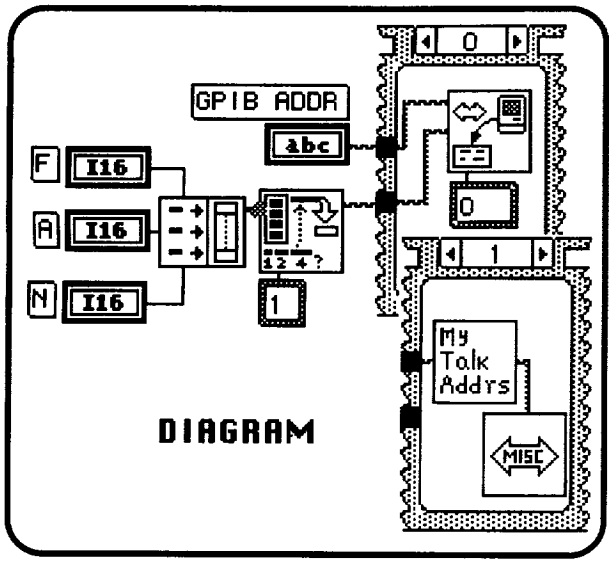


Figure 15. SEND FAN VI Diagram.

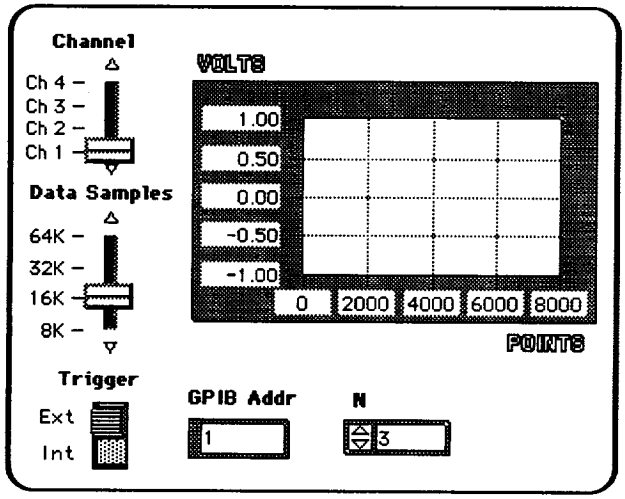


Figure 16. Waveform Analyzer VI Front Panel.

The digitizer control sequence is shown in Figure 17. These commands are specified in the block diagram of the *Waveform Analyzer* VI (see appendix for the VI block diagram sequences). Sequence-0 in the VI diagram issues an F(9) (Initialize digitizer) via the *SEND FAN* subVI. Sequence-1 enables the LAM (Look At Me) by issuing an F(26). LAM is a signal from the module to the interface indicating that the module requests attention. Sequence-2 selects the trigger type according to the front panel boolean input. If false, an internal stop trigger, F(25), is issued. If true, no command is issued. The system will then expect an external stop trigger to occur at a later time. When a trigger is received, the digitizing sequence completes and the digitizer generates a LAM.

