

GENERAL-PURPOSE AUTOMATION PROGRAMMING
USING A GRAPHIC LANGUAGE

By

SYED MOHAMMAD MAHMOOD

Master of Science
Stanford University
Stanford, California
1980

Doctor of Philosophy
Stanford University
Stanford, California
1987

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1996

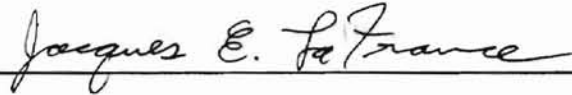
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1996

GENERAL-PURPOSE AUTOMATION PROGRAMMING
USING A GRAPHIC LANGUAGE

Thesis Approved:



Thesis Adviser



Dean of the Graduate College

PREFACE

Automation is a useful tool in enhancing the performance of laboratories/plants where the response of several instruments need to be coordinated. The cost of developing custom-made software as well as the cost of acquiring and configuring commercial software are often prohibitive for many operations that could benefit from automation.

This situation is rapidly changing with the emergence of a new generation of object-oriented graphical languages in the past few years. These new languages allow programmers to develop sophisticated systems with unprecedented ease and speed. This report introduces the concepts behind this new generation of languages and their use for every day automation. A general-purpose automation program developed by the author for the U.S. Department of Energy at NIPER (National Institute for Petroleum and Energy Research) is presented as a case-example. Two example uses of this program are also presented.

This menu-driven and user friendly program for data acquisition/control/analysis/presentation is a valuable tool for laboratories where test requirements change frequently. Its features include: run-time system setup and reconfiguration; built-in constraints to eliminate operator errors; real-time graphic display of current and previous data with editing and smoothing; data export to other applications for report generation or animation such as automatic 3-D rotation; pictorial display of test status for quick troubleshooting; error and out-of-range audio/video warnings to designated users on the network; automatic resetting of instrument(s) to rectify minor problems; and automatic shutdown in emergencies when operator does not respond.

ACKNOWLEDGMENTS

I wish to express sincere appreciation to my advisor, Dr. Jacques E. LaFrance. He has been a source of inspiration through the course of my education in computer science. I very much appreciated his free-spirited style of teaching which allowed both educational and personal growth for me. I thought of him more as a friend than a counselor.

The work was sponsored by the U.S. Department of Energy under a cooperative agreement with NIPER (my employer) DE-FC22-83FE60149. Parts of this report were copied and edited from NIPER-689, September 1993, entitled "NIPER LabWARDEN: Description and LabVIEW® Executable Code of a General-Purpose Laboratory-Automation Program." I wish to thank the co-author of this report, Dr. David K. Olsen, NIPER management, and Tom B. Reid of the DOE Bartlesville Project Office for permission to copy the contents and providing assistance in completing this report. The staff of National Instruments (the developer of LabVIEW graphical language) deserves credit for reviewing the software. Jonathan Grigsby, Rebecca Horning, Ameer Kendall, Danny Lemmons, William Lucas, Gautam Sharma and Yesh Tyagi tested the software and provided help in improving the manuscript.

And finally, Annalea Kerr deserves much credit for preparing this report. Her help, perseverance, and patience are very much appreciated.

TABLE OF CONTENTS

	<u>Page</u>
Abstract	1
Executive Summary	1
Objective	2
Format of Report	2
Advantages of Object-Oriented and Graphical Programming Approach	3
Technical Requirements	4
Bibliography	5
 CHAPTER 1. DESCRIPTION OF LabVIEW	 6
Brief Introduction of LabVIEW	6
Programming Concepts Relevant to LabVIEW	7
A Sample of LabVIEW Program Algorithm	7
Object-Oriented Programming	8
Procedure-Oriented Programming Versus Object-Oriented Programming	11
Programming Structure of LabVIEW	13
Basic Facilities in LabVIEW	14
Writing and Editing Program	14
Built-in Library	17
Control Structures	17
On-Line Help	20
Bibliography	20

TABLES

1.1 Legend for figure 1.5	10
1.2 Legend for figure 1.10	19

ILLUSTRATIONS

1.1 A simple dataflow diagram	6
1.2 Dataflow diagram simplified	7
1.3 A sample FORTRAN program	8
1.4 The control-flow diagram of the example FORTRAN program (shown in Fig. 1.3)	9
1.5 A sample program segment in LabVIEW	9
1.6 Different ways of classification of objects: (a) POP (b) OOP	11
1.7 NIPER MAIN FACILITY panel	15
1.8 Descriptive hierarchy of programming for the NIPER MAIN FACILITY	16
1.9 Example of icons from LabVIEW 2 library	18
1.10 Basic control structures in LabVIEW	19

	<u>Page</u>
CHAPTER 2. AN OVERVIEW OF NIPER Lab WARDEN FACILITIES	23
NIPER MAIN FACILITY	23
NIPER DISPLAY FACILITY	25
NIPER GRAPHICS FACILITY	26
Bibliography	28

ILLUSTRATIONS

2.1 Major functional units of NIPER's automation program	23
2.2 Hierarchical structure and front panel of NIPER Lab WARDEN	24
2.3 NIPER MAIN FACILITIES panel	24
2.4 NIPER DISPLAY FACILITY panel	25
2.5 NIPER GRAPHIC FACILITY panel	26
2.6 A sample snapshot of automatic 3-D display of data	27

CHAPTER 3. DESCRIPTION OF NIPER MAIN FACILITY	29
Features	29
Technical Information for Programmers	30
Displays and Controls	35
Control Bar	35
Other Controls and Displays	37
Installing Drivers to the Automation Program	37
To Classify Instruments on the Basis of Their Functionality	39
To Analyze a System Configuration Problem	39
To Determine Correct Parameters for the NIPER INDICATOR VI	39
To Determine Correct Parameters to be Entered into the NIPER CONTROL VI ..	41
To Configure NIPER MAIN FACILITY for Specific Automation Setups	41

TABLES

3.1 Legend for figure 3.1	32
3.2 Legend for figure 3.2	33
3.3 Legend for figure 3.3	34
3.4 Problem analysis work sheet	40
3.5 NIPER Indicator Panel settings	42
3.6 NIPER Control Panel settings	43
3.7 List of drivers and directory location(s)	44

ILLUSTRATIONS

3.1 Front panel of NIPER MAIN FACILITY	31
3.2 Front panel of NIPER Indicator VI	33
3.3 Front panel of NIPER Control VI	34

CHAPTER 4. DESCRIPTION OF NIPER DISPLAY FACILITY	45
Features	45
Technical Information for Programmers	45
Displays and Controls	46

	<u>Page</u>
CHAPTER 5. DESCRIPTION OF NIPER GRAPHIC FACILITY	49
Features	49
Technical Information for Programmers	49
Displays and Controls	52
Menu Buttons	52
STOP Button	52
HELP Button	52
EDIT GRAPH Button	53
EXTERNAL GRAPHIC Button	53
REVIEW EX-PLOT Button	53
RECENT PLOT Button	54
CURVE FIT Button	54
ADJUST GRAPHIC Button	54
LOG GRAPH SETTINGS Button	54
Run Number Displays	54
NO. OF EX-RUNS Display	54
RUN NO. Display	56
RUN INFORMATION Display	56
Information Displays	56
DATA & DIAGNOSTICS ALARM	56
DATA & DIAGNOSTICS Display	56
GRAPHIC MESSAGES Display	56
Graphical Display	57
Graphic Display Legend	57
Graphics Cursors	57
Cursor Position	57
Cursor Movement	57

TABLES

5.1 Legend for figure 5.1	50
5.2 Legend for figure 5.4	55

ILLUSTRATIONS

5.1 The front panel of NIPER GRAPHIC FACILITY VI	50
5.2 The front panel of NIPER HELP VI	52
5.3 The front panel of Full size graph VI	53
5.4 The front panel of NIPER graphic configuration VI	55

CHAPTER 6. EXAMPLE PROBLEMS	59
Problem 1. Operation of an Electronic Balance	59
Guidance for the Problem	59
Sample Solution for Problem 1	60
Panels and Diagrams for Configuration of Problem 1	61
Problem 2. Operation of an Ice Cream Manufacturing Plant	67
Sample Solution for Problem 2	68
Bibliography	73

6.1	Problem analysis work sheet	69
6.2	Settings for indicator instruments	70
6.3	Settings for control instruments	71
6.4	List of drivers and directory locations	72

ILLUSTRATIONS

6.1	Schematic of electronic balance and computer set-up for problem 1	59
6.2	Schematic of the Macintosh and Mettler PJ-15 pin configuration	60
6.3	Diagram of "Indicator Driver Selector" VI after the driver has been loaded	62
6.4	Steps involved in using the "Find File" function from the menu	62
6.5	Front panel of "NIPER Lab WARDEN" VI	63
6.6	Front panel of "NIPER I/O Facility" VI	63
6.7	Front panel of "NIPER Indicator" VI	64
6.8	How to set default values for the current run from the menu	64
6.9	Front panel of "NIPER Control" VI	65
6.10	Front panel of "NIPER MAIN FACILITY" VI	65
6.11	Front panel of "NIPER GRAPHIC FACILITY" VI	66
6.12	Front panel of "NIPER Graph Configuration" VI	66
6.13	Major units in ice cream plant	67

GENERAL-PURPOSE AUTOMATION PROGRAMMING USING A GRAPHIC LANGUAGE

by Syed Mohammad Mahmood

ABSTRACT

This report describes a general-purpose automation program developed by the author for data acquisition/control/analysis/presentation. This software provides interactive computer control of a variety of instruments typically found in laboratories and pilot plants in order to improve operational efficiency and safe handling of potentially hazardous operations. For example, it can be easily adapted to operate a laboratory that conducts experiments at extreme conditions of pressure and temperature, such as those found in a steamflooding laboratory. The software was developed in an object-oriented graphical language around National Instruments' LabVIEW® which is the future trend in automation programming.

EXECUTIVE SUMMARY

This report describes a computer program that occupies two 1.4 Meg floppy disks (auto extracting compressed file). The software was originally developed to operate NIPER's thermal oil production research laboratory (a U.S. DOE facility). This is a general program that can be readily adapted by other users to their specific laboratory or pilot plant application. The use of an object-oriented graphical (symbolic) computer language in this program permits easier and faster adaptation than the typical line-code languages such as FORTRAN. The programs look like logic flow diagrams (electrical circuitry) rather than typical syntax line codes. Due to its open architecture, this program is very likely to provide enhancements over custom-made programs in situations where instrument requirements change frequently, or when several groups need to interact or share data over a network. With a uniform, standardized, and flexible automation program, information/experience/training can be shared to increase efficiency, safety, and reliability. More specifically, this program provides the following features:

- (1) allows the user to select any number of instruments and specify their parameters for a particular run. The selection can be made during run-time, or one of the previous selections can be chosen prior to a run. The devices/instruments can be added/deleted/reset anytime during a run;

- (2) acts as a liaison between instrument-related software (driver programs) for various instruments, i.e., links up or connects various driver programs in order to bring about proper coordination of activities;
- (3) provides means to monitor instruments so that the user is informed if values go out of user-specified range;
- (4) eliminates operator errors by setting constraints;
- (5) allows the user to automatically rectify minor problems by specifying what to do if an instrument's value falls into a warning range, or if there is an abrupt change, i.e., automatically reset one or more devices to a specified percentage of their current value;
- (6) allows the user to automatically handle emergencies by specifying an emergency response sequence so that if an instrument(s) value(s) fall outside the acceptable range, or if there is an abrupt change, the devices can be automatically reset to safe values in an orderly fashion;
- (7) provides easy access and editing to current and previous data with advanced graphic features; and
- (8) provides a visual display of the status of a test in progress for quick troubleshooting.

OBJECTIVE

The objective of this report is to describe a software program that was primarily developed to provide laboratories or pilot plants of moderate complexity: (1) a simple yet powerful automation program that is inexpensive, flexible, easy to use/modify, and capable of monitoring test progress and taking corrective actions automatically when needed; (2) an effective program that does not require a workstation for effective utilization; (3) a program that allows interaction with other major software packages to enhance its capabilities, e.g., to automatically open Microsoft Excel, transfer data in its preferred format, and plot this data; thus, saving time and effort over doing it manually; and (4) a program that allows real-time access to the input and output data so that system set-up can be changed and reports can be generated in real-time.

FORMAT OF REPORT

Chapter 1 of this report introduces LabVIEW®—the language used to develop this software—and presents the concepts behind object-oriented and graphical languages. Chapter 2 provides an overview and a conceptual framework of NIPER's automation programs. Chapters 3, 4, and 5 provide general description of the three main facilities:

NIPER MAIN FACILITY, NIPER DISPLAY FACILITY, and NIPER GRAPHIC FACILITY, respectively. Chapter 6 includes two example applications/problems.

Individuals have different levels of proficiency in the use of computer operating systems, object-oriented programming (LabVIEW), and communication hardware (plug-in boards, etc.). The author has assumed two different levels of users. On one level are operators who will take the program and adapt the current configuration to their needs. On another level are programmers who may wish to extensively revise and develop their own program around the NIPER programs provided. This second type of user is assumed to be more computer literate and knowledgeable about object-oriented programming structure. With regard to Macintosh computers, the user is assumed to have a good understanding of the fundamentals of System 7[®] operations (Apple[®], 1991). The second level user should also have basic proficiency in LabVIEW programming obtained by going through the training exercises (learn by doing), which are found in the National Instruments LabVIEW literature (National Instruments Corp., 1991).

For automation programs (laboratory, pilot plant, or production facilities) dealing with instrument control, it is imperative that the validity and/or reliability of the program be established with reasonable certainty. No warranty or guarantee of the applicability of the programs is implied by NIPER or DOE. The user must ascertain the suitability of the system for the user's own specific needs.

ADVANTAGES OF OBJECT-ORIENTED AND GRAPHICAL PROGRAMMING APPROACH

Object-oriented graphical programming is an approach that has received great attention and acceptance in recent years. Since Macintosh[®] and Microsoft Windows[®] for PC operating systems became available, most commercial software has been developed using the object-oriented approach. The same trend is also seen in lab automation programming, where object-oriented graphical languages are replacing line-code programs as the language of choice.

In object-oriented graphical programming, the program is coded as a series of symbols (objects). The related data for each object are packaged along with the program code. Objects can be defined within objects. This hierarchical structure allows the breaking down of complex programs into smaller objects. Since data are packaged along with the code for each object, their present value is saved each time the program is run and saved.

There are many advantages of object-oriented graphical languages. First, writing, debugging, and modifying the code are easier because each object can be treated as an

independent program not linked to other programs (each object interacts with other objects only by sending and receiving messages). Second, the objects and their input/output data can be pictorially depicted by icons which increases the clarity for both programmers and users. Third, several processes or programs can be run simultaneously by considering each program as an object. By merely switching their active status, they can run intermittently and interact with each other. This also extends a programmer's capabilities to use external codes. The linking of multiple processes and objects increases user-friendliness and error-handling capabilities.

TECHNICAL REQUIREMENTS

NIPER's automation software was written on an Apple Macintosh II computer using LabVIEW®, an object-oriented language from National Instruments. The user must acquire a licensed version of National Instruments' LabVIEW software to legally execute the NIPER program. LabVIEW was selected as an operating platform because it allows these programs to be run on either Macintosh, IBM-compatibles, or Sun workstations with an appropriate version of LabVIEW software. LabVIEW 2 requires 4 megabytes of RAM and >16 megabytes of hard disk space. The NIPER programs, along with necessary LabVIEW software, require 6 megabytes of RAM and 6 megabytes of hard disk space. If supporting software such as Microsoft Excel®, Microsoft Word®, or other graphics software such as Spyglass® are to be interacted with NIPER programs, their memory requirement needs to be added. However, it is not necessary to run every NIPER program simultaneously; thus, the program can be run with 2-3 megabytes of memory at a minimal level.

Users do need to provide and load the driver programs for communicating with their unique instruments. A driver program is a mediator/translator between the computer and an instrument. Some instruments may communicate in unique command languages. The LabVIEW package includes several driver programs, but many more standard driver programs are available from third party consulting/software development companies. A template and examples of driver programs are included in NIPER's package to ease the task of writing a driver program. A driver program can be written or modified by experts, on the average, in a few hours. National Instruments also provides excellent technical support by telephone, fax, or mail. All above options require at least a cursory knowledge of LabVIEW. Inexperienced users should seek help from others in obtaining driver programs.

BIBLIOGRAPHY

- Apple Computer Inc., 1986. *Macintosh II User Manual*. Cupertino, CA.
- Apple Computer Inc., 1986. *Inside Macintosh, Volume VI*. Cupertino, CA.
- Dijkstra, E. W. Notes on Structured Programming. Contained in Reference 10, pp. 1-82.
- Dahl, O. J., E. W. Dijkstra, and C. A. R. Hoare, 1972. *Structured Programming*. Academic Press, London.
- Kirkman, I. W. and P. A. Buksh, 1992. *Data Acquisition and Control Using National Instruments' "LabVIEW" Software*. Rev. Sci. Instrum., v. 63, No. 1, January, pp. 869-872.
- Kodosky, J., J. MacCracken, and G. Rymar, 1991. *Visual Programming Using Structured Dataflow*. Proceedings of the 1991 IEEE (Institute of Electrical and Electronics Engineers) Workshop on Visual Languages, Kobe, Japan, Oct. 8-11. Reprinted by IEEE Computer Society, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264.
- Liles, K., 1991. *Data Acquisition Software Automates Automotive Fuel Injector Test Facility*. Engineering & Management, October/November, pp. 18-21.
- Liles, K., 1992. *Data Acquisition—A Mac-Based System for Fuel and Lubricant Testing*. Scientific Computing & Automation, January, pp. 19-23.
- National Instruments Corp., 1991. *LabVIEW 2—Getting Started Manual*, Part No. 320246-01. Austin, TX, April.
- National Instruments Corp., 1991. *LabVIEW 2—User Manual*, Part No. 320244-01. Austin, TX, September.
- National Instruments Corp., 1991. *Training In-depth course on LabVIEW 2*, Version 1.4, Part No. 776393-01, Austin, TX, April.
- National Instruments Corp., 1991. *LabVIEW® 2 Lab Driver® VI Library Reference Manual*, Part No. 320249-01, Austin, TX, February.
- Olsen, D. K., S. M. Mahmood, P. S. Sarathi and E. B. Ramzel, 1992. *Operating Guide and Specifications for NIPER Steamflood Laboratory*. DOE Report NIPER-603, August.
- Olsen, D. K., S. M. Mahmood, P. S. Sarathi and E. B. Ramzel, 1990. *Thermal Processes for Light Oil Recovery*. DOE Report NIPER-515, December, pp. 18-20.
- Sethi, R., 1989. *Programming Languages: Concepts and Constructs*. AT&T Bell Laboratories, Murray Hill, NJ, Addison-Wesley, Reading, MA.
- Stroustrup, B., 1986. *The C++ Programming Language*. Addison-Wesley, Reading, MA.

Chapter 1

DESCRIPTION OF LabVIEW

BRIEF INTRODUCTION OF LabVIEW

LabVIEW® is a visual programming environment that can be effectively used by a broad range of people with different levels of programming skills. The two-dimensional graphical notations that LabVIEW uses are much easier to comprehend than textual notations in line-code languages. Programs look like a dataflow diagram, as shown in Fig. 1.1. Elements are pictorially represented, and data flow between elements is shown through color-coded wire-connections (lines). With this representation, constructing or understanding a program is easier and faster (National Instruments Corp., *LabVIEW 2—Getting Started Manual, 1991a*, and *LabVIEW 2-Users Manual, 1991c*). Complex process flow diagrams of large plant operations can be broken down into small logical units and then recombined. Thus, complex operations and their interdependence can be understood with relative ease. Kodosky, MacCrisken and Rymar (1991) have described some of the programming structure behind LabVIEW, and a number of their examples have been used in this section because of their clarity.

A simplified version of Fig. 1.1 is shown schematically in Fig. 1.2 and explained in detail under “Programming Structure of LabVIEW.” Input can be an electrical signal (voltage) from an instrument, an assigned value from the front panel (which can be numeric, a string, or a series of alphanumeric characters), a string (series of alphanumeric characters), or a calculated value. A node is a process (addition, subtraction, integration,

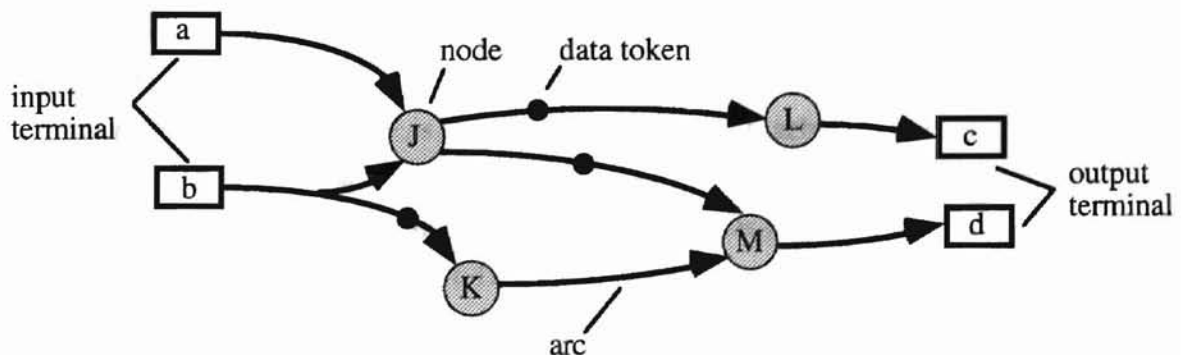


FIGURE 1.1 - A simple dataflow diagram.

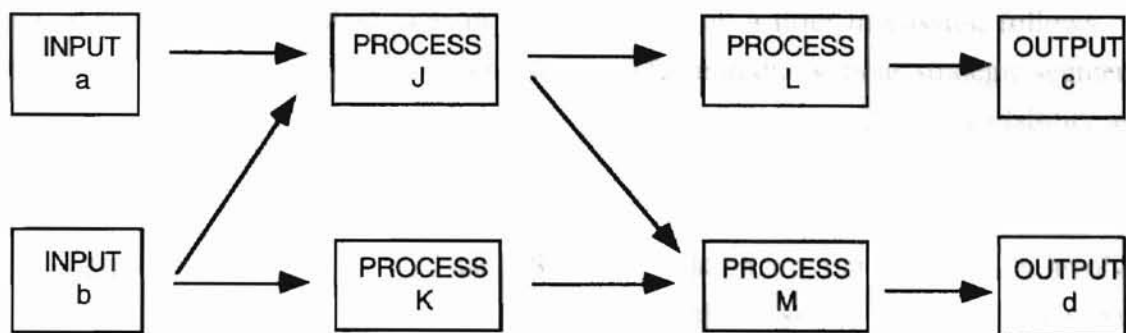


FIGURE 1.2 - Dataflow diagram simplified.

listing, sorting, etc.). An arc is the connecting link between operations, i.e., it shows where an input value leads to. A data token is an imaginary symbol superimposed on arcs to indicate that the data packet has arrived. Data tokens are used to show the execution sequence of various nodes in the program. An output terminal may be a display on the front panel, an instrument, valve, alarm, or may involve printing a file, etc.

PROGRAMMING CONCEPTS RELEVANT TO LabVIEW

This section describes the concepts of programming languages that are pertinent to LabVIEW. The following discussion is aimed at highlighting the basic differences between the conventional languages and LabVIEW.

A Sample of LabVIEW Program Algorithm

Consider a simple algorithm that iteratively reads data from a file, converts it to °F, and warns the user (via an audible beep) if the value exceeds a predetermined limit. Figure 1.3 lists a FORTRAN program to achieve this objective. To help understand the structure of the program, a control-flow diagram is customarily used, such as the one shown in Fig. 1.4.

A program in LabVIEW is shown in Fig. 1.5 which achieves the same objective as the FORTRAN program shown in Fig 1.3. Notice that the LabVIEW program itself is in the form of a flow diagram, so no control flow diagram is needed. Further explanation of this program is provided in Table 1.1.

At first glance, it may appear that the FORTRAN program is easier to follow. This may be true for small programs like the one presented in this example. However, experience in writing and maintaining software shows that as the size and the complexity of the program increases, line-code programs become progressively more difficult to follow than a visual program in LabVIEW. This is because LabVIEW has some other features not depicted in Fig. 1.5. Two of its useful features are structure and data flow.

These features are described later in more detail, only a brief discussion follows. A structured language allows to write programs in a fashion that various strategic segments of the program are lumped together in logical units, thus improving the readability and understandability of the program. Unlike sequential processing (by default) in most other languages, a data-flow program executes in the order in which various program elements (or segments) receive their needed data. Since automation programming mostly involves operation on data, data-flow languages are particularly suitable. They allow easy comprehension and debugging of the program because the path and operations on each set of data can be easily followed as it is being transformed from cradle to grave. The graphical representation is a natural choice for a data-flow language to fully exploit the traceability of the loci of data migration.

Object-Oriented Programming (OOP)

The important concepts of object-oriented programming (OOP) are described below. This description is included to help readers visualize the significant advantages that this approach to programming offers over familiar line code languages such as Basic, FORTRAN, Pascal, and C. National Instrument's LabVIEW, the basis of NIPER's automation software, is a programming platform (compiler) that uses a high-level object-oriented language called "G."

	+	OPEN (5, FILE = 'ALLIANCE:LABVIEW 2:INPUT TEMPERATURE', STATUS = 'OLD') CELSIUS
C		ALLIANCE IS THE VOLUME I.E., HARD DISK, LABVIEW2 IS THE
C		FOLDER CONTAINING FILE NAMED INPUT TEMPERATURE
	10	READ (5, *, END = 99)
		FAHRENHEIT = ((CELSIUS * 9/5) + 32)
		IF (FAHRENHEIT.GT.450) GOTO 20
		GOTO 10
	20	CALL BEEP (1)
C		BEEP IS A SUBROUTINE THAT ACTIVATES AN AUDIBLE BEEP
	99	STOP
		END

FIGURE 1.3 - A FORTRAN program that iteratively reads data from a file, converts it to °F, and beeps if the value is higher than 450° F.

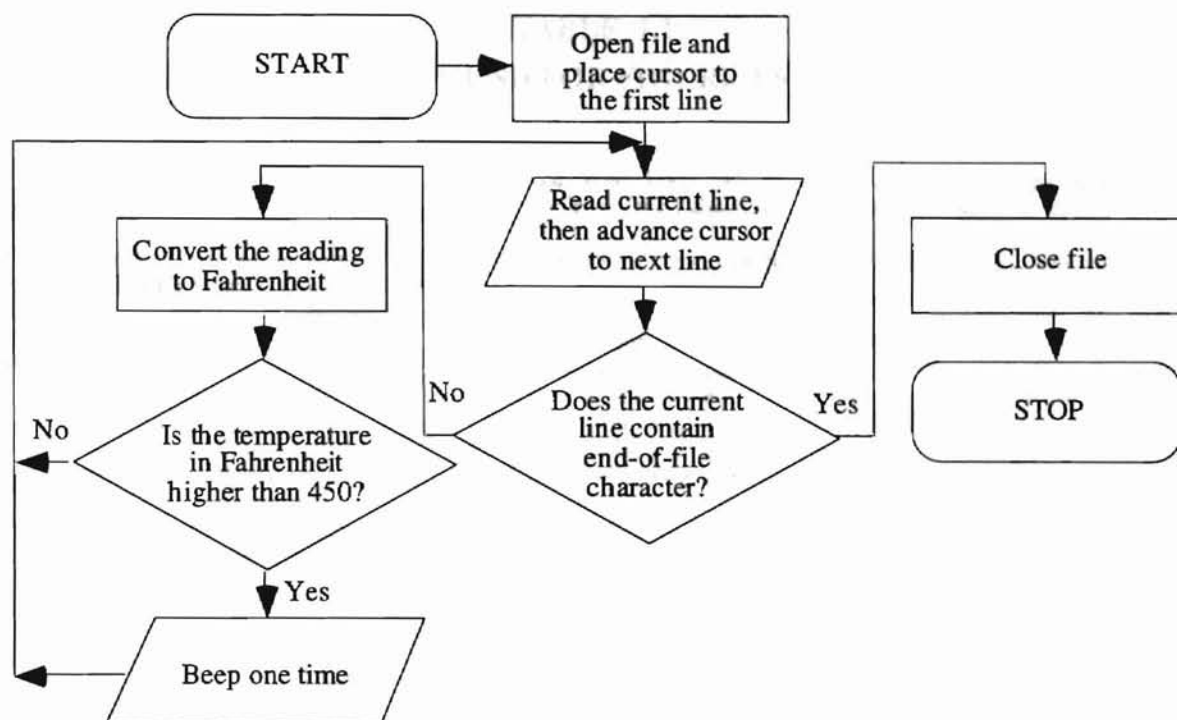


FIGURE 1.4 - The control-flow diagram of the example FORTRAN program shown in Fig. 1.3 that iteratively reads data from a file, converts it to F, and beeps if the value is higher than 450° F.

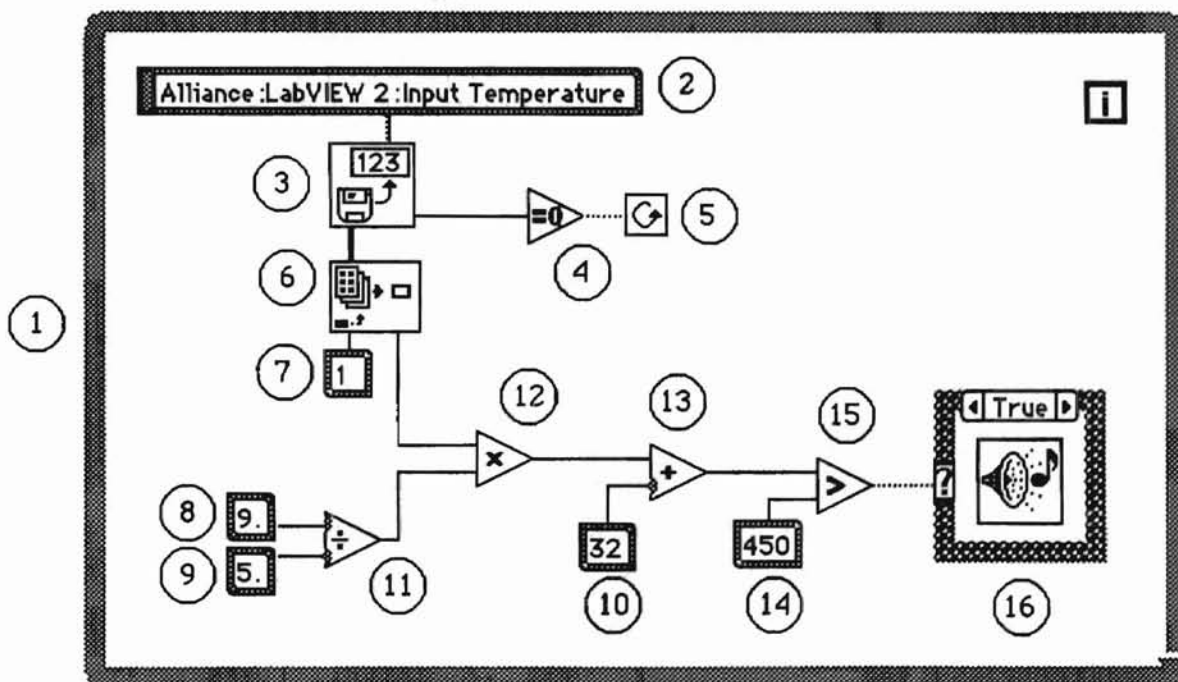


FIGURE 1.5 - A sample program segment in LabVIEW that iteratively reads data from a file, converts it to °F, and beeps if the value is higher than 450° F (see Table 1.1 for legend).

TABLE 1.1
LEGEND FOR FIGURE 1.5

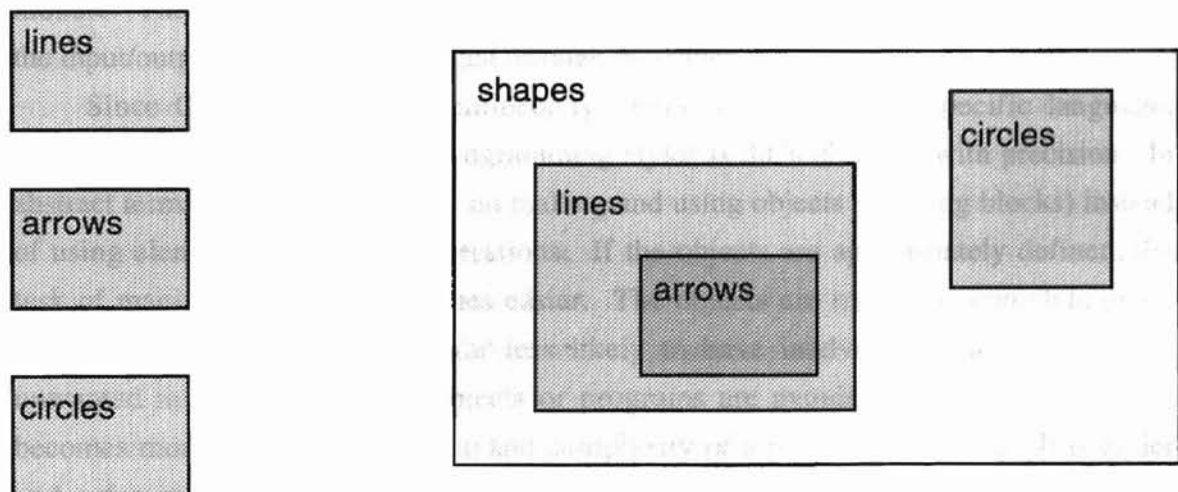
-
1. This entire box is a "While Loop" structure. The "While Loop", which can be thought of as a single node itself, executes the diagram inside the box until the Boolean (True-False) value passed to the conditional terminal (box 5) is False.
 2. This box contains the string which specifies the volume name, directory name, and file name from which the temperature data (in Celsius) is to be read.
 3. This node opens the specified file and reads the current line of temperature data into a numeric array. The node then closes the file, passes the array to node 6, and passes a numeric error message value to node 4.
 4. This node checks to see if the error message from node 3 is equal to zero (indicating no error). If so, the node outputs a Boolean value of True. If the error message does not equal zero, the Boolean output is False.
 5. This box is the conditional terminal for the entire "While Loop". The terminal is checked at the end of each iteration and exits the "While Loop" box once the Boolean value from node 4 is False.
 6. This node reads a specified element of the numeric array from node 3 and then passes the value of that element to node 12.
 7. This box contains the integer value that specifies which element of the array node 6 is to read.
 8. This box contains a numeric constant used in the temperature conversion from Celsius to Fahrenheit. The value is passed to node 11.
 9. This box contains a numeric constant used in the temperature conversion from Celsius to Fahrenheit. The value is passed to node 11.
 10. This box contains a numeric constant used in the temperature conversion from Celsius to Fahrenheit. The value is passed to node 13.
 11. This node divides the value from box 8 by the value from box 9. The numeric result (a conversion factor of 9/5) is then passed on to node 12.
 12. This node multiplies the output value from node 6 by the output value from node 11. The numeric result is passed on to node 13.
 13. This node adds the value from box 10 to the output value from node 11. The resulting numeric output (Fahrenheit temperature) is passed on to node 15.
 14. This box contains a constant numeric value specified as the temperature limit.
 15. This node checks to see if the output value from node 13 is greater than the temperature limit of box 14. If so, then the resulting Boolean output is True. If the value from node 13 is less than the temperature limit of box 14, then the Boolean output is False.
 16. This structure beeps if the Boolean output from node 15 is True, which means that the temperature value read from the file is higher than the temperature limit specified by box 14.
-

OOP refers to a programming style that relies on the concepts of inheritance and data encapsulation. Inheritance is a language facility for defining a new class of objects as an extension of previously defined classes. The new class inherits the variables and operations of the previous classes. Inheritance helps in building complex structures by using the existing simpler objects. Since common properties of objects can be preprogrammed by defining classes, programming effort can be significantly reduced. Data encapsulation (also called implementation hiding, meaning certain details of implementation code are deliberately hidden from the user) allows objects to be packaged so that unnecessary details of implementation are not visible from outside the object. An object may include a set of functions, procedures, subroutines, data, type-definitions, arithmetic and/or other operations. Any or all entries in an object may be defined as either public, private, or protected, depending upon their intended use. Objects can only interact with each other by sending and receiving messages.

The properties of OOP allow one module (an independent program segment like a subroutine) to be written with little knowledge of the code in another module. Modules can be reassembled and replaced without reassembling the whole system (program). OOP's programming style can be practiced with widely differing languages. For example, C++ (a line code OOP language) allows both inheritance and data encapsulation to deal with the most demanding systems' tasks yet retains C (also a line code language) as a subset for tasks requiring low-level programming.⁷ LabVIEW is also an OOP language that provides a simple yet powerful visual programming environment. It superimposes a graphical editing and execution system upon the object-oriented "G" language to create a platform for the users wherein the modules can be built by copying the objects from the LabVIEW library and user's own library of modules. Its libraries have essentially all of the objects needed for automation programming.

Procedure-Oriented Programming (POP) Versus Object-Oriented Programming (OOP)

Figure 1.6 illustrates the difference between procedure-oriented programming (POP) and OOP (Sethi, 1989). The requirement in this example is to build figures, which are basically composed of basic shapes—lines, rectangles and circles. POP will structure a program around the operations on shapes. This may include operations for drawing, rotating, and scaling a figure. For each shape in the figure, the procedure will classify the shape and then execute the code that is appropriate for drawing that kind of shape.



(a) Disjoint sets of objects

(b) Nested sets of objects

FIGURE 1.6 - Different ways of classification of objects: (a) POP (b) OOP.

However, the code for each shape will contain only very elementary operations. Because of the use of elementary operations, the code for manipulating shapes is spread across the various procedures, and this is often problematic. If a new shape is added, e.g., an arrow head, then the code for handling the new shape has to be added to each procedure. Even if the new information is small, it is spread across procedures, each of which must be analyzed before the new code is added to ensure that there are no conflicting directions or assignments.

The OOP approach of handling this problem is different: A class named "shapes" is defined, which has subclasses of lines and circles. Class shapes then collects common properties, such as the height, the width, the position, and an operation for moving the shape. Properties that are specific to lines, rectangles, and circles appear in the appropriate subclasses. Inheritance, an OOP property, allows arrows to be added by extending the subclass "lines," without touching the code for the other objects. An arrow inherits all the properties of a line (as a default), so the only additional code needed to draw an arrow is the code for drawing an arrowhead. Another property of OOP—data encapsulation—allows the drawing of various shapes by simply sending messages to the class shape, without the need to see its implementation details. In OOP, each module is a completely executable program in its entirety, and it does not interact with other program segments in any way except by receiving and sending messages. This message-passing mechanism is superior to the use of "subroutine calls" in traditional programming because it eliminates any chances of inadvertently altering the data in the calling program. The user does not need to know the implementation details of a module to use it in any other module. They merely need to know the abstract information about its actions and about the input/output data to be exchanged through messages.

Since OOP is more of a philosophy (methodology) than a specific language, differentiating OOP from other programming styles is difficult to do with precision. In abstract terms, OOP relies heavily on making and using objects (building blocks) instead of using elementary units and operations. If the objects are appropriately defined, the task of manipulating them becomes easier. The objects are treated as complete units; hence, operations on them are far less likely to have inadvertent side effects, i.e., unwanted influences on other objects or programs are avoided. The power of OOP becomes more apparent as the size and complexity of a program increases. It is easier and safer in OOP to extend classes, and the code for each individual operation is relatively small, often to simply "pass the buck" by invoking operations in other objects (Sethi, 1989).

Programming Structure of LabVIEW

National Instruments LabVIEW, by using the high-level, object-oriented language "G," allows programming in a highly structured and modular fashion. Each module (also called "VI" by National Instruments) is a totally independent, interactively executable program which can be used as a subprogram by other modules. To use a module as a subprogram, its icon is copied into the program. Copying an icon into a program is analogous to a subroutine call. Data exchange (input/output) with the module can be accomplished by making wired-connections between the terminals shown on the icon and other elements in the program. Large programs can be developed in an hierarchical manner by starting with small modules and using them within other modules. Since each module has its own independent program and a separate input/output interface, debugging or modifying a large program is quite easy.

LabVIEW handles the execution sequence of a program in a different manner than line-code languages. In traditional line-code languages such as Basic, FORTRAN, C, or Pascal, the execution of instructions (statements) takes place sequentially by default; the order is changed occasionally according to the control-flow diagram designed by the programmer. LabVIEW, on the other hand, is based on a modified dataflow model such that the sequence of execution need not be predefined.