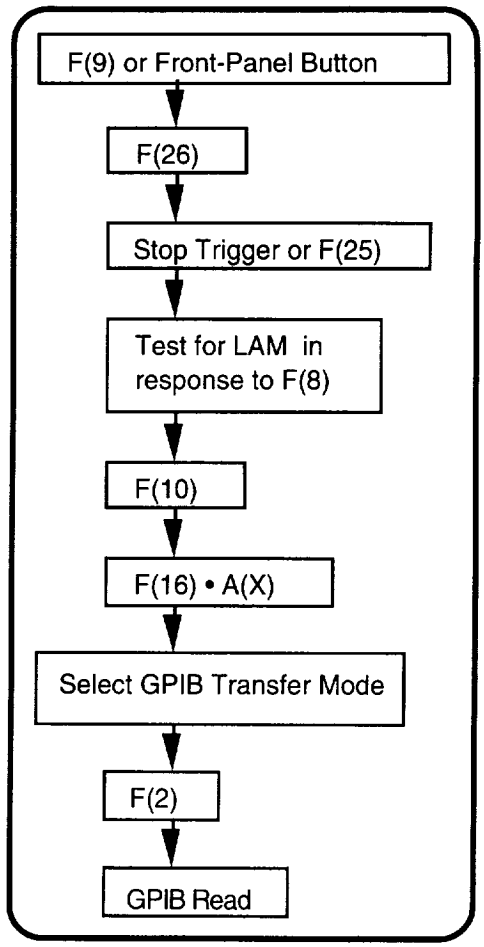


Figure 17. Control Sequence.

Sequence-3 verifies that all the data has been acquired by the digitizer. To do this, Sequence-3 uses a two-step inner sequence structure placed within a *While* Loop. The first inner sequence issues an F(8) (Test for LAM). This function tells the digitizer to send a status byte to the interface. The second inner sequence conducts a 1-byte serial poll using *GPIB Serial Poll* VI to read the status byte. The status byte is then evaluated to determine if the LAM has been generated. The result of this evaluation is wired to the conditional terminal of the *While* Loop. The *While* Loop will continue to cycle until the conditional response is true; i.e., until LAM is generated by the digitizer.

Sequence-4 clears the LAM by issuing an F(10). In sequence-5 the channel to be read is selected by issuing an F(16) A(X). For example, F(16) A(0) selects channel one, F(16) A(1) selects channel two, and so forth. Sequence-6 selects the interface data transfer mode (ref. 3). To specify a 2-byte block transfer mode, 122 is

converted from a decimal integer to a binary string. The string is sent to the interface via the *GPIB Write* VI.

Sequence-7 issues an F(2) (Read data). This command puts the digitizer into the read out mode. In sequence-8, the *GPIB Read* VI extracts the specified number of data bytes from the digitizer through the interface and into the computer. In the same sequence, the digitized data is transformed from a binary string array into a numerical array. Two other operations are necessary to transform the raw data into voltage. Having completed these steps, we transfer the data into Sequence-9 where it is wired to the graphical terminal for display on the front panel.

Summary and Overall Comments

We have just demonstrated how a graphical programming language is used to write CAMAC/GPIB instrumentation drivers. The GPIB VIs provided by the software enable the user to create instrumentation drivers without having to learn the more complicated details of the CAMAC and GPIB protocols.

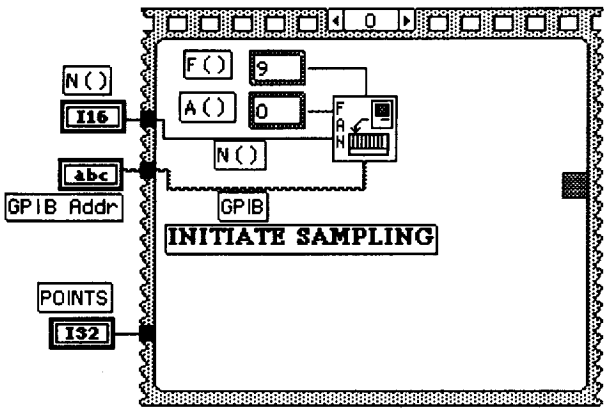
Using these VIs, two DAS systems have been developed at Ames Research Center for high-speed data acquisition, control and processing. Each system used different CAMAC and GPIB instrumentation hardware. Both systems were developed in a short period of time, by individuals with no previous DAS experience.

References

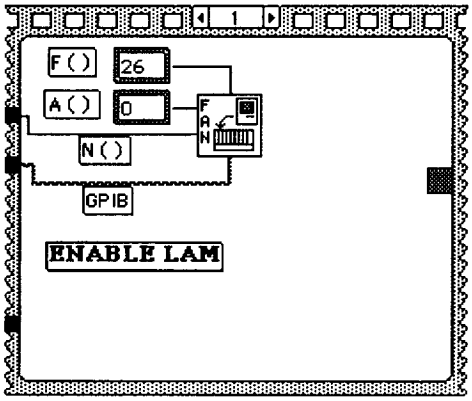
1. LabVIEW 2 User Manual. National Instrument Corporation, Austin, Texas, Jan. 1990, p. 1-3.
2. CAMAC TO GPIB INTERFACE MODEL 8901A (manual), LeCroy, Chestnut Ridge, New York, Jan. 1986, p. 2-1.
3. QUAD 10-BIT TRANSIENT DIGITIZER MODEL 8210 (manual), LeCroy, Chestnut Ridge, New York, June 1987, p. 1-4.