Dataflow diagrams specify the data dependency between computations, but they do not specifically force any particular sequence of independent computations. The dataflow diagram of Fig. 1.1 is an example. It is a directed, acyclic graph consisting of nodes, arcs, terminals, and data tokens. Terminals are the connections to the external world, and act as the sources, or sinks, of data tokens. Arcs are the directed paths over which data tokens move, and nodes are the locations in which computations (or other instructions) are performed. A node consumes tokens on its input arcs and produces new tokens on its output arcs. The diagram becomes data-driven because of its *firing rule*, which states that a node cannot execute until all of its input arcs have a data token available, at which time the node consumes one token from each input arc, performs the computation or instruction, and produces one token for each output arc. In Fig. 1.1, for example, node J has already executed, K and L are eligible to execute, and M is still ineligible because it needs a token on its second input.

In contrast to the control-flow model, the dataflow model has no concept of locus-of-control, no program counter (i.e., no sequence numbers like in text-based program codes), and no global variables (globally accessible memory). A data token exists only from its production by a node or input terminal to its consumption by another node or

output terminal. All nodes that are eligible to execute can do so in any order or even in parallel; the results of the diagram will be the same in all cases.

The classical dataflow model, however, lacks the provisions for conditional or iterative computations. LabVIEW provides an extension to overcome this limitation. This extension not only preserves its firing rules and acyclic structure (thus, preserving program clarity), but also incorporates the proven benefits of the structured programming methodology. This extension involves redefining a node to be any program segment enclosed in a box-like structure that separates the body (or inside) of the structure from the rest of the program. Because the box behaves like a node as far as the rest of the program is concerned, the overall dataflow methodology is preserved. The body of a node (inside the box structure) behaves like an isolated diagram, in which access to the code is only from the top (or beginning). The program structure-semantics such as loop behavior or conditional behavior have been superimposed on the body of the box. This can be thought of as a macro structure (program as a whole) containing some micro structures (program segments inside the node), both of which independently follow the dataflow model. The micro structures, however, have some additional control properties.

Using the extended dataflow strategy, LabVIEW is able to retain the important benefits of both structured programming and dataflow strategy. Furthermore, the performance of the executable code generated by compiler is comparable to that produced by a C or Pascal compiler.

Basic Facilities in LabVIEW

Writing and Editing Programs

LabVIEW contains three interrelated editors, one for each of the three parts of a module: block diagram, front panel, and icon/connector. The front panel is the means of controlling the execution of the module (VI) and interacting with the program. It specifies the inputs and outputs of the module and is analogous to a front panel of a real instrument or an operators panel controlling a cluster of instruments. It is also analogous to the data declaration and type definition section of a subroutine. The data input terminals are called controls, whereas data display terminals are called indicators. An example panel is shown in Fig. 1.7, which is a rendition of the "NIPER MAIN FACILITY" panel of the NIPER automation program (some of the features removed for clarity). Each front panel has an accompanying block diagram. The block diagram is the actual program (source code) created by the user with LabVIEW. An example of a block diagram is Fig. 1.8 (this is the accompanying block diagram of the panel shown in

Test Status Display

SUMMARY FOR DIAGNOSTIC

3	Item Name	Surfactant Tank	500.00	Cutoff Higher Limit
Frame No.	3	Item no	10.00	Cutoff Lower Limit
EMERGENCY?	173.00	Value	20.00	Cutoff Accel. (% per min)
	8.30	Acceleration	20.00	Cutoff Decel. (% per min)
WARNING?	-8.30	Deceleration	400.00	Warning Higher Limit
	187	Prev. Value	200.00	Warning Lower Limit
	20	Elapsed Time	15.00	Warning Accel. (% per min)
			15.00	Warning Decel. (% per min)

ERROR MESSAGES

No Error.

PRE-EMERGENCY ADJUSTMENTS

0	5	Item to be adjusted
Frame No.	999.00	% value to be set
	Warning Limit Type	
	Lower	Higher

CHANNELS WITH WARNING AND/OR EMERGENCY CONDITIONS

Item with Problem	No.	Driver Error(s)	Too High	Too Low	Hi
Surfactant Tank	3			173	

Figure 1.7 - NIPER MAIN FACILITY panel.

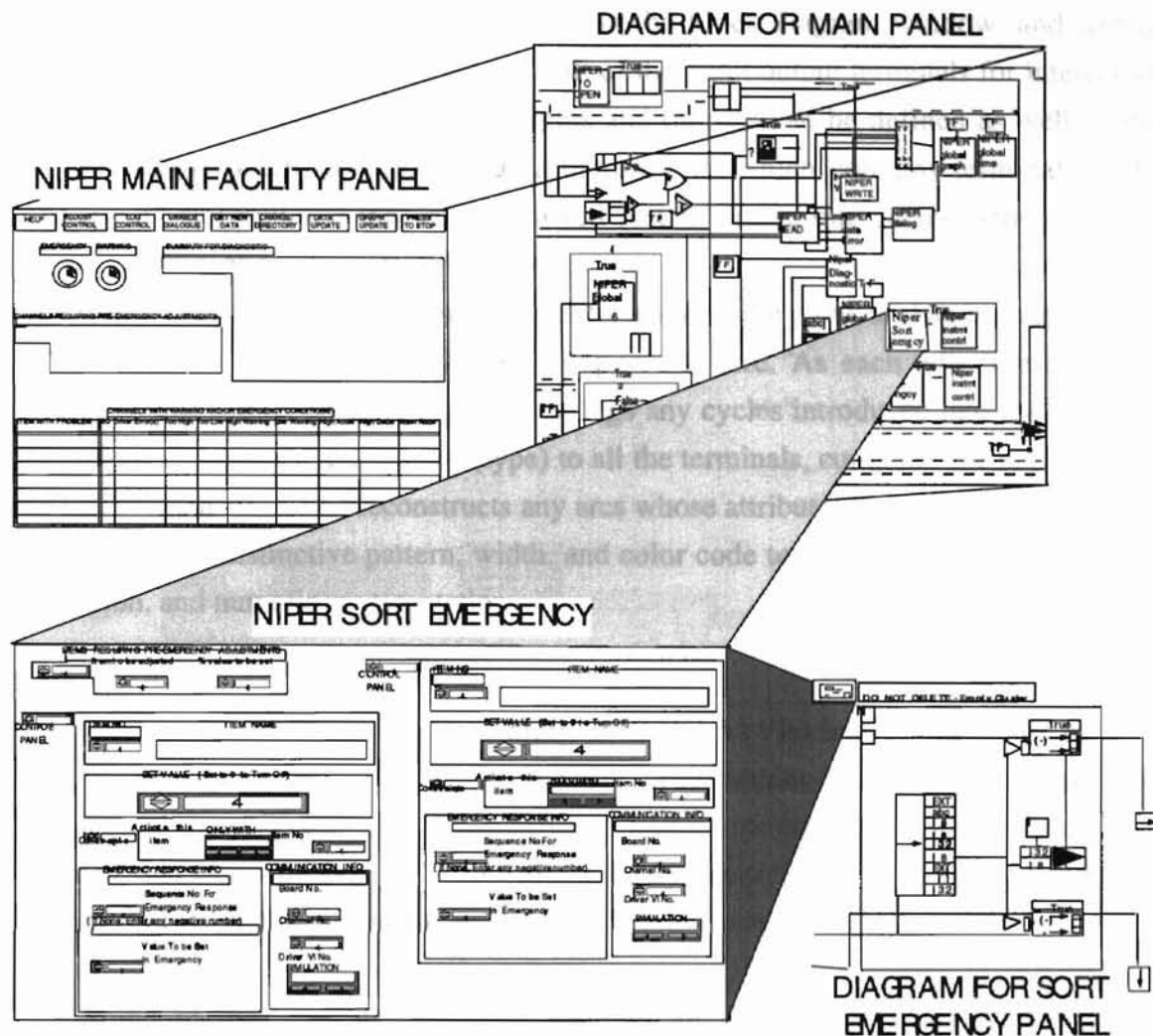


FIGURE 1.8 - Descriptive hierarchy of programming for the NIPER MAIN FACILITY showing panels, diagrams, and icons for select portions of the program.

Fig. 1.7) with one of the icon/connectors expanded with its panel and block diagram icon icon/connectors expanded with its panel and block diagram icon to show the hierarchical structure. The icon/connector is a means of turning a module (VI) into an object that can be used in the block diagrams of other modules (VIs) as if it were a subroutine. The icon graphically represents a subroutine call in the block diagram of other modules. The connector terminals on the icon of a module determine where the inputs and outputs must be wired (connected) on the icon. The terminals are analogous to parameters of a subroutine, except that their location can be arbitrarily fixed in any order.

The block diagram is a directed acyclic graph containing nodes, interconnecting wires, and source/sink terminals corresponding to the front panel controls and indicators, respectively. It is constructed by copying built-in functions, structures, and previously

constructed modules (VIs), arranging them in the block diagram window, and wiring them as needed. The front panel contains all of the input/output terminals for interactive programming. The data types of the inputs and outputs can be defined as well as the default values. Each front panel has a unique icon which contains terminals (non-overlapping sub regions) that can be linked (set in one-to-one correspondence) with various controls and indicators on the panel.

The wiring is done using a wiring tool to establish the paths of data exchange. The wiring tool is a cursor that looks like a spool of wire. As each edit transaction is performed, the syntax checker detects and flags any cycles introduced into the dataflow diagram, propagates data attributes (type) to all the terminals, computes the data type for each built-in function, and reconstructs any arcs whose attributes have changed. Each arc is drawn with a distinctive pattern, width, and color code to indicate the data type, array dimension, and numeric representation.

Built-in Library

LabVIEW comes with a large library of modules (VIs) that can take care of most low-level programming details, freeing the programmer to concentrate more on customizing the program. Numerous driver programs for common instruments are also included, and many are available from a growing number of third-party vendors and user-groups. For some unique instruments, the user may have to write their own instrument drivers.

Some of the icons/connectors are shown in Fig. 1.9 as an example. Since LabVIEW is a programming environment, familiarity is acquired by practice and experience. National Instruments provides adequate training material as part of the purchase of the LabVIEW.

Control Structures

LabVIEW provides five box-like control structures and one file-linking structure as shown in Fig. 1.10. The legend for Fig. 1.10 (numbers in circles) is listed in Table 1.2. The top three are quite similar to the "for loop," the "while loop," and the "case structure" (case selector) used in other programming languages, while the sequencer, the formula node, and the Code Interface node are used as follows: to impose an order on execution, evaluate text-like expressions, and link the program to external subroutines. Structure "A" depicts a "Numeric Iteration." This structure is comparable to a "For Loop" in other languages. It executes a specified number of times. The "Repeat Until" structure, or "Conditional Loop," is displayed in B. This loop continues execution as long as the

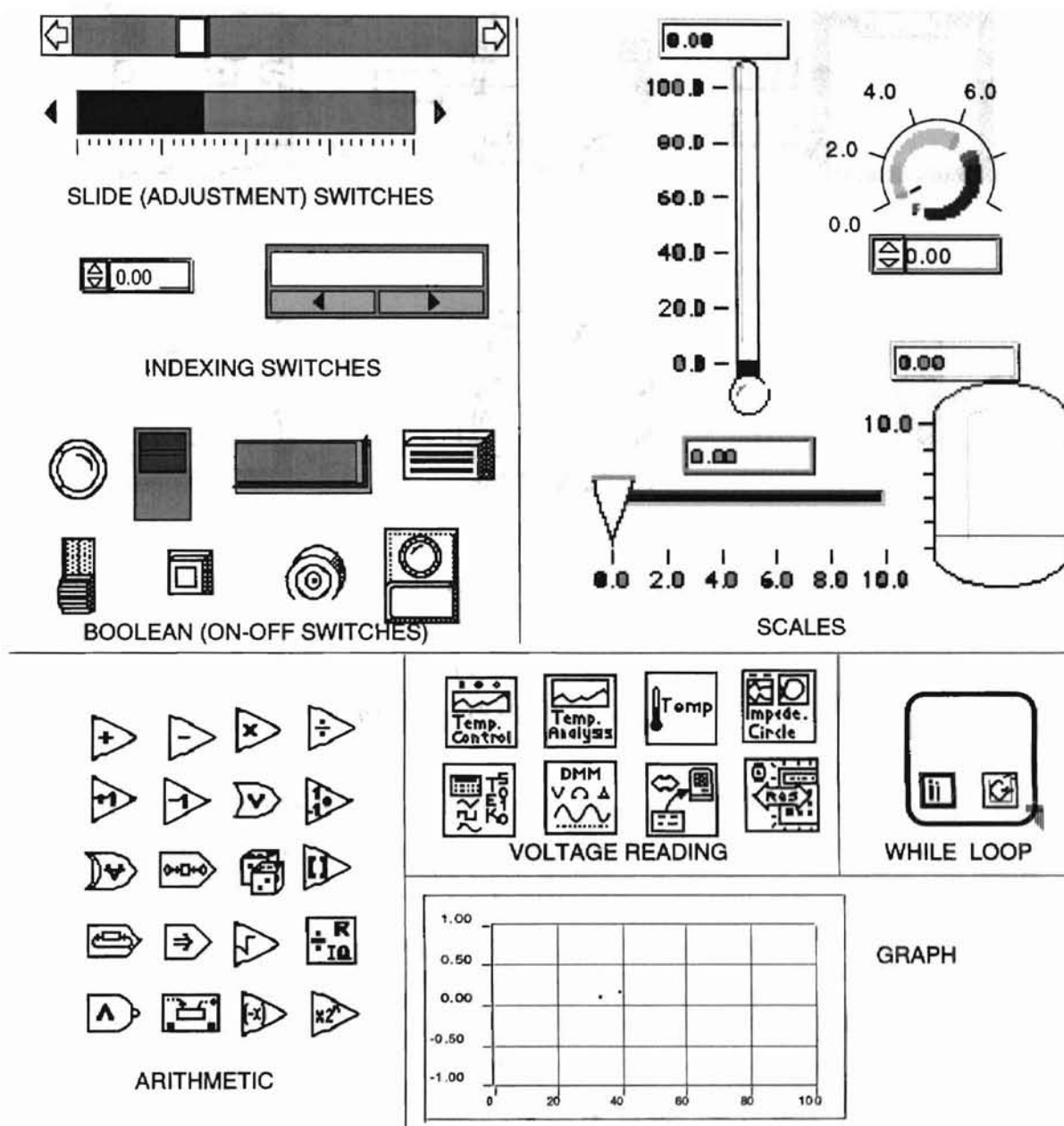


FIGURE 1.9 - Example of icons from LabVIEW Library.

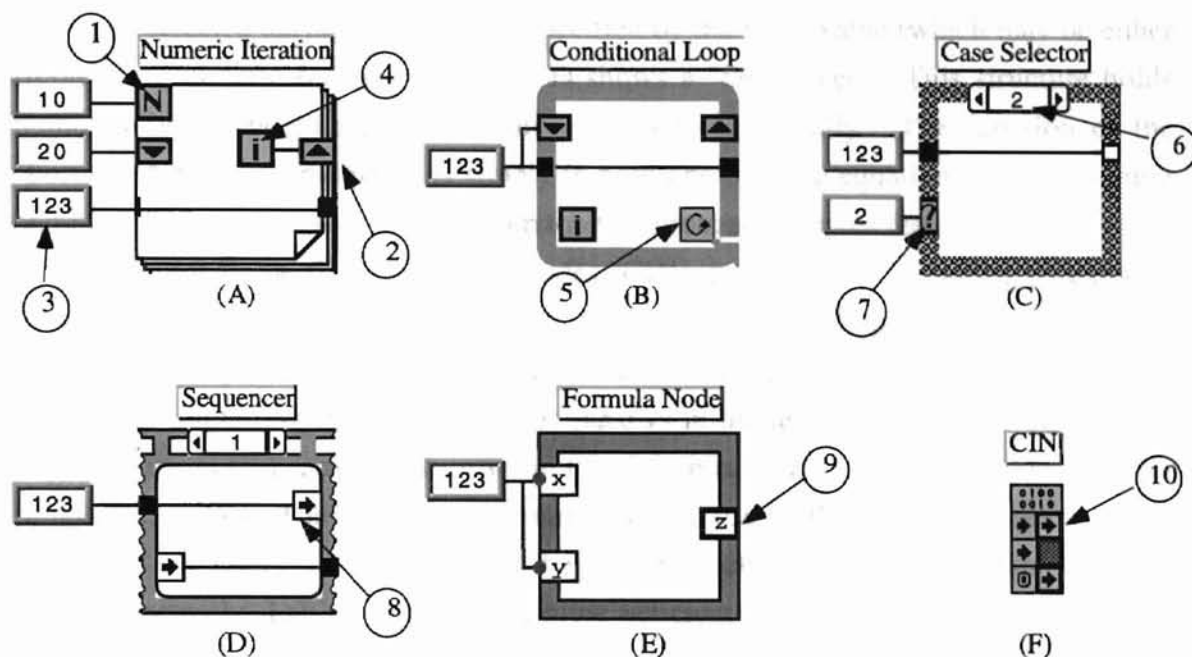


FIGURE 1.10 - Basic control structures in LabVIEW (see Table 1.2 for legend).

TABLE 1.2
LEGEND FOR FIGURE 1.10

1. The N holds the count value (supplied by the constant wired to it) of how many times the loop is to be executed.
2. The arrows on the vertical edges are shift registers, which are used to pass values from one iteration cycle to the next. Any data stored in the up-arrow (right side) is available at the beginning of next iteration cycle through down-arrows (left side). The down-arrow may be initialized by wiring values to be used in the first iteration cycle.
3. This is an example of parameter passing through the structure. A numeric constant, the value 123, is being passed to be used in the structures.
4. The i is the iteration count symbol. It holds the current number of completed cycles.
5. This curved arrow is the loop terminator. It receives the Boolean value of the test condition at the end of each iteration. If the test value is false, it finishes the iteration.
6. This window shows the case being displayed. The right and left arrows allow user to observe different sub diagrams, or cases. The cases may be numeric or Boolean.
7. The question mark receives the case selection information; i.e., the case to be executed.
8. The arrows inside the frame hold the local variables. These variables are used to pass data from one frame to the subsequent frame. The inward arrow indicates a local variable which is wired to receive the value, whereas the outward arrow indicates a local variable which already has a value which can be distributed.
9. An example of the parameter passing method for the formula node. Parameters may be passed through the vertical edges. In this example, z is the output variable and x and y are input variables.
10. The "Code Interface Node," CIN, allows an external routine written in "C" or "Pascal" to be executed here. The parameter can be passed in or out through inward or outward arrows.

specified Boolean condition is true. Since it checks for the true or false condition at the end of each cycle, it always executes at least once. The "Case Selector," shown in structure C, may contain one or more sub diagrams, also called "cases." One of these

"cases" is selected during execution as specified by the input value (which may be either Boolean or numeric scalar). Structure D shows a "Sequencer." This structure holds numerically numbered frames that are executed sequentially. The function of the "Formula Node" in structure E is simply to hold one or more equations. This structure computes the equation sequentially from top to bottom and outputs the result.

Structure F, the Code Interface Nodes (CIN), is equivalent to calling an external subroutine that is accessible to the program yet is outside the program body itself. The subroutine is imported during run-time and evaluated using the input parameters provided through the input-terminals on the nodes (arrows in the left boxes). The result is then copied to the output-terminals (arrows in the right boxes) where it is available for other program elements. The CIN allows programmers to use C, Pascal, or an assembly code language, while still enabling them to benefit from object-oriented programming; but most importantly, LabVIEW allows a more efficient dynamic memory allocation for arrays and strings and minimizes memory fragmentation.

On-line Help

Information about any of the sub-modules can be conveniently obtained by pointing to its icon with a wiring tool and then pressing \mathbb{H} -H (or selecting help from the menu bar). The type of information available on-line includes the name of a module, a brief description of its intended use, and a brief visual/textual description of input/output data and terminal locations. For more details, both the front panel (input/output displays) and the block diagram (program) can be viewed by double-clicking on the icon of a module. The entire hierarchy of a module (or a part of it) can also be conveniently viewed.