Appendix

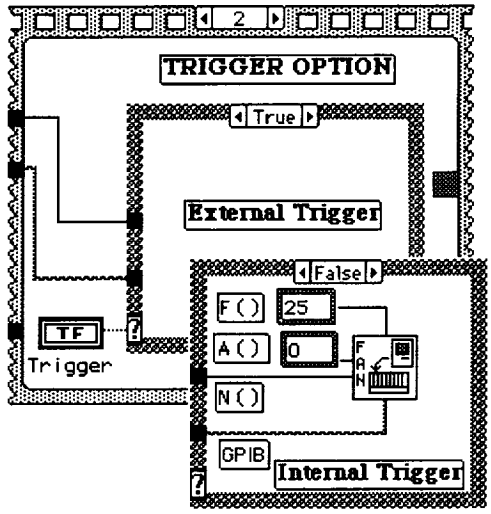
Block Diagram for Waveform Analyzer



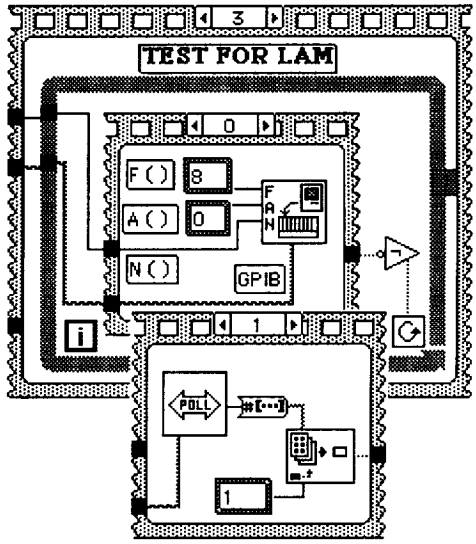
Sequence 0: Initiate Sampling



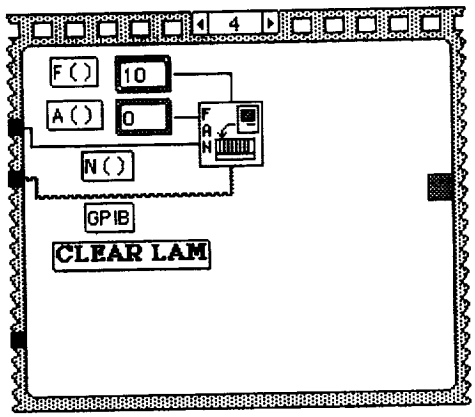
Sequence 1: Enable LAM



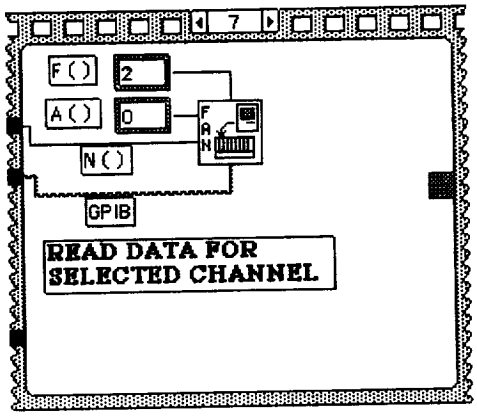
Sequence 2: Select Trigger



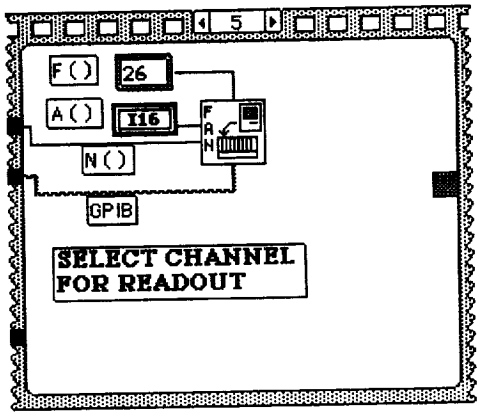
Sequence 3: Test For LAM



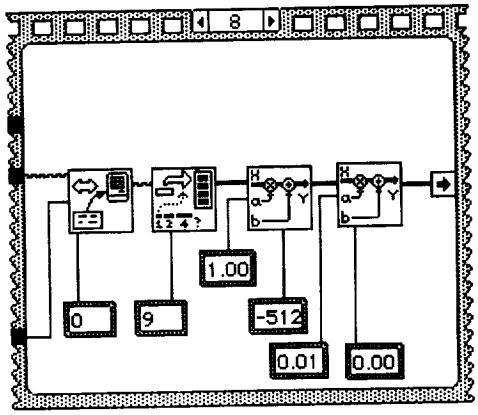
Sequence 4: Clear LAM



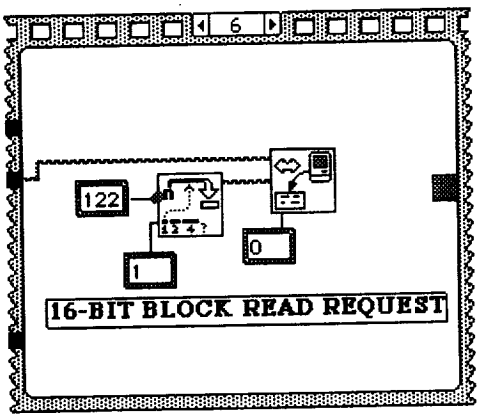
Sequence 7: Read Data



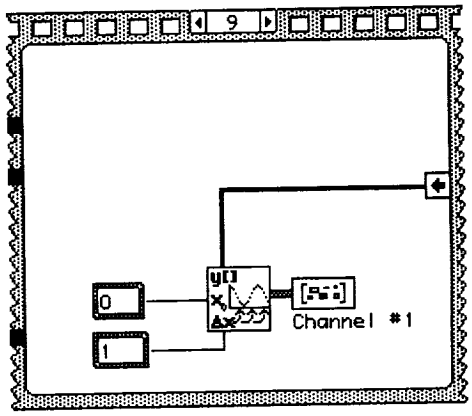
Sequence 5: Select Channel



Sequence 8: Convert Data



Sequence 6: Specify Data Transfer Mode



Sequence 9: Prepare and Graph Data

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-103849		

4. Title and Subtitle

Using a Graphical Programming Language to Write CAMAC/ GPIB Instrument Drivers

5. Report Date

May 1991

6. Performing Organization Code

7. Author(s)

Horacio Zambrana* and William Johanson

8. Performing Organization Report No.

A-91100

10. Work Unit No.

592-01-11

9. Performing Organization Name and Address

Ames Research Center
Moffett Field, CA 94035-1000

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: William Johanson, Ames Research Center, MS 240-8,
Moffett Field, CA 94035-1000
(415) 604-3780 or FTS 464-3780

*Eloret Institute, Sunnyvale, CA 94087

16. Abstract

To reduce the complexities of conventional programming, we have used graphical software in the development of instrumentation drivers. The graphical software provides a standard set of tools (graphical subroutines) which are sufficient to program the most sophisticated CAMAC/GPIB drivers. We made use of these tools and successfully developed instrumentation drivers for operating CAMAC/GPIB hardware from two different manufacturers: LeCroy and DSP. Our paper will present the use of these tools for programming a LeCroy A/D Waveform Analyzer.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
GPIB CAMAC LabVIEW Instrument driver	Unclassified-Unlimited Subject Category - 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	12	A02

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161