BIBLIOGRAPHY

- Dijkstra, E. W., 1972. Notes on Structured Programming in O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare *Structured Programming*. Academic Press, London, pp. 1-82.
- Kirkman, I. W. and P. A. Buksh, 1992. *Data Acquisition and Control Using National Instruments' "LabVIEW" Software*. Rev. Sci. Instrum., v. 63, No. 1, January, pp. 869-872.
- Kodosky, J., J. MacCracken and G. Rymar, 1991. *Visual Programming Using Structured Dataflow*. Proceedings of the 1991 IEEE (Institute of Electrical and Electronics Engineers) Workshop on Visual Languages, Kobe, Japan, Oct. 8-11. Reprinted by IEEE Computer Society, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264.
- Liles, K., 1991. *Data Acquisition Software Automates Automotive Fuel Injector Test Facility*. Engineering & Management, October/November, pp. 18-21.

- Liles, K., 1992. *Data Acquisition—A Mac-Based System for Fuel and Lubricant Testing*. Scientific Computing & Automation, January, pp. 19-23.
- National Instruments Corp., 1991a. *LabVIEW 2—Getting Started Manual*, Part No. 320246-01, Austin, TX, April.
- National Instruments Corp., 1991b. *Training In-Depth Course on LabVIEW 2*, Version 1.4, Part No. 776393-01, Austin, TX, April.
- National Instruments Corp., 1991c. *LabVIEW 2—User Manual*, Part No. 320244-01, Austin, TX, September.
- Sethi, R., 1989. *Programming Languages: Concepts and Constructs*. AT&T Bell Laboratories, Murray Hill, NJ, Addison-Wesley, Reading, MA.
- Stroustrup, B., 1986. *The C++ Programming Language*. Addison-Wesley, Reading, MA.

(This page intentionally left blank)

Chapter 2

AN OVERVIEW OF NIPER Lab WARDEN FACILITIES

The following is a brief description of NIPER's Lab WARDEN automation software. It consists of three main interactive sections to provide a central data management and instrument control system. These sections are: "NIPER MAIN FACILITY, NIPER DISPLAY FACILITY and NIPER GRAPHIC FACILITY." All three of these facilities are independent but integrated such that while the user interacts with one, the others may process in the background when desired. Facilities can be easily alternated by simply pointing and clicking at the selected facilities. For example, the "NIPER MAIN FACILITY" panel may be activated during the actual scanning of data from pumps, pressure transducers, flow meters, etc., to display the acquired data and report any errors. Then the "NIPER GRAPHIC FACILITY" can be used to display the data graphically. "NIPER DISPLAY FACILITY" panel may be activated to give a visual picture of the process. The major functional units of the program, the hierarchical structure, and the components in relationship to National Instruments LabVIEW® are shown in Figs. 2.1 and 2.2.

NIPER MAIN FACILITY

This facility has several features: (1) to read and control instruments; (2) to monitor instruments and when appropriate, issue system errors; (3) to allow reconfiguration of system set-ups, e.g., connect or disconnect instruments, define or redefine allowable ranges outside of which a warning is issued and the system is shut down, establish emergency shutdown sequence, define corrective actions when instrument values fall in the warning range, and set constraints so that instruments are not operated illogically. All of these features are controlled through the "NIPER MAIN FACILITY" panel shown in Fig. 2.3. By pointing and clicking on the buttons in the title bar (see Fig. 2.3), the user may select certain features. Some of these buttons open a new window, allowing new selections. Since these sub-windows have lower priority, they do not interfere with other activities, i.e., data acquisition and control functions continue in the background.

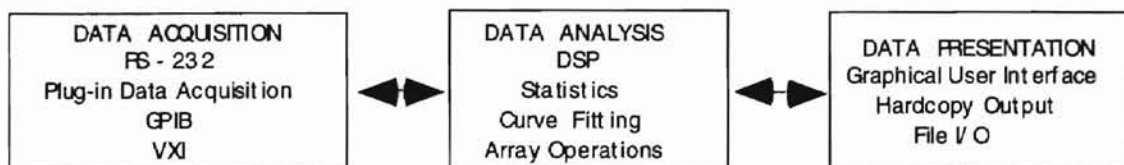


FIGURE 2.1 - Major functional units of NIPER's automation program.

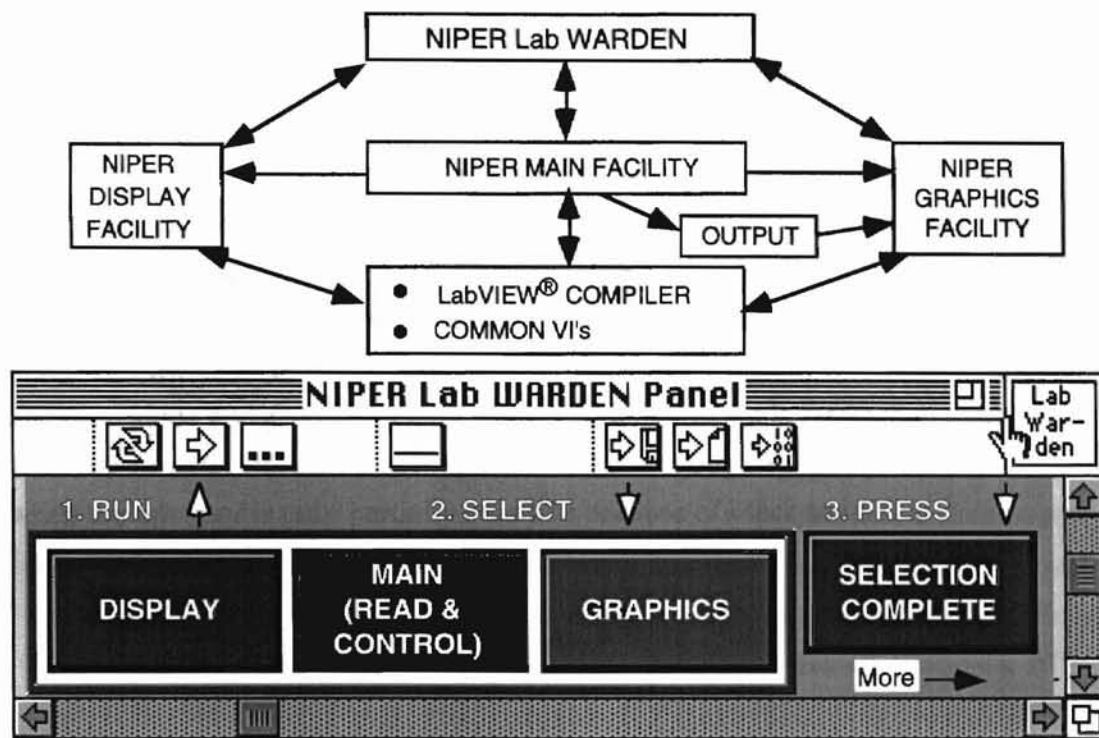


FIGURE 2.2 - Hierarchical structure and front panel of NIPER Lab WARDEN.

The NIPER MAIN FACILITY panel displays various control and monitoring elements:

- Buttons:** HELP, ADJUST CONTROL, LOG CONTROL, ENABLE DIALOGUE, GET NEW DATA, CHANGE DIRECTORY, DATA UPDATE, GRAPH UPDATE, PRESS TO STOP.
- Emergency/Warning Indicators:** Two circular indicators labeled EMERGENCY and WARNING.
- SUMMARY FOR DIAGNOSTIC:** A section with input fields for Surf. Tank, Item No, Emergency, Warning, Value, Acc, Dec, Previous Value, and Elapsed Time. It also includes cutoff and warning limits for acceleration and deceleration.
- CHANNELS REQUIRING PRE-EMERGENCY ADJUSTMENTS:** A section with a dropdown menu and a table for adjusting items.
- CHANNELS WITH WARNING AND/OR EMERGENCY CONDITIONS:** A table with columns for ITEM WITH PROBLEM, NO, Driver Error(s), Too High, Too Low, High Warning, Low Warning, High Accel, High Decel, and Warn Accel.

ITEM WITH PROBLEM	NO	Driver Error(s)	Too High	Too Low	High Warning	Low Warning	High Accel	High Decel	Warn Accel

FIGURE 2.3 - NIPER MAIN FACILITY panel.

NIPER DISPLAY FACILITY

The front panel for this facility is shown in Fig. 2.4. "NIPER DISPLAY FACILITY" allows the user to pictorially view the status of remote-sensing instruments such as pumps, controllers, balances, temperature sensors, pressure sensors, fluid flow controllers, and alarms through the computer panel. Besides pictorial representation, the data, warning messages, and corrective actions can also be viewed in numerical form.

"NIPER DISPLAY FACILITY" is an optional facility that does not need to be running during automation process (reading/controlling/monitoring instruments). Its main usefulness is in quick recognition of possible trouble-spots that need immediate attention since a visual examination of the status of items is faster than extracting it from a cluster of data. Its use is recommended when enough RAM memory is available and when the operator is only partially attentive because of other activities.

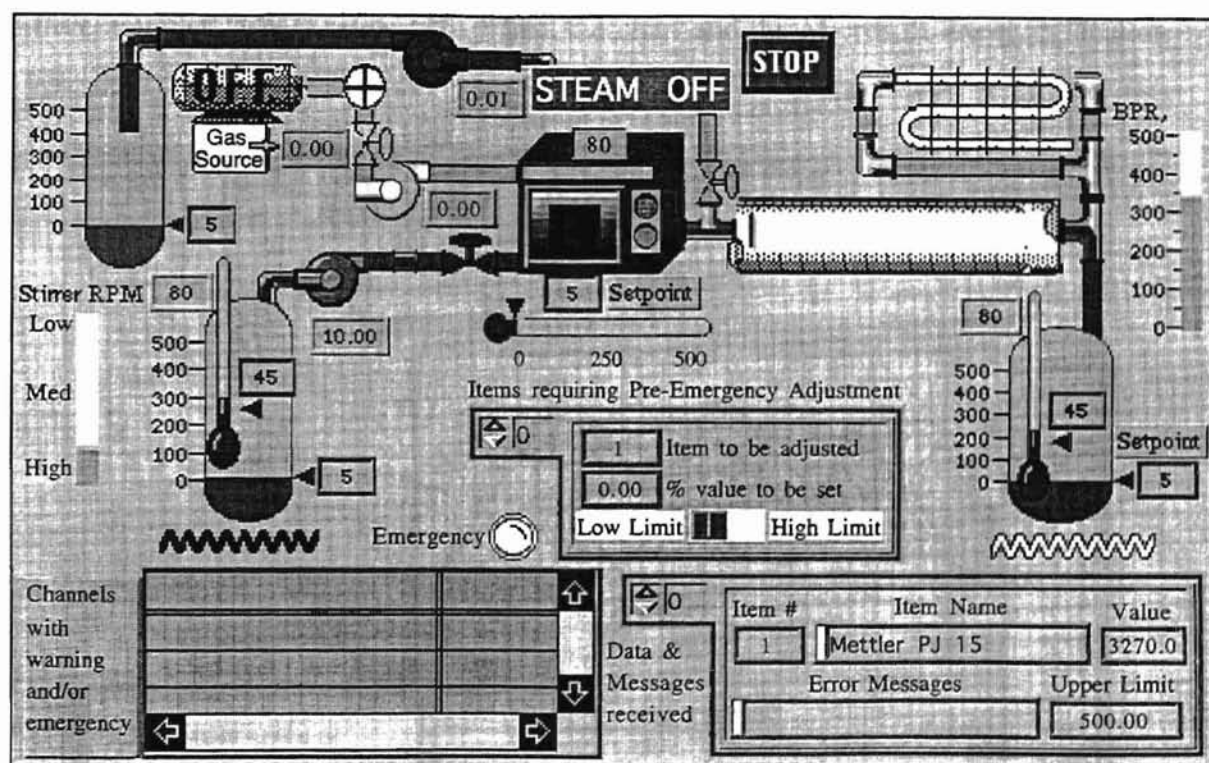


FIGURE 2.4 - NIPER DISPLAY FACILITY panel.

NIPER GRAPHIC FACILITY

The front panel for the NIPER GRAPHIC FACILITY is shown in Fig. 2.5. This facility allows the user to select or manipulate any portion or combination of data from a current or previous experimental run, e.g., analyze, compare, plot, curve-fit, add, delete, and copy interactively. The selected data can be printed or exported to other programs such as Microsoft Excel. Data can also be sent to a full-size graph where they can be edited using a mouse, i.e., a point or a segment of curves can be deleted, relocated, added, or copied/pasted from another graph. This interaction with data for graphical display or printing can be conducted independently, or during runtime while the program is collecting data and monitoring the process.

The 3-D graphic display shown in Fig. 2.6 is an example of the data export and communication capability with other applications. In this example, the data were reviewed in "NIPER GRAPHIC FACILITY" and then exported by pressing the "External Graphic" button, which opened a Microsoft Excel file containing the new data. The data were plotted, and the chart was automatically rotated with different attributes such as different angles, aspect ratios, etc. Then, the chart and the Microsoft Excel application

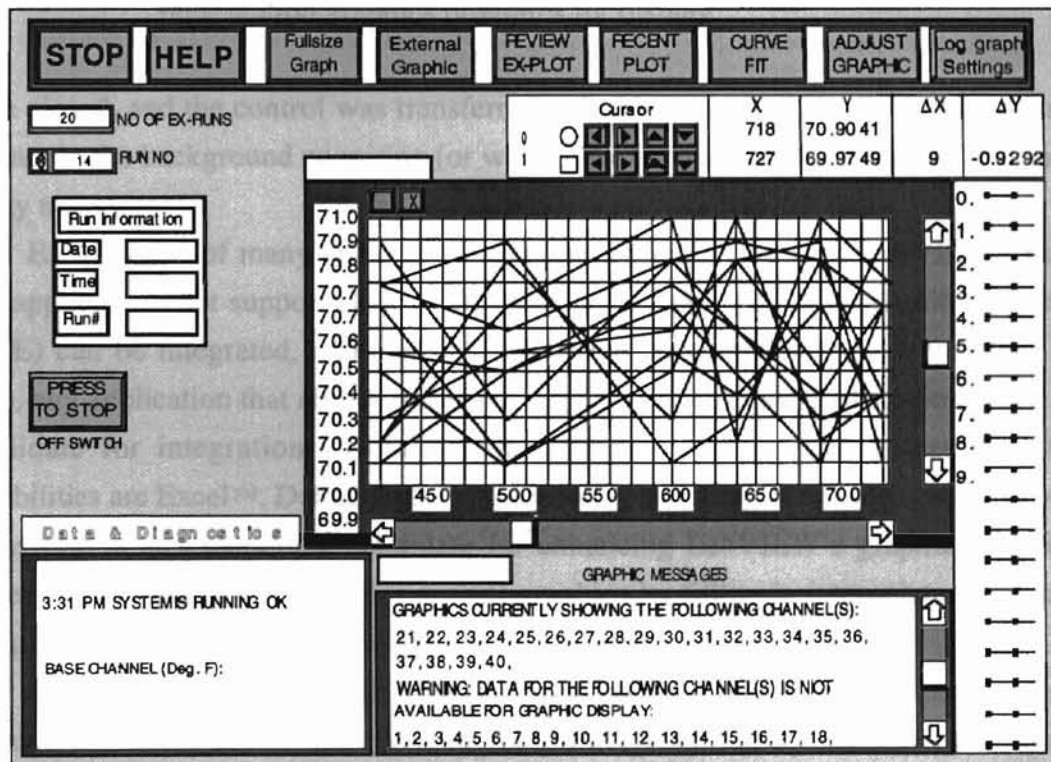


FIGURE 2.5 - NIPER GRAPHIC FACILITY panel.

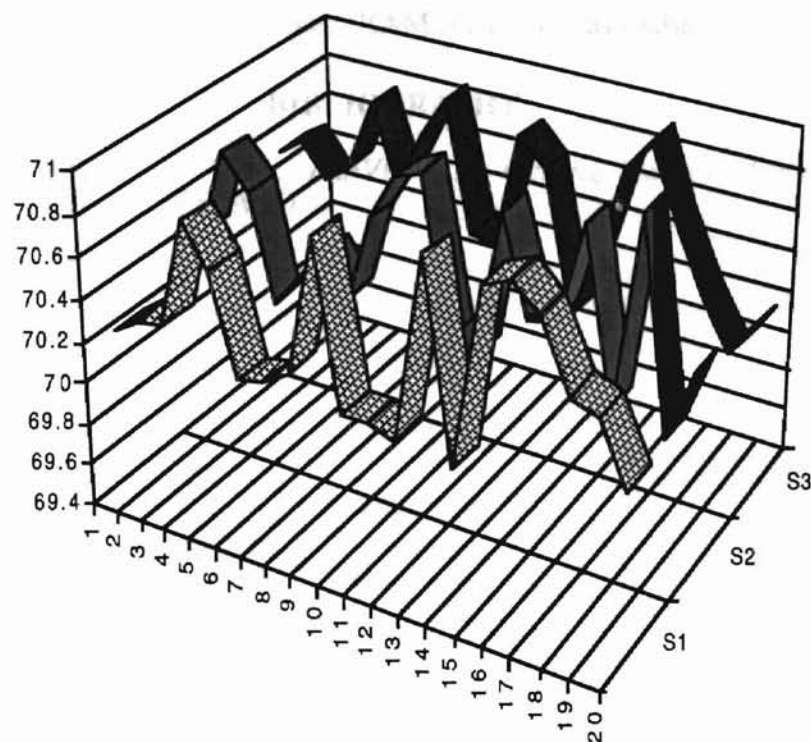


FIGURE 2.6 - A sample snapshot of automatic 3-D display of data in Microsoft Excel. DeltaGraph Professional and Spyglass Transform are similar linkable presentation graphics programs for display.

were closed, and the control was transferred to LabVIEW. Lab WARDEN continues to operate in the background acquiring (or waiting for) new data and checking for error and safety messages.

Excel is one of many applications that can be integrated with LabVIEW. In fact, any application that supports Microsoft System 7's feature of Dynamic Data Exchange (DDE) can be integrated, and can link, subscribe, and publish to or from other files. Also, any application that allows macro-expansion (sometimes called scripting) is a good candidate for integration. Some of the commercial applications that have these capabilities are Excel™, DeltaGraph Professional™, and Spyglass Transform™. Spyglass Transform is an exceptional candidate for enhancing LabVIEW's graphic capabilities since it can display diffused-color surreal graphics by filling in interpolated data points along with the actual data in a multidimensional spatial field.

"NIPER GRAPHIC FACILITY" is also an optional facility, i.e., it does not need to be running during automation (reading/controlling/monitoring instruments). Its sole purpose is to provide off- or on-line (real time) data selection capabilities and advanced

graphics for review and printing purposes. The use of this facility should be minimized while a test is in progress, unless enough RAM memory is available.

BIBLIOGRAPHY

National Instruments Corp., 1991. *LabVIEW 2—Getting Started Manual*, Part No. 320246-01, Austin, TX, April.

Chapter 3

DESCRIPTION OF NIPER MAIN FACILITY

Features

This software (facility) allows the user to:

- (1) Configure the software for the desired experimental set-up,
- (2) Save several experimental configurations for future recall,
- (3) View and recall a previously saved configuration for use in the current run,
- (4) Scan and view data from the selected instruments at fixed intervals or as desired,
- (5) Control (change settings) selected instruments at start, and set stop, warning, emergency conditions as desired,
- (6) View complete status of the test including warning and error messages,
- (7) Provide data and run status information to other NIPER facilities and commercial programs.

“NIPER MAIN FACILITY” is the core of the software program that enables the user to interface with the instruments (read data, send control commands, define the sequence of instruments to be set at startup/shutdown/emergency, reset instruments during the run, analyze raw data received from the instruments, compare the data with the user-defined ranges, and display the status of each instrument (warning, error, and normal conditions). It also provides two safety features: it can correct a warning situation by adjusting the value of certain instruments according to the user specifications, and it can systematically shut-down the experiment by following the prescribed sequence.

“NIPER MAIN FACILITY” is a self-sufficient facility that does not require any other NIPER facility to accomplish its features. When run, it asks the user to provide information about the instruments and the design (control) parameters of the experiment. Such information can be saved and retrieved freely, avoiding repetition. This allows multiple experimental set-ups to be handled easily by a single program, by saving and recalling the settings for each set-up.

The facility provides complete information about the status of the test. This is achieved by comparing the recently acquired data of each instrument with respect to its defined limits and the instrument's previous value. Besides providing audio-visual warnings and error messages to the operator screen, such messages can be sent to remote terminals. The facility can be configured to take corrective actions when the value of an instrument exceeds user preset limits (warning limits). If a critical condition exists as defined by the user in their configuration, the facility may automatically initiate a

user-defined systematic shut-down procedure. The facility also allows the user to define a start-up sequence, i.e., the reset value (or ON/OFF status) and its position in the sequence can be defined for as many instruments as desired. Similarly, the user may also define a shut-down sequence to be initiated at the termination of the test.

Technical Information for Programmers

The program executes in three stages: (1) initialization, when files are opened and are initialized to the default parameters, and the control-instruments are reset to their start-up values sequentially; (2) a loop, in which the program handles service requests such as opening the front panels for user interaction; and (3) termination, where the system is shut-down systematically, i.e., the control-instruments are reset to their stop values sequentially, files are closed and the current status is saved. This stage begins when the "STOP" button is pressed. At the beginning of each loop, the program notes the lapse time after the last reading. If this lapse time has exceeded the user-defined scan interval, or if the "GET NEW DATA" button on the front panel is pressed, data from each instrument are read and saved in one of the open files. These data are then analyzed and compared to the defined ranges. The analysis of data and emergency/warning conditions are reported on the front panel. If an emergency condition exists on any item, all control-instruments are reset to their emergency status as defined in the "NIPER CONTROL PANEL" VI. If a warning condition exists on one or more items, the corrective actions are taken according to their preset sequence defined in the "NIPER INDICATOR PANEL" VI. These corrective actions involve resetting the control-instruments to an adjusted value.

Since the data are stored in files as soon as they arrive, a copy is immediately available for user handling. The user can perform any operation on the data using any software that handles "tab-delimited-text" format, e.g., Microsoft Word and Excel. Manipulating these data does not interfere with LabVIEW operations such as data acquisition and instrument control. The original file containing the data is locked and not available for editing while the program is running.

A description of the "NIPER MAIN FACILITY" panel is given in Fig. 3.1. The circled reference numbers refer to the description of various attributes described in Table 3.1. When "NIPER MAIN FACILITY" is operated, a new window named "NIPER Indicator" is opened whose front panel is shown in Fig 3.2. The circled reference numbers refer to the description of various elements presented in Table 3.2. Clicking on the "Accept Changes?" or "Reject Changes?" opens the "NIPER Control"

panel (Fig. 3.3). Finally, clicking on "Accept Changes" or "Reject Changes" closes this window and re-displays "NIPER MAIN FACILITY" panel.

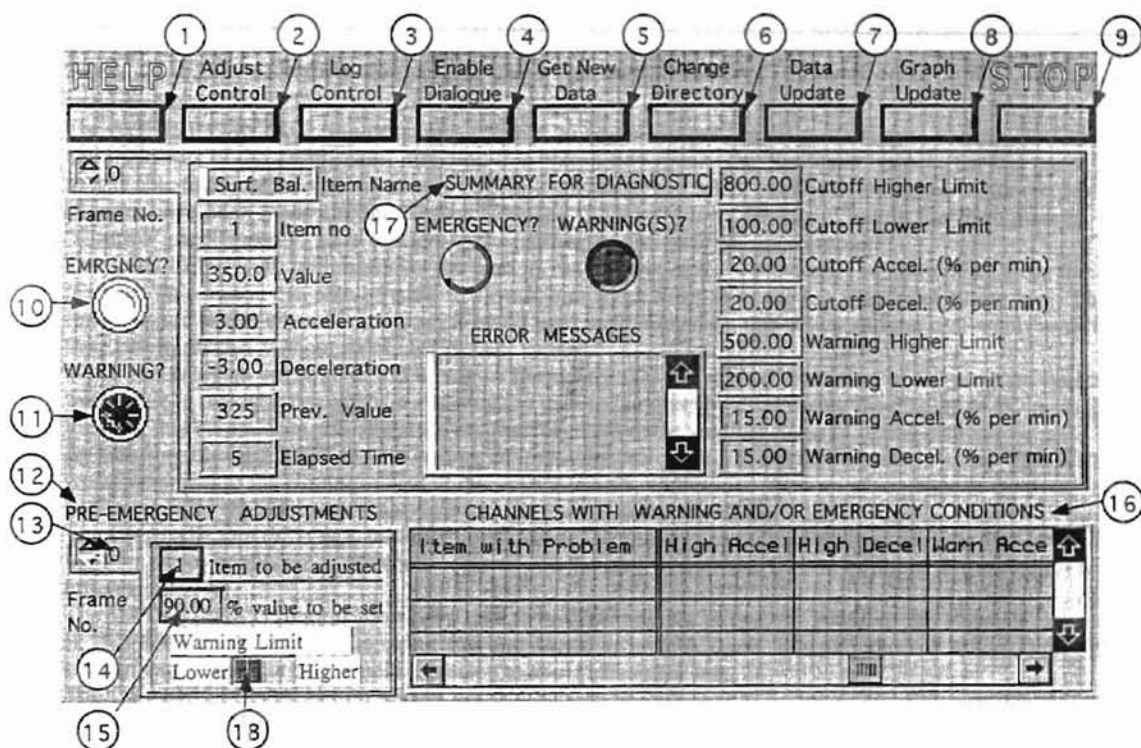


FIGURE. 3.1 - Front panel of "NIPER MAIN FACILITY."

TABLE 3.1
LEGEND FOR FIGURE 3.1

1. This button opens the NIPER HELP panel which provides information about LabVIEW and object-oriented programming.
2. The ADJUST CONTROL button allows the user to adjust the parameters pertaining to various instruments and indicators of the experiment. Pressing this button first opens the "Indicator" panel. On this panel, the user is able to adjust the various controls of the experiment and select various instruments. After the desired changes have been made and the "Reject Changes" and/or "Accept Changes" button has been pressed, the "Control" panel appears. Here changes to the hardware configuration can be made.
3. The LOG CONTROL button allows the user to run the test using a previously stored setting from the log library. This button can also be used to save one or more settings in the log library for future use. These functions are useful when the program is being used with more than one set-up and the user wants to switch from one to another without redefining default values.
4. When the program encounters I/O errors so frequently that the normal execution of the program becomes impossible, a dialogue box appears allowing the user to continue by disallowing the display of I/O messages. Pressing this button once will allow the program to revert to the original status of dialog display.
5. By selecting the GET NEW DATA button, an instantaneous reading of the instruments is taken. This reading is supplemental to the readings taken at due scan intervals, adding one extra set of readings to the data. This button is useful when the user wants to know the current status of the run.
6. The CHANGE DIRECTORY button allows the user to specify the directory where various data, error, and log files are to be placed or searched. This allows the user to organize data in different directories for more efficient retrieval.
7. The DATA UPDATE button allows raw data to be exported to a global VI. Pressing this button is necessary if "NIPER Lab WARDEN" is being run and needs to receive data. If data export is not required, it is recommended that this button not be pressed to avoid efficiency loss.
8. The GRAPH UPDATE button allows graphic data to be exported to a global VI. Pressing this button is necessary if "NIPER GRAPHIC FACILITY" is being run so that it can receive data. If data export is not required, it is recommended that this button should not be pressed to increase efficiency.
9. The PRESS TO STOP button stops the operation being run. The control instruments are reset to their defined termination values or status in a systematic sequence. All of the collected data is stored, and the total number of ex-runs is incremented by one. Stop does not quit NIPER Lab WARDEN or LabVIEW. To exit these programs, the user must manually close the windows.
10. This emergency light flashes when the data readings do not lie within the specified acceptable range of parameters (initially set by the user). If action is not taken when the emergency light is flashing, the automation program will automatically shut down the entire system in an orderly fashion.
11. The warning light is less serious than the emergency light. This light flashes to warn the user that data readings are in the warning range (initially set by the user). A set of actions is taken by the computer to rectify the situation. These corrections involve resetting some control-instruments.
12. This box contains a list of instruments being automatically reset by the computer due to a warning condition, if any.
13. The frame numbers are the order in which the instruments (in box #12) will be reset.
14. This number identifies the instrument (in box #12) to be reset. The number corresponds to the frame number of the "NIPER Control" front panel box on which the item is described.
15. This number determines by what percentage the current values will be reset, or implies the instrument will be turned off, and 999/ implies it will be turned on.
16. This box contains information about the instruments experiencing warning or emergency conditions, if any.
17. The Summary for Diagnostics box contains information about each indicator-instrument. It is useful in warning and emergency situations when a quick diagnostic of the situation is desired to decide whether to override the corrective actions soon to be taken by the program.
18. This display Boolean switch shows whether the instrument displayed in box #14 will be reset to the value shown in box #15 as a result of lower limit warning conditions or higher limit warning conditions.

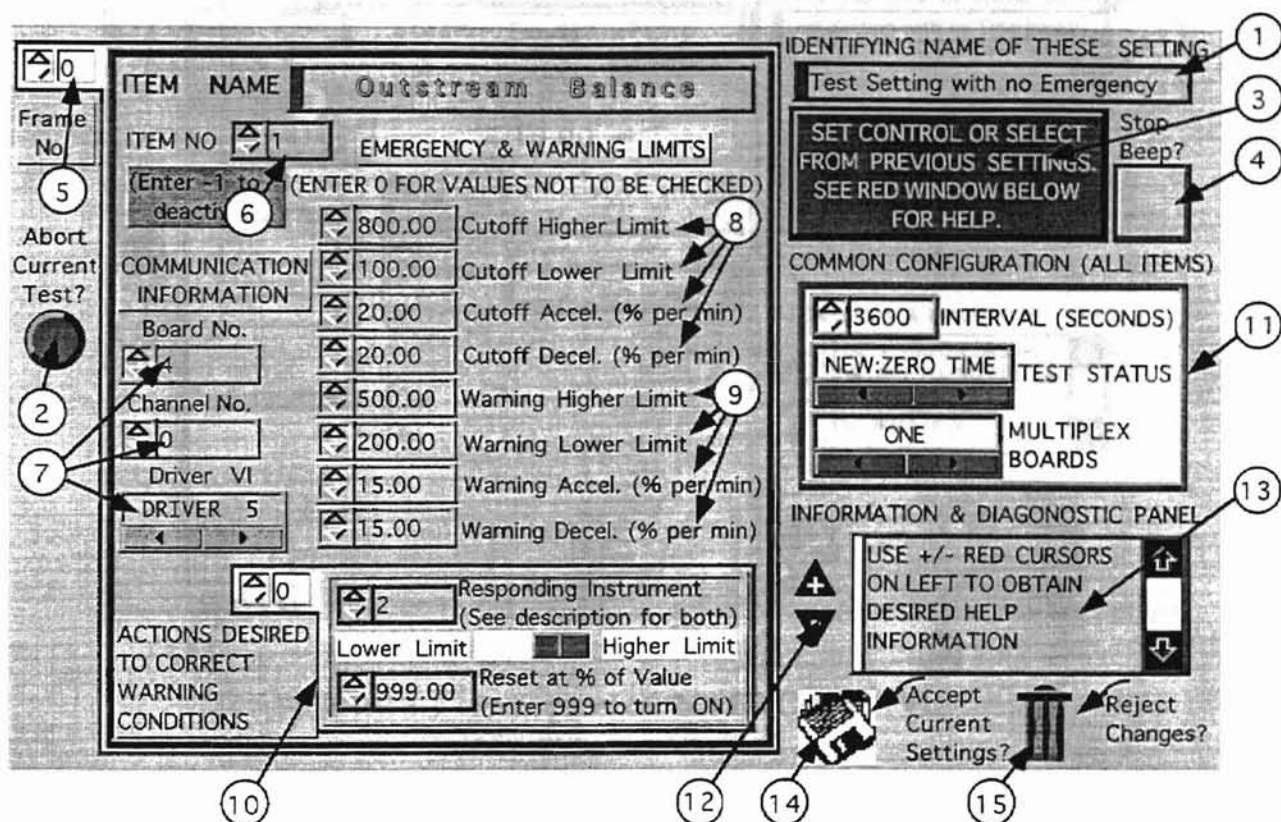


FIGURE 3.2 - Front panel of "NIPER Indicator" VI.

TABLE 3.2
LEGEND FOR FIGURE 3.2

1. This box contains the name of the experimental set-up. This name is arbitrary and does not alter the functionality of the program of the displayed instrument in the list.
2. This button stops all active VI's. It can be used to halt the experiment at this stage.
3. This button serves as an audio-visual reminder to the user that an input is needed here to proceed.
4. This button stops the audio-visual reminder explained in number 3 above.
5. The frame number corresponds to the sequence number of the displayed instrument in the list.
6. This is the item number that the user assigns to his instrument whose name appears in the ITEM NAME box. This is an arbitrary number which does not alter the functionality of the program. However, -1 or any other negative integer will deactivate it from the list.
7. This display contains information about how to communicate with the instrument, i.e., board number, channel number, and driver VI.
8. The emergency cut-off parameters are set in this box.
9. The warning parameters are set in this box.
10. Here the user can set the value of each instrument to be reset by a certain percentage in a warning condition. The resetting is done in the order of the frame number.
11. This display allows the user to set various controls for the experiment. The TEST STATUS sets how the initial data is to be read (data can be read at time zero, a continued time, or appended to a previous run). The number of multiplex boards being used can be set here. Also, the time interval at which each data reading is to be taken by the instruments is set here.
12. The "+" and "-" buttons are used to select the information to be displayed.
13. This provides helpful information about the use of the indicator panel.
14. This is pressed to accept the current settings.
15. This is pressed to reject any changes that may have been made.

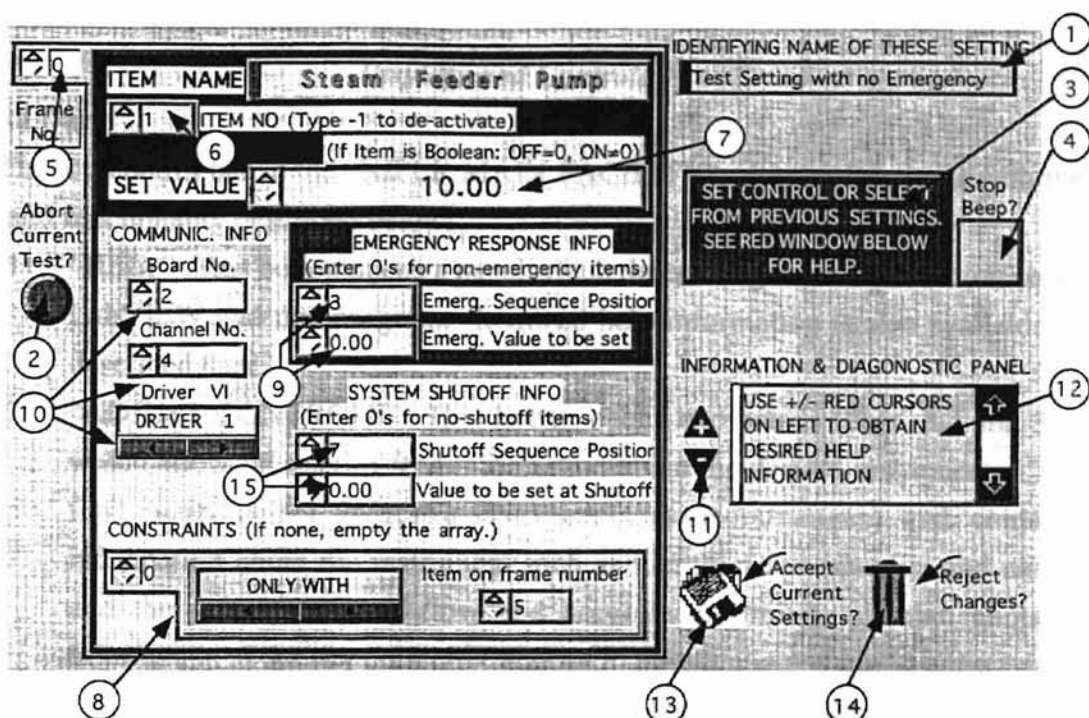


FIGURE 3.3 - Front panel of "NIPER Control" VI.

TABLE 3.3
LEGEND FOR FIGURE 3.3

1. This box contains the name of the experimental set-up. This name is arbitrary and does not alter the functionality of the program.
2. This button stops all active VI's. It can be used to halt the experiment at this stage.
3. This button serves as an audio-visual reminder to the user that an input is needed here to proceed.
4. This button stops the audio-visual reminder explained in number 3 above.
5. The frame number corresponds to the sequence number of the displayed instrument in the list.
6. This is the item number that the user assigns to his instrument whose name appears in the adjacent box. This is an arbitrary number which does not alter the functionality of the program.
7. The user can set the default value at which the instrument will operate unless otherwise changed by the program or the operator. For ON/OFF, use 999 to activate items, even though any non-zero number will activate the instrument while zero will deactivate.
8. This box allows the user to set constraints on the instruments. Under a constraint, an instrument's activation is made conditional upon the active/inactive state of other instruments. Constraining is done with two controls inside the box. The item on frame number box corresponds to the frame number (box #5) on which the selected instrument is located.
9. These two boxes enable the user to establish an emergency response where the values of the selected instruments are reset during the emergency. The Emerg. Sequence position indicates the position of the instrument (if included) in the sequence of emergency reset responses. The Emerg. value to be set is the value at which the instrument will reset also if it is included in the emergency response sequence.
10. This display contains information about how to communicate with the instrument, i.e., board number, channel number, and driver VI.
11. The "+" and "-" buttons are used to select the information to be displayed.
12. This provides helpful information about the use of the indicator panel.
13. This is pressed to accept the current settings.
14. This is pressed to reject any changes that may have been made.
15. These two boxes enable the user to establish a shut-off response where the values of the selected instruments are reset when stop button is pressed by the user in NIPER MAIN FACILITY. The shut-off Sequence Position indicates the position of the instrument (if it is included) in the sequence of shut-off reset responses. The value to be set at shut-off is the value at which the instrument will set to if it is included in the shut-off response sequence.

Control Bar

The top control bar in the "NIPER MAIN FACILITY" contains 9 buttons with the labels and functionality described below:

- (1) "HELP" This button opens a panel which describes general information about LabVIEW and this program. Itemized help information is also available as described later.
- (2) "ADJUST CONTROLS" This button opens two panels (Figs. 3.2 and 3.3) which allow the user to feed information about the desired instruments. The first panel is for indicator-instruments (sensors/gauges) and contains information about the instrument such as identification (number and name) and the set values (high/low and accel/decel limits). It also contains information about any corrective actions to be taken in a warning situation. A corrective action is defined as adjusting the current value of a control-instrument by the specified percentage, or changing its current status (ON/OFF). It also contains general information pertaining to all instruments, i.e., test status, number of multiplex board in the computer, and scan interval.

The second panel is for control-instruments. It contains information about the instrument identification (number and name), the communication port (board number is the slot number in which the board is physically plugged into the computer), and it also contains information about the instrument's set values at start-up, at shut-down, in an emergency, and the sequence for emergency and other response actions. It can also be used to set constraints, e.g., to disable turning an instrument ON when another instrument is OFF.

- (3) "LOG CONTROLS" This button allows selection among a set of instrument settings previously stored by the user. This is useful when the user wants to use the program for multiple sets of experiments. In this situation, each setting is saved with a unique name for quick identification. Any of these saved settings can be quickly retrieved later. This button can also be used to store a new set of instrument settings. The function of this button is to open the front panels of the "NIPER Control" and "NIPER Indicator" VIs without going through the finder to open them. Alternately, these files can be opened directly from the finder. Once these files are opened, the settings can be saved and/or retrieved.

- (4) "ENABLE DIALOGUE" This button allows the file I/O error messages to be displayed in dialogue boxes. When I/O messages are displayed, the user is given an option to discontinue displaying such messages to avoid disruption of the test in progress. This button is used to remove this previously set "non-display" flag. This error-recovery mechanism is built-in to allow the user to continue the test even if there are errors in the system.
- (5) "GET NEW DATA" Data are scanned once every time this button is pressed. It is used to examine the current status of the test. Pressing this button adds an additional set of data besides those scanned at preset intervals.
- (6) "CHANGE DIRECTORY" This button opens a panel containing information about the location of files and the directories. The most likely cause of I/O error messages is an incorrect path description of the directory containing the needed file. This button is used to change the directory paths and change the file name selection.
- (7) "DATA UPDATE" Pressing this button allows data to be sent to the "NIPER DISPLAY FACILITY" when it is running. If "NIPER DISPLAY FACILITY" facility is not running, pressing this button will have no effect except slightly reducing computer efficiency. Press this button only when "NIPER DISPLAY FACILITY" is running.
- (8) "GRAPH UPDATE" Pressing this button allows data to be sent to the "NIPER GRAPHIC FACILITY" when it is running. If "NIPER GRAPHIC FACILITY" is not running, pressing this button will have no effect except slightly reducing computer efficiency. Press this button only when "NIPER GRAPHIC FACILITY" is running.
- (9) "PRESS TO STOP" This button allows the main facility to stop execution after carrying out a normal shut-down procedure (i.e., reset selected control-instruments in prescribed sequence), completing the current cycle and closing all files. Using this button to stop execution is safer than using "Operate" menu, pressing '⌘+.', or closing the file—all of which terminate the program immediately without resetting instruments. It is, therefore, recommended that the execution be stopped using this button to avoid unknown side-effects. When this button is pressed, user is given options (through dialogue boxes) to exit with or without system shut-down. Like emergency shut-down, the normal shut-down procedure (sequence) is defined (or selected) by user through "NIPER CONTROL" front panel during the run.

Other Controls and Displays

In the "NIPER MAIN FACILITY," the front panel contains additional controls and display capabilities including:

"EMERGENCY?" This indicator lights up when one or more scanned data fall outside the allowable range. When this happens, user can quickly browse through "Summary for diagnostic" indicator box to locate the channel(s) having emergency situation(s) and examine the pertinent data to decide whether to override emergency shut-down.

"WARNING?" This indicator lights up when one or more scanned data is in a warning range. When this happens, the program may take some corrective actions by resetting selected control-instruments. A list of these corrective actions is displayed in "Channels requiring pre-emergency adjustments" box.

"Channels requiring pre-emergency adjustments." This indicator box contains a list of the instruments that are being reset to the indicated percentage of their current value. The box remains empty except in a warning situation. Zero percent implies the instrument will be turned-off, and 99.9% implies it will be turned ON. Other values indicate the percent change in the setting.

"Summary for diagnostic." This box contains comprehensive information about each indicator-instrument. It is useful in warning and emergency situations—when a quick diagnostic of the situation is desired—to decide whether to override the corrective actions soon to be taken by the program.

"Channels with Warning and/or Emergency Conditions." This table provides a summary of the status of the test. The name of items with warning or emergency conditions are displayed along with their numbers, error messages, and values in appropriate columns. If no such conditions exist, the table is blank. The scroll windows can be used to browse through the table.

Installing Drivers to the Automation Program

1. By using the "Find" command under the menu option "File" (or using $\text{⌘}+F$) in the "Finder" mode, locate the "Instrument Drivers" folder and place the drivers in it. Remember the name of drivers and their current locations. This step is optional because a driver can be loaded irrespective of where it is located, but it helps to keep all drivers in one folder for future reference.
2. Locate and open the file "Indicator Driver Selector" or the file "Control Driver Selector," depending upon the type of driver, i.e., whether it is an indicator

driver or a control driver. If not sure, see "To Classify Instruments On the Basis of Their Functionality."

3. Under the menu option "Windows," select the "Show Diagram" command.
4. The box, with the number and arrows at the top of the border, is called the case box. For further information on the case box, refer to the instructions provided on the VI, or LabVIEW 2 User's Manual. Using the arrows, go to the last case; i.e., the case showing the highest number.
5. Point the mouse anywhere along the border of the case box. Hold the command key along with the mouse button (command-click). A "Pop Up Menu" will appear.
6. Select the command "Add Case After." An empty case window is now created to hold the new driver that is to be installed. Alternately, select the command "Duplicate Case" instead of "Add Case After." It will create a new case with a copy of the contents of the previous case. In this new case, command-click on the driver icon to select "Replace.." from the "Pop Up Menu" and then select "VI" from the second window that pops up. Since connections are already made, skip Step 9 below.
7. Command click on the empty window. From the "Pop Up Menu" select "VI."
8. Select the driver (file) to be installed. The name and location of the driver were noted in step 1.
9. Under the menu option "Tools," select the wiring tool (the spool of wire located in the middle of the second column). The National Instruments "Getting Started Manual" (1991) explains how the wiring tool works. Wire the top left of the driver icon to the Indicator Panel frame (to the black spot on the bottom of the frame); wire the top right of the driver icon to the Drivers Outputs (to the black spot on the right side of the frame). See previous frames to use as examples.
10. Note the case number of the window in which the driver was just installed; the case number now corresponds to the driver number. Suppose the case number of the window containing the PJ-15 driver is 5, hence select "driver 5" as the driver to communicate with a PJ-15 balance.
11. Save this set-up.
12. Now the driver is ready to allow communication between the computer and the instrument.

To Classify Instruments On The Basis Of Their Functionality

Instruments can be classified as follows:

Indicator instruments. Indicator instruments send signals to the computer. They do not need or respond to commands from the computer. Examples of indicator instruments are thermocouples, pressure sensors, etc.

Control instruments. Control instruments receive signals from the computer. They respond to the commands from the computer and change their status accordingly, e.g. turn themselves ON or OFF, change their pumping rate, etc.

Dual-Function instruments. Dual-function instruments both send signals to the computer and receive signals from the computer. They send signals to the computer when queried, as well as respond to the commands from the computer and change their status accordingly. Thus, they act both like an indicator instrument and a control instrument. One example of dual-function instruments is a Mettler's PJ-15 balance which sends weight reading (an indicator function) when asked by the computer, and resets its "TARE" or turns ON/OFF when commanded by the computer. Another example is a Scannivalve (scanning valve instrument), which expects the computer to tell which channel's reading is desired, positions itself to the commanded channel (a control function), and then sends the reading (an indicator function) of that channel to the computer. The dual-function instruments require two separate drivers, one compatible with indicator driver's format and the other with control driver's format. Although dual-function instruments can provide both functions, it is not required that both functions be used. Thus, either function can be used alone. In the Mettler PJ-15 example above, it is OK to use the dual-function instrument for reading the weight only and not worry about "TARE," or vice versa.

To Analyze A System Configuration Problem

The worksheet in Table 3.4 can be used to analyze a configuration problem. See problem sets in chapter 9 for example use of this worksheet.

To Determine Correct Parameters To Be Entered Into The NIPER INDICATOR VI

The worksheet in Table 3.4 can be used to organize data to be entered into the front panel of the NIPER INDICATOR VI when it opens as the NIPER MAIN FACILITY is run. See problem sets in chapter 9 for example use of this worksheet.

TABLE 3.4
Problem Analysis Work Sheet

STEP 1: Enlist All Automation Instruments In Three Categories (See *To Classify Instruments On The Basis Of Their Functionality* for more details):

Indicator instruments:	
Control instruments:	
Dual-Function instruments:	

STEP 2: Enlist all automation instruments to be reset at start-up (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above):

STEP 3: Enlist all automation instruments to be reset in case of emergency (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above):

STEP 4: Enlist all automation instruments to be reset at run shut-off (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above):

STEP 5: Enlist all automation instruments to be reset in case of warning (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above):

WARNING CONDITIONS	CONTROL INSTRUMENTS	RESPONSE

STEP 6: Enlist All Automation Instruments Whose Operation Is Subjected To Meeting Constraints With Object Instrument(s) (i.e. the control instruments which can be operated only when the status of target instrument(s) meets the constraint conditions):

CONSTRAINED INSTRUMENT	OBJECT INSTRUMENT	LOGICAL CONSTRAINT

To Determine Correct Parameters To Be Entered Into The NIPER CONTROL VI

The worksheet in Table 3.5 can be used to organize data to be entered into the front panel of the NIPER CONTROL VI when it opens as the NIPER MAIN FACILITY is run. See problem sets in chapter 9 for example use of this worksheet.

To Configure NIPER MAIN FACILITY for Specific Automation Setups

The following general steps can be used to configure the NIPER MAIN FACILITY for most automation requirements (See example use of these steps in example problems of Chapter 9):

1. Using Table 3.4, analyze the system configuration.
2. Organize data to be entered into the front panel of the NIPER INDICATOR VI. Use Table 3.5 as template to facilitate this task. Reviewing Table 3.4 carefully helps in accomplishing this task because this table is essentially a transformation of the system configuration to a format readily feedable to the NIPER INDICATOR VI.
3. Organize data to be entered into the front panel of the NIPER CONTROL VI. Use Table 3.6 as template to facilitate this task. Again, reviewing Table 3.1 helps in accomplishing this task because this table is essentially a transformation of the system configuration to a format readily feedable to the NIPER CONTROL VI.
4. Locate a suitable driver for each item and make a list of their names and locations. Table 3.7 can be used as example.
5. Load all drivers in the "Indicator Driver Selector" VI or in the "Control Driver Selector" VI depending upon whether the instrument is an indicator or a control (i.e. whether it is listed in the worksheet of step 2 or step 3). The section ***"Installing Drivers to the Automation Program"*** contains details on how to install a driver. Load every driver in its correct case. For example, if a driver has been designated as driver 0 in Table 3.5 (row 8, col. 2) or Table 3.6 (row 9, col. 2), it should be loaded in case 0. If a case already has a driver, the existing driver can be replaced with the new driver. However, if previous drivers are desired to stay loaded, load the driver into the last case, and consequently, re-designate the driver number in Table 3.5 or Table 3.6. For example, as the driver 0 in Table 3.5 was being loaded into case 0, it was decided to leave the existing driver in case 0 for future use. Instead, the driver 0 was loaded into case 7 (case 0 through 6 already had drivers). Thus, the driver designation in Table 3.5 had to be changed from driver 0 to driver 7.

TABLE 3.5
NIPER Indicator Panel Settings

BOX NAME	Entry For Set #0	Entry For Set #1
Frame No.		
ITEM NAME		
Item No.		
Communication Information		
Board No.		
Channel No.		
Driver VI		
EMERGENCY & WARNING LIMITS		
Cutoff Higher Limit		
Cutoff Lower Limit		
Cutoff Accel. (% per min)		
Cutoff Decel. (% per min)		
Warning Higher Limit		
Warning Lower Limit		
Warning Accel. (% per min)		
Warning Decel. (% per min)		
IDENTIFYING NAME OF THESE SETTINGS		
COMMON CONFIGURATION (ALL ITEMS)		
Interval (seconds)		
Test Status		
Multiplex Boards		
ACTIONS DESIRED TO CORRECT WARN. COND.		
FRAME NO = 0		
Responding Instrument		
Lower or Higher Limit Status		
Reset at % of Value		
FRAME NO = 1		
Responding Instrument		
Lower or Higher Limit Status		
Reset at % of Value		

TABLE 3.6
NIPER Control Panel Settings

BOX NAME	Frame 0 Entry	Frame 1 Entry	Frame 2 Entry
Frame No.			
ITEM NAME			
Item No.			
SET VALUE			
COMMUNICATION INFORMATION			
Board No.			
Channel No.			
Driver VI			
EMERGENCY RESPONSE INFO			
Emerg. Sequence Position			
Emerg. Value to be set			
SYSTEM SHUTOFF INFO			
Shutoff Sequence Position			
Value to be set at Shutoff			
IDENTIFYING NAME OF THESE SETTINGS			
CONSTRAINTS (If none, empty array)			
FRAME NO = 0			
Logical Condition (No name on box)			
Item on frame number			
FRAME NO = 1			
Logical Condition (No name on box)			
Item on frame number			
FRAME NO = 2			
Logical Condition (No name on box)			
Item on frame number			

TABLE 3.7
List of Drivers and Directory Location(s)

Instrument	Driver Name	Directory Location
Balance	DRIVER RS232 Mettler PJ-15	La Cie 200-Q:NIPER Lab WARDEN: Readout & Control Facility: Instrument Drivers:
Timer	DRIVER Timer Example	ditto
Valve	DRIVER Solenoid ON/OFF EXAMPL	ditto
Pump	DRIVER Solenoid ON/OFF EXAMPL	ditto

6. Open and run NIPER Lab WARDEN from the pull-down apple menu or by using "Find File" functions ($\mathbb{A}+F$).
7. Open and run NIPER MAIN FACILITY. Use of NIPER Lab WARDEN to open this facility is more convenient than using "Find File" functions ($\mathbb{A}+F$). As the program is run, the front panel of NIPER INDICATOR VI will open up.
8. Enter information from Table 3.5 (worksheet for indicator instruments) into this panel. Be aware that some boxes have more than one frames. Use arrows to select the right frame number before entering information for that frame. If you want to save it for future use, then select "Make Current Values Default" from the pull-down "Operate" menu. Finally, press Accept Current Settings? to use these settings in the current run. A dialogue box may appear asking if you want to save changes. Respond as desired.
9. The front panel of NIPER CONTROL VI will open up next. Do the same as in step 8 except that use Table 3.6 (worksheet for control instruments) instead of Table 3.5.

Steps 1 through 9 will configure the "NIPER MAIN FACILITY" for a specific set-up. It is now ready to be run.

DESCRIPTION OF NIPER DISPLAY FACILITY

Features

The "NIPER DISPLAY FACILITY" allows the user to examine the current run in progress and has the following attributes:

- (1) Gives a pictorial representation of the physical configuration of the process being controlled,
- (2) Examines data in numeric form,
- (3) Reads error messages,
- (4) Examines the status of the test with regard to warning and emergency conditions, and
- (5) Examines the instruments being adjusted by the computer to overcome warning conditions and their reset values.

In essence, this facility provides a pictorial view of the data and status of the test currently in progress. Its pictorial representation helps to identify potential problems and assess their urgency, and therefore is the facility of choice when a test is in progress and operator is not fully attentive. It is an optional facility that need not run for reading/controlling/monitoring the instruments or graphic display of the data. It only displays data when "NIPER MAIN FACILITY" is running in the background.

Technical Information for Programmers

This program reads data to be displayed from "NIPER GLOBAL Data" VI. This VI is initialized and updated every time new data is scanned by "NIPER MAIN FACILITY" program. The raw data received is sorted according to each item's number designated by the user in the "NIPER Indicator" VI. This sorted array is then indexed to update the displays in sequential order. The advantage of this strategy is that the instrument need not be in order and can be positioned anywhere on the "NIPER Indicator" box. Therefore, any instrument can be added or removed easily without worrying about the sequence.

Because of the sequential handling of the data, the following are prerequisites for complete and proper functioning: (1) The number of active instruments (i.e., the one whose item number defined by the user in the "NIPER Indicator" VI is not a negative number) must be equal to the number of items on the front panel of this facility. If not, there will not be enough data to update all items or vice versa (i.e., there will not be enough items to display all data); (2) No two items on "NIPER Indicator" VI should

have the same item number. The entire ordering after each set of duplicate item numbers will be offset by one.

It must be emphasized that the process diagram shown in Fig. 2.4 (the front panel for "NIPER DISPLAY FACILITY") is not a universal diagram, nor is it necessary to have this facility. Fig. 2.4 was designed for a visual display of high temperature and pressure experiments in NIPER's steamflood laboratory. Users should build their own front panel by copying the indicators from LabVIEW's built-in libraries, or other VIs available to the user. The "NIPER DISPLAY FACILITY" VI can be used as a template to draw/configure such a diagram. After drawing such diagram, and designating each item in the diagram a unique number (in increment of 1, starting from zero), their terminals have to be repositioned within the diagram (i.e., the terminals have to be dragged into the frames of corresponding case numbers). This simple process is explained in NIPER Manuals. National Instruments operating/programming manuals also provide information on how to build pictorial indicators (active display items).

Displays and Controls

In the "NIPER DISPLAY FACILITY" (Fig. 2.4) some of the features that have been incorporated include:

"STOP BUTTON" stops the execution of this VI and lets user activate another VI without unloading the program. Use this button instead of "close file" options when this VI is to be used again in the session to avoid unnecessary unloading/reloading.

"DATA & MESSAGES RECEIVED" box contains a set of indicators for each item displaying the item's name and number, its current value, and error messages. It also displays the user-specified cutoffs (higher-limit and/or lower limit) for each item for comparison with its current value. Information about the desired item can be read by using the arrows on the box or typing in the frame number on which the item is located.

"ITEMS REQUIRING PRE-EMERGENCY ADJUSTMENTS" box contains information about the corrective actions being taken by the "NIPER MAIN FACILITY" during a warning situation. Each frame of this box contains the item number and the percentage change to be made to the item's current value. (Two special numbers are: Zero percent implies the item is being turned OFF and 999% implies it is being turned ON.) Using arrows on the box, or typing in a frame number, all the items requiring adjustments can be seen.

"CHANNELS WITH WARNING AND/OR EMERGENCY CONDITIONS" is a table which provides a summary of the status of the test. The names of items

with warning or emergency conditions are displayed along with item number, error messages, and values in appropriate columns. If no such condition exist, the table is blank. The scroll windows can be used to browse through the table.

"EMERGENCY?" indicator lights up when one or more items are encountering an emergency situation. It is recommended that the VI be closed and "NIPER MAIN FACILITY" be activated when this button lights up to handle emergencies more proficiently.

"CUSTOM DISPLAYS" are pictorial representations of various instruments and their actions created by the user in the process of building the process diagram. The user should keep adequate documentation for all the elements in his diagram. An easy way to keep a description of an item is to package it along with the item itself by opening the description window and typing in necessary information. The custom displays shown in "NIPER DISPLAY FACILITY" (Fig. 2.4) depict various elements of a high-temperature, high-pressure experimental laboratory.

ANALYTICAL DATA

(This page intentionally left blank)

DESCRIPTION OF NIPER GRAPHIC FACILITY

The "NIPER GRAPHIC FACILITY" panel is shown in Fig. 5.1. The circled references for the icons are described in Table 5.1. This chapter describes the features of the "NIPER GRAPHIC FACILITY."

Features

This facility allows the user to:

- (1) View the data from the current run or the previous runs,
- (2) View the data in both numeric and graphical form,
- (3) View only selected data from a run,
- (4) Browse through previous data sequentially,
- (5) View the smoothed data after curve fitting,
- (6) Edit graphs in real-time; i.e. add, delete, copy points or segments, and
- (7) Export data for viewing and processing in external applications such as Microsoft Excel.

It is an optional facility not required for interacting with the instruments. The user does not need to run this facility for automation needs. However, it can be used to browse, curve-fit, reduce, copy, and print previously logged data—and current data, if a run is in progress. The selected data can be easily edited by adding, deleting, or copying points or plot segments in real-time using mouse. The edited graph can be printed with LabVIEW print functions.

Technical Information for Programmers

This program reads data to be displayed from either a global VI or a log-file (in "DATA" format) saved by "NIPER MAIN FACILITY." The format of log-file does not allow it to be opened by most other applications besides LabVIEW.

The raw data received is processed before it is displayed. First, the data is reduced to the set of channels (items) that user has selected for graphic display in "NIPER GRAPH CONFIGURATION" VI. This selected data is then curve-fitted according to the user-selected curve-fitting technique.

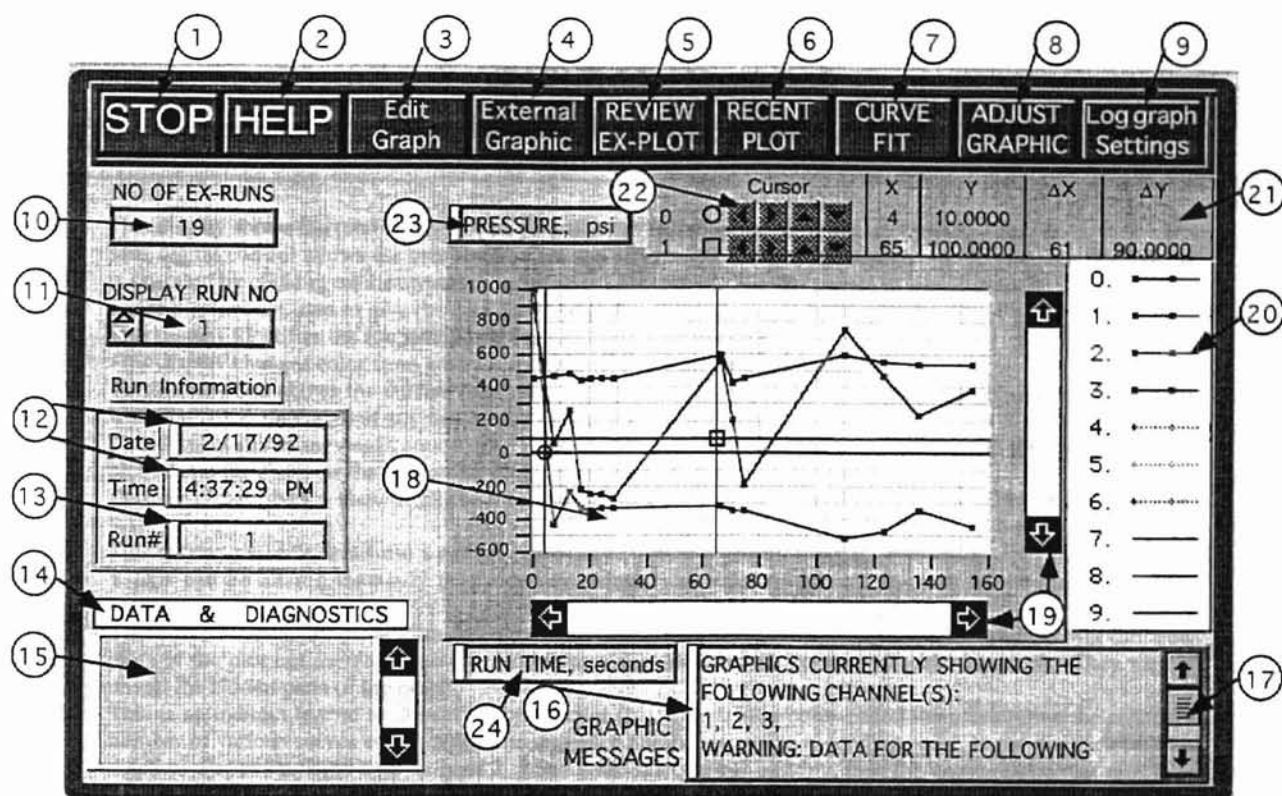


FIGURE 5.1 - The front panel of "NIPER GRAPHIC FACILITY" VI.

TABLE 5.1
LEGEND FOR FIGURE 5.1

1. This button stops the run.
2. This button opens the "NIPER HELP" panel which provides information about LabVIEW and object-oriented programming. Like all of the other windows that are opened from the main panel, the "NIPER HELP" panel automatically disappears after a certain time duration, and the main panel then reappears.
3. This button magnifies the graph shown in Fig. 5.1 to a full size screen-editable graph. This feature provides a better view of the data being graphed, and allows real-time editing of the plot: i.e., points or segments can be added, deleted, and copied from this or other plots by just clicking and pointing the mouse. The edited plot can be saved and printed.
4. This button relays information to graphics programs outside of the main application (or LabVIEW). The particular program may be reselected if desired. The exported data can be automatically plotted and rotated.
5. This button allows the user to view plots of previous runs one at a time. The desired run number is chosen using the selector (11 in Fig. 5.1). The selected number is shown in the display (13 in Fig. 5.1). After the run number is chosen, pressing this button will display the plot of that particular run.
6. This button displays plots of the data scanned from a run currently in progress, provided that the "NIPER MAIN FACILITY" is running and has the "UPDATE GRAPH" button pressed. If there is no run in progress, or the "NIPER MAIN FACILITY" is not running or does not have "UPDATE GRAPH" button pressed, the graphic palette becomes blank and a warning message appears in box 16. The display is updated automatically each time data arrives.
7. This button provides a curve-fit (either linear, polynomial, or exponential) to the data (regression analysis selection made by "ADJUST GRAPHIC" button, described in No. 8). The curve-fit option must already be activated when the new data arrives in order to update and display the curve-fit to the data.
8. This button opens a panel that allows user to adjust various aspects of the graphic display of data. This includes the selection of items to be plotted, data duration, x-axis time base, y-axis unit, type and order of curve-fit, and the name and location of log-file from which to read logged data. Any changes made to these settings may be saved or canceled.
9. This button allows selection of one of the set of graph settings previously stored by the user. This is useful when the user is using the program with multiple test configurations and wants to switch frequently from one setting to another. Each series of settings can be given a name for quick identification. This button can also be used to store a new set of graphic settings.

TABLE 5.1—Continued
LEGEND FOR FIGURE 5.1

-
10. This display shows the total number of ex-runs that are currently saved in the selected log-file.
 11. This digital control allows the user to choose a specific ex-run number for plotting on the graphical display. It is operated by clicking on the arrows to increase or decrease the number shown in the adjacent box.
 12. These displays are used to give the date and time of the selected ex-run, or the current run.
 13. This display identifies the run which is being displayed on the graph.
 14. This display changes color from green to red when there is an error message.
 15. This display box shows the current data values received from an active test in progress. It also warns the user with diagnostic messages of any values that are outside the set limits.
 16. This display shows messages concerning graphics. The display indicates the channels that are currently being plotted and the channels that are set for plotting but inactive for data acquisition.
 17. This scroll bar gives a view of all the information presented in the "GRAPHIC MESSAGES" box (16 in Fig. 5.1).
 18. This display presents data from a current run or an ex-run in a graphic format. The scales along the X-axis and Y-axis can be altered manually (expanded or reduced) in order to present any part of the plot desired for viewing.
 19. These scroll bars along the bottom and right hand sides of the graphic display are another way the user can view parts of the plot not currently shown. However, using these scrolls does not rescale the graph. They merely unveil the hidden parts of the graph.
 20. This is an indirect legend to the curves on the plot (No. 18). These color-coded lines determine the sequence number of various curves on the plot. The sequence number relates in ascending order, to the item selected for display. For example, if item number 2, 5 and 9 are selected for plotting, then the curve on the plot matching the legend line 0 will correspond to the item number 2; the legend line 1 will correspond to the item number 5, and the legend line 2 will correspond to the item number 9.
 21. These displays both indicate and control the locations of the two cursors and their relative displacement from one another. If a cursor is moved by dragging, the current location of that cursor is automatically updated here to show accurate data values from the plot. Or, the desired cursor location (coordinates) can be typed in here, which will move the cursor to the typed-in position. The part of the plot displayed will change to follow the moving cursor. The X-column values indicate the X-coordinate locations of the cursors on the plot. The Y-column values indicate their Y-coordinate locations. The ΔX value indicates the horizontal displacement between the two cursors, and the ΔY value indicates their vertical displacement from one another.
 22. These buttons are used to move the two cursors around the graph by small incremental distances. The part of the plot displayed will change to follow the moving cursor. The circle and square symbols in front of the buttons represent the two cursors, and the arrows on the buttons indicate the direction of movement on the plot. These buttons are particularly useful when jumping from one point on a curve to another directly, which happens when the cursor is locked over a point.
 23. This display provides unit information (temperature, pressure, etc.) for the Y-axis.
 24. This display provides the unit information (seconds, minutes, hours, etc.) for the X-axis.
-

The processed data is displayed on the graphic palette of the panel, on an editable full-size graph of another VI, or exported to another application according to the user's choice. The export of data is carried out by storing the processed data in a temporary file. A macro or script file of an external application is then opened to accomplish these functions. An example macro in Microsoft Excel® is provided with NIPER programs. The macro file should be auto-executable so that it can perform the tasks associated with the external application upon its opening. This macro should call LabVIEW just before finishing the execution.

Displays and Controls

Menu Buttons

Located across the top of the "NIPER GRAPHIC FACILITY" panel, Fig. 5.1, are menu buttons that control various features of the program's operation. Each button is herein discussed.

STOP Button

The STOP button (1) allows the graphic facility to stop execution after completing the current cycle. This button is different than stopping execution by pressing the hexagon in the control palette, selecting "Abort" from the pull-down "Operate" menu, pressing '⌘+.', or closing the file—all of which terminate the program immediately. Use of this button to stop execution avoids any unlikely side-effects.

HELP Button

The HELP button (2) opens a panel titled "NIPER HELP" (Fig. 5.2), which provides information about LabVIEW and object-oriented programming. Like all of the panels that are opened from the main panel, "NIPER HELP" automatically disappears after a certain time duration. The main panel then reappears.

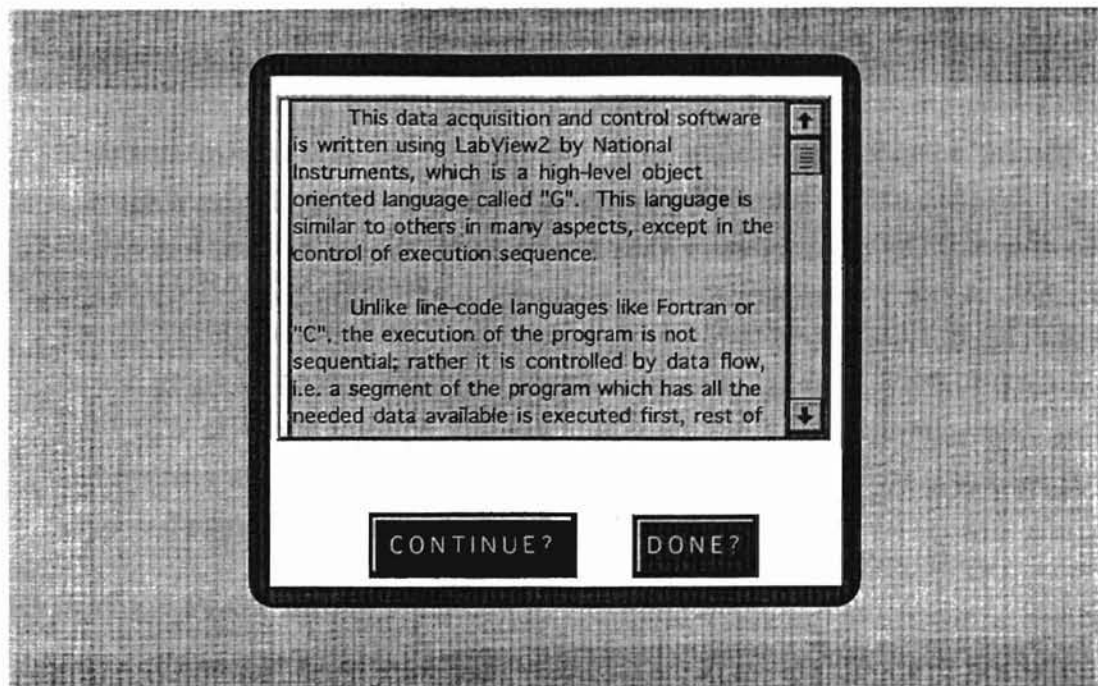


FIGURE. 5.2 - The front panel of "NIPER HELP" VI.

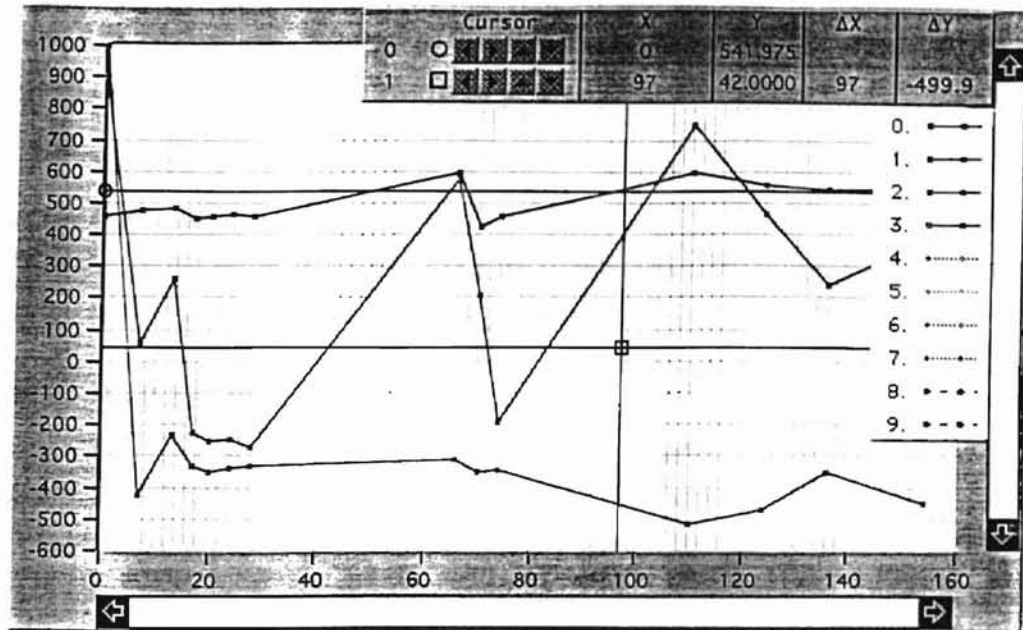


FIGURE 5.3 .- The front panel of "Full Size Graph" VI.

EDIT GRAPH Button

The EDIT GRAPH button (3) magnifies the graph shown in Fig. 5.1 to the graph shown in Fig. 5.3. This feature allows the data currently being displayed on the graphic palette to be displayed, printed, and saved in a full screen size format. The selected data can be easily edited by adding, deleting, or copying points or plot segments in real-time using a mouse. The edited graph can be printed with LabVIEW print functions and saved for future reference.

EXTERNAL GRAPHIC Button

The "EXTERNAL GRAPHIC" button (4) allows the data currently being displayed on the graph to be displayed and saved in an external application program. The LabVIEW saves the selected data in a temporary file in "tab-delimited-text" format, and opens a macro or script file anywhere on the network. The script file is then supposed to graph data and return to LabVIEW.

REVIEW EX-PLOT Button

The "REVIEW EX-PLOT" button (5) allows viewing a selected plot from previous runs. After the run number is chosen, pressing this button will display the plot of that particular run.

RECENT PLOT Button

The "RECENT PLOT" button (6) displays the graph associated with the current run in progress. If no run is in progress at the time, the screen will become (or stay) blank.

CURVE FIT Button

This button (7) smoothes whatever data is selected prior to displaying it according to the type of curve-fit selected by the user.

ADJUST GRAPHIC Button

The "ADJUST GRAPHIC" button (8) opens a panel titled "NIPER GRAPHIC CONFIGURATION" as shown in Fig. 5.4. The circles in Fig. 5.4 are reference numbers that are described in Table 5.2. Through this panel, several data selection and graphic choices can be made such as the item numbers for which data is to be displayed, the name of the log-file to be used during the current graphic session, the type and order of curve-fit desired, the time scale to be used (e.g. hours), and the X-axis units to be displayed (e.g., temperature and pressure). One could also select a previously saved set of graphic settings for the current session and/or for future sessions by making the selection as default. The identifying names (12 in Fig. 5.4) of the previously logged settings are helpful in locating a particular configuration.

LOG GRAPH SETTINGS Button

The "LOG GRAPH SETTINGS" button (9) opens the front panel of "NIPER GRAPH CONFIGURATION" (Fig. 5.4) and stops all VIs. This allows to select a set from graph settings previously saved by the user. This button can also be used to add a new set of graphic settings. This function is useful with multiple test configurations in which user wants to switch frequently from one setting to another. Each setting can be given a name for quick identification.

Run Number Displays

There are various displays on the "NIPER GRAPHIC FACILITY" (Fig. 5.1) that keep the user informed of different experimental conditions, as discussed below:

NO. OF EX-RUNS Display

The display "NO. OF EX-RUNS" (10) shows the total number of ex-runs that are currently saved in the selected log-file.

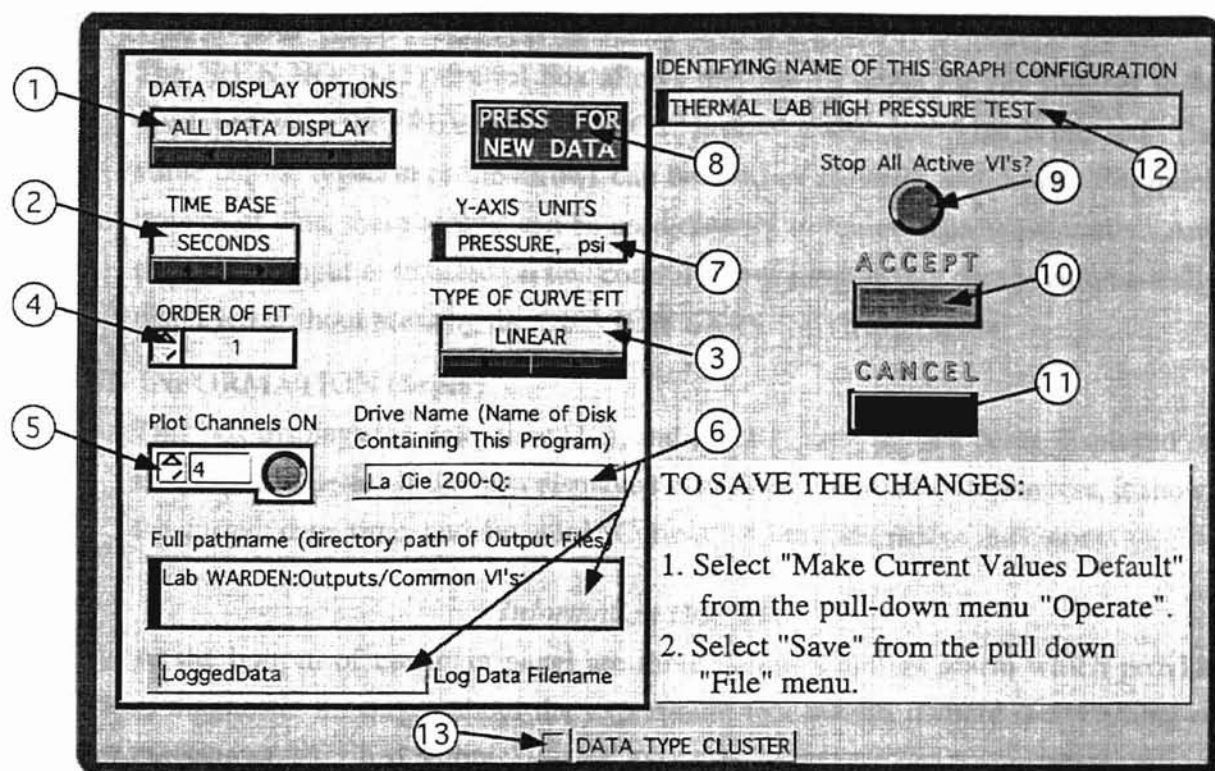


FIGURE. 5.4 - The front panel of "NIPER GRAPHIC CONFIGURATION" VI.

TABLE 5.2

LEGEND FOR FIGURE 5.4

1. This control allows plotting of only selected data. The options include no data display, all data display, new zero time, new append time. These functions are useful when multiple runs are carried out sequentially, or when a run has been interrupted for some reason and resumed later.
2. This control changes the time base of the plot in Fig. 5.1. The available time unit options include the following: HOURS, MINUTES, SECONDS (PORE VOLUMES and PICTORIAL are currently not available).
3. The "TYPE OF CURVE FIT" button is used to set the type of curve-fitting. Three curve-fit options are available: LINEAR, POLYNOMIAL, and EXPONENTIAL.
4. The "ORDER OF FIT" control selects the degree of the curve to be fitted. If either LINEAR or EXPONENTIAL is selected from REGRESSION ANALYSIS, the ORDER OF FIT setting is ignored. However, if the POLYNOMIAL option is selected, then the degree (order) of the polynomial fit needs to be set.
5. The "PLOT CHANNELS ON" button selects the channels (items) to be displayed on the plot located in Fig. 5.1. The circular button indicates the status of the currently selected channel. If the button is "in" (darker shade), the channel is active for graphical display. If the button is "out" (lighter shade), the selected channel is inactive.
6. These boxes describe the log-file's name and location (directory path name) for reading logged data. Several log-files can be stored and used selectively by using this function.
7. This box describes the units to be displayed on the Y-axis of the graph.
8. This button activates the selection made in the "DATA DISPLAY OPTION" box (#1 in Fig. 5.4).
9. All active VI's are stopped by this button. Useful for immediate termination of the run.
10. This button is used for accepting the changes for use in the current session.
11. Pressing this button will default to the original settings, thus nullifying the changes made during the current session.
12. This identifying name is useful in selecting this setting for future use.
13. This is an indicator version of the cluster containing boxes 1 through 8. It has been reduced in size to save the space, thus hiding the boxes.

RUN NO. Display

The "RUN NO." (11) control box allows the user to select the run number to be displayed when the "REVIEW EX-PLOT" button on the control bar is pressed. The value can be typed in or the arrows can be used to move up and down. For quick browse of data, these arrows can be used directly to review data sequentially. Any time a new input is selected on this control, the corresponding data is automatically displayed without pressing the "REVIEW EX-PLOT" button again.

RUN INFORMATION Display

This box displays the date, time (12), and run # (13) of the run being displayed on the graphic palette. If the data displayed is of the run currently in progress, it shows the current date, time, and the word "Current" in their respective indicators.

Information Displays

At the bottom of the main panel are three displays and an alarm which provide various diagnostic messages about the experiment and inform if there is a warning or error message as described below.

DATA & DIAGNOSTICS ALARM

This indicator (14) changes color from green to red when a run currently in progress has an error or warning message. Like the previous box, it only functions when a run is currently in progress. When no run is in progress, it stays green.

DATA & DIAGNOSTICS Display

The "DATA & DIAGNOSTICS" display (15) shows the messages received from the "NIPER MAIN FACILITY" if a run is currently in progress. The messages include data, errors and their probable causes, and warnings. This box keeps the user informed of the current run status. It only displays information when a run is currently in progress, otherwise it stays blank.

GRAPHIC MESSAGES Display

The "GRAPHIC MESSAGES" display (16) gives information concerning graphics. The scrolling indicator box displays the item numbers of the data being displayed in the plot and other graphic error messages. When pertinent, it also suggests possible causes of any problems and remedial actions. These graphic messages pertain to the run selected, which could be the current run or one of the previously logged runs.

Graphical Displays

In the center of the "NIPER GRAPHIC FACILITY" panel is the display (18) that presents data from the current runs or previous runs in a graphic format. The scale is automatically selected to show the entire plot every time new data is added; however, the numbers on the X-axis and Y-axis can also be altered manually in order to present any part of the plot desired for viewing. The scales on the X and Y-axis can also be expanded or reduced automatically using the mouse. The scroll bars along the bottom and right hand sides are another way the user can view parts of the plot not currently shown (19).

Graphic Display Legend

This legend (20) matches the colored lines on the plot with data channels activated for graphic display. However, the numbers in the legend do not correspond directly with the channel number. Instead, the numbers in the legend correspond to the index numbers of the series of selected channels arranged in ascending order and displayed in GRAPHIC MESSAGES display (See Table 5.1, item #20 for details).

Graphic Cursors

The graphic cursor controls (21) are located above the graphic display (18). The circle and square symbols left of the arrow buttons (22) represent the two cursors. The purpose of these cursors is to help obtain accurate data values from the plot.

Cursor Position

Located near the top of the graphic display (18) are columns (21) that pertain to the cursors' position. Both the square and circle cursors share the X, Y, ΔX , and ΔY columns. The X column values indicate the X coordinate locations of the cursors. The Y column values indicate the Y coordinate locations of the cursors. The ΔX value indicates the horizontal displacement between the two cursors. The ΔY value indicates the vertical displacement between the two cursors. These windows are dual-purpose. They can be used to move cursor(s) to desired locations(s) by typing in coordinates, and they can be used to read accurate data when the cursor is placed over a point in the graph using mouse or arrows.

Cursor Movement

Located near the top of the graphic display (18) are buttons (22) that pertain to the cursors' movement. The arrows on the buttons indicate the directions of movement of the cursor in the graphic display. As discussed in the above section "Cursor Position," the cursor can also be moved by typing in the coordinates.

58

Chapter 6

EXAMPLE PROBLEMS

The two sample problems described in this chapter use each of the three main interactive sections in NIPER's automation software: NIPER MAIN FACILITY, NIPER DISPLAY FACILITY, and NIPER GRAPHIC FACILITY. Each problem is briefly described, and the operator is asked to configure the software to accommodate the operation of a different experiment or plant set-up.

Problem 1—Operation of an Electronic Balance

Using NIPER Lab WARDEN software, control an electronic balance (Mettler® PJ-15) using the modem port or the printer port of a low-end Macintosh computer. Using the software, record the weights at pre-determined time intervals and then plot the results as a function of time. Use the schematic of the major components as shown in Fig. 6.1. The driver for PJ-15 Mettler balance "DRIVER RS232 Mettler PJ-15" is located in "Instrument Drivers" folder.

Guidance for the Problem

Problem 1 was designed for inexperienced users. This step-by-step approach was taken by some of the students (who were first time users) testing NIPER's software. The problem is intentionally explained in more detail than necessary to assist first time users. Additional guidance can be obtained by working through the "Getting Started Manual" supplied with the LabVIEW software (1991a) and "Training In-Depth Course on LabVIEW 2" (1991b), both by National Instrument. The RS-232 connection for the Mettler PJ-15 electronic balance is shown in detail in Fig. 6.2. Refer to the Mettler PJ-15 user's manual for the description of the electronics within the balance (Mettler, 1991).

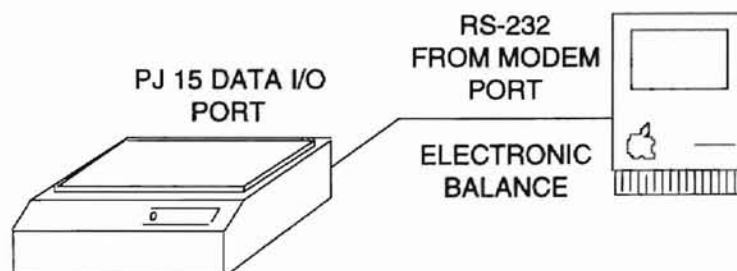


FIGURE 6.1 - Schematic of electronic balance and computer set-up for problem 1.

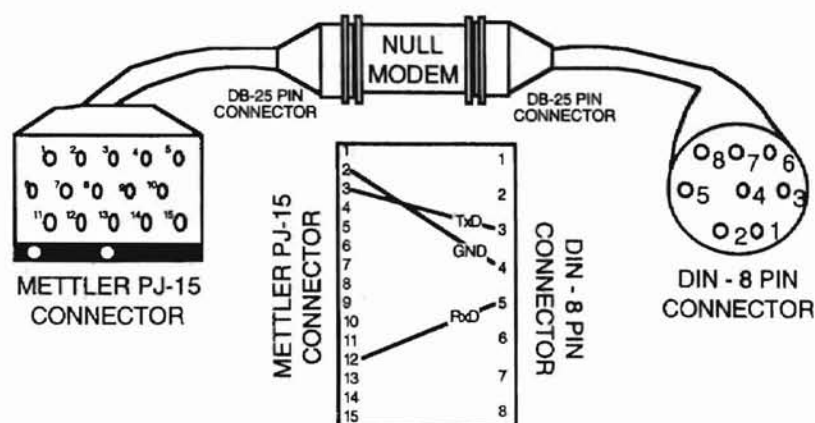


FIGURE 6.2 - Schematic of the Macintosh and Mettler PJ-15 pin configuration.

Sample Solution for Problem 1

1. Install the driver for the Mettler PJ-15, as shown in Fig. 6.3, using the installation procedure under Installing The Driver to the Automation Program section of chapter 3, if it has not been already installed.
2. Using the "Find" command (Fig. 6.4), locate the NIPER Lab WARDEN program (Fig. 6.5), or select it from pull-down apple menu (if you have installed it there). Use the program to open and run the "NIPER MAIN FACILITY." It will open "NIPER I/O Facility" panel (Fig. 6.6).
3. After reviewing and editing the NIPER I/O Facility, click the blue button.
4. The NIPER Indicator Panel (Fig. 6.7) now appears. Refer to chapter 3 for a thorough description of the NIPER Indicator Panel.
 - a. Select the "Frame No." to be 0. Note that the frame number corresponds to the channel number in the NIPER GRAPHIC FACILITY.
 - b. Choose a name for the instrument next to the label "Item Name."
 - c. Next set the "ITEM NO" to 3. The number 3 was chosen at random. If other instruments are being used make sure they do not have conflicting item numbers.
 - d. Below the "ITEM NO" are 8 parameters that pertain to the data values that are read from the balance. Set these ranges as desired.
 - e. Now set the parameters under the "Communication Information." Since the balance is hooked to the modem port, the board number is set to 0 (the default for modem port).
 - f. Here the channel number is irrelevant because a board with multiple channels is not being used. Recall that the modem serial port is a unique singular port.
 - g. Now choose the correct "Driver VI." Remember that the case number of the location of the driver is now selected in the Driver VI box. The case number of the window in which the PJ-15 driver was installed was 5 (see step 10 in the section Installing the Driver to the Automation Program). Therefore, select "DRIVER 5" in the DRIVER VI box.
 - h. Under "Common Configuration," set the interval of time between each data reading to 300 seconds. Again, this value was assumed to be desirable for this experiment.
 - i. Set the "Test Status" to "NEW: ZERO TIME."
 - j. The "MULTIPLEX BOARDS" selector can be ignored since no multiplex boards are being used in this example.
 - k. After adjusting the parameters, press "enter" or click outside any control box (but still on the panel).
 - l. Select "Make Current Values Default," which is located under the menu option "Operate," Fig. 6.8.
 - m. Press the "Accept Current Changes" button when finished. A dialogue box will ask whether to save changes. Respond as you wish.
5. Now the Control Panel (Fig. 6.9) appears; since the balance is not being used as a control instrument in this example (i.e., it does not receive any command, it only sends data), no changes need to be made here except erasing previous entries or disabling them.

- a. Deactivate all the instruments on this control panel to speed up the response of the balance. To deactivate an instrument, enter -1 (a negative integer) in ITEM NO box.
 - b. To turn the instruments off, the "SET VALUE" has to be 0 for each frame number that has an instrument active. In this example, there is no active control instrument involved, thus the value in this box is irrelevant, and this step is optional.
 - c. Select "Make Current Values Default," which is located under the menu option "Operate."
 - d. Press the "Accept Current Changes" button when finished. A dialogue box will ask whether to save changes. Respond as you desire.
6. Now the NIPER MAIN FACILITY (Fig. 6.10) is ready to take data readings from the balance. Review "SUMMARY FOR DIAGNOSTICS" display for useful information such as the warning and emergency values (which were set in the NIPER Indicator Panel), recent and previous data readings from the scale, and other parameters. For further description and use of the NIPER MAIN FACILITY refer to chapter 3 of the manual.
 - a. Wait for 10 seconds for data values to be read, or press the button "Get New Data" to obtain additional data readings in addition to the selected scan intervals.
 - b. After some data readings have been taken, press the Graph Update button. This will allow data to be graphed on the NIPER GRAPHIC FACILITY.
7. Find or use "NIPER Lab WARDEN" to open the NIPER GRAPHIC FACILITY (Fig. 6.11). Refer to chapter 5 for detailed information on the NIPER GRAPHIC FACILITY.
 - a. The NIPER Graph Configuration Panel (Fig. 6.12) should now be displayed. Under "Data Display Options," select the option "All Data Display."
 - b. Select the proper time base and the Y-axis units for the graph to be plotted.
 - c. Since we are not curve fitting the graph, ignore the "Order of Fit" and "Type of Curve Fit."
 - d. Now select which plot channels need to be turned "ON." Recall that the frame number from the NIPER Indicator Panel (Step 5) corresponds to the plot channel on the NIPER GRAPHIC FACILITY. In step 5, the balance was configured on frame No. 0, thus turn channel #0 ON.
 - e. Press the Accept button after making the changes. Respond to the dialogue box as you desire.
8. The front panel now showing is the NIPER GRAPHIC FACILITY. Press the Recent Plot button to display the plot of the data values received from the balance.

Panels and Diagrams for Configuration of Problem 1

The solution to problem 1, configuration of the software for the Mettler PJ-15 balance, is shown in Figs. 6.3 through 6.12 that resulted from sequentially going through the sample solution. The settings shown in these figures are an appropriate set of configurations for the balance. The driver used for this problem to interface with the balance was "DRIVER RS-232 Mettler PJ15" program. This program is included with NIPER Lab WARDEN's library of example instrument drivers. Whereas configuring NIPER Lab WARDEN does not require much familiarity with LabVIEW (even though it is highly recommended), writing a driver program does require proficiency in LabVIEW.

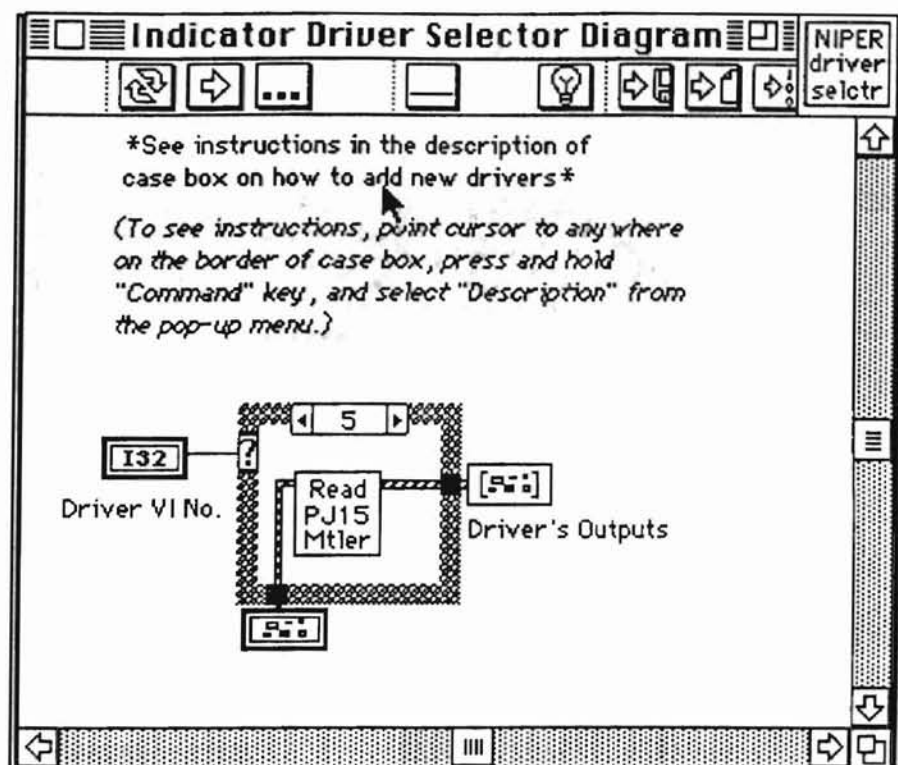


FIGURE 6.3 - Diagram of "Indicator Driver Selector" VI after the driver has been correctly loaded. The icon "Read PJ15 Mtlr" was copied from "Instrument Drivers" folder and represents "DRIVER RS 232 Mettler PJ15" VI.

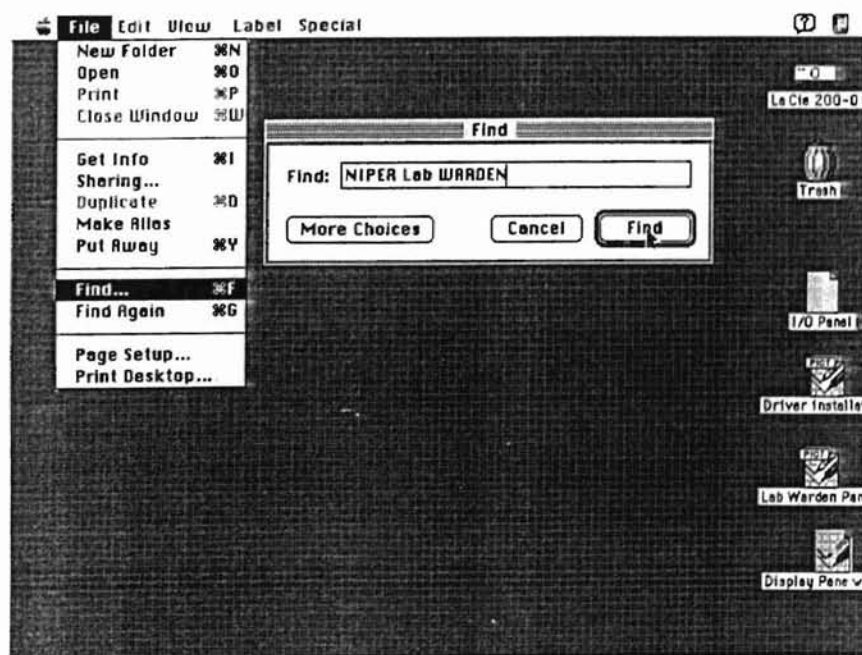


FIGURE 6.4 - Steps involved in using the "Find File" function from the menu.

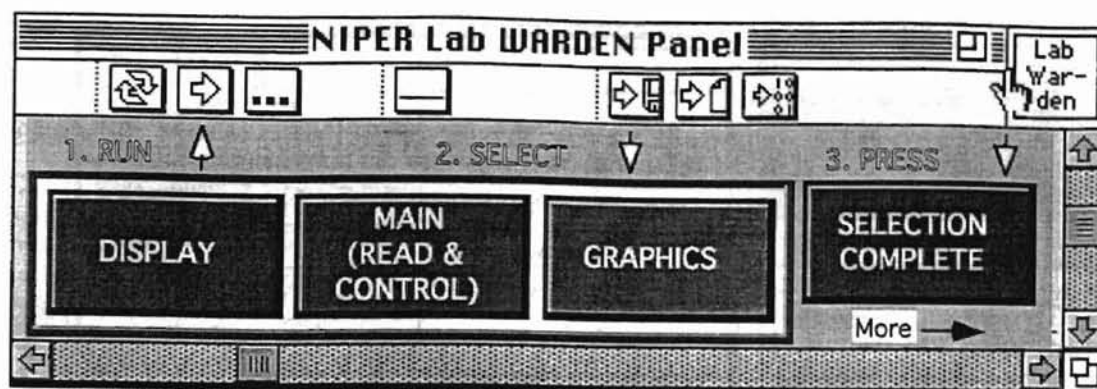


FIGURE 6.5 - Front panel of "NIPER Lab WARDEN" VI.

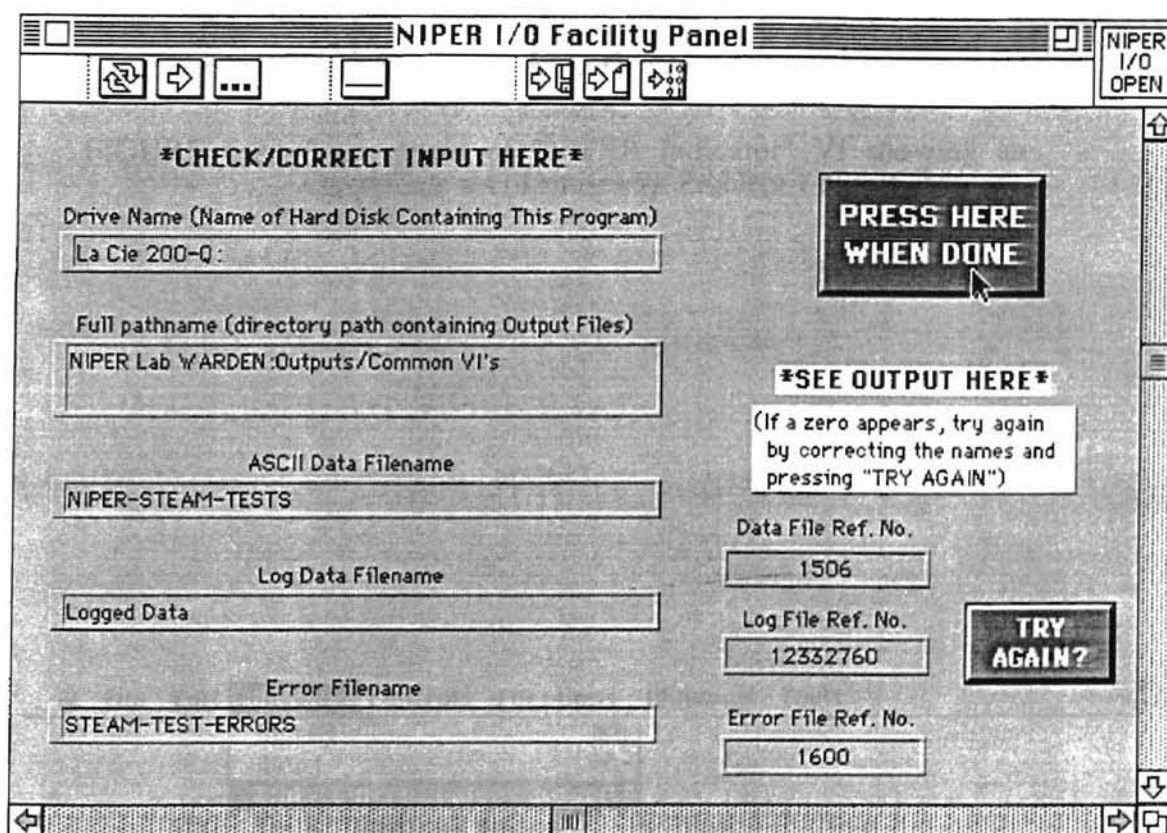


FIGURE 6.6 - Front panel of "NIPER I/O Facility" VI.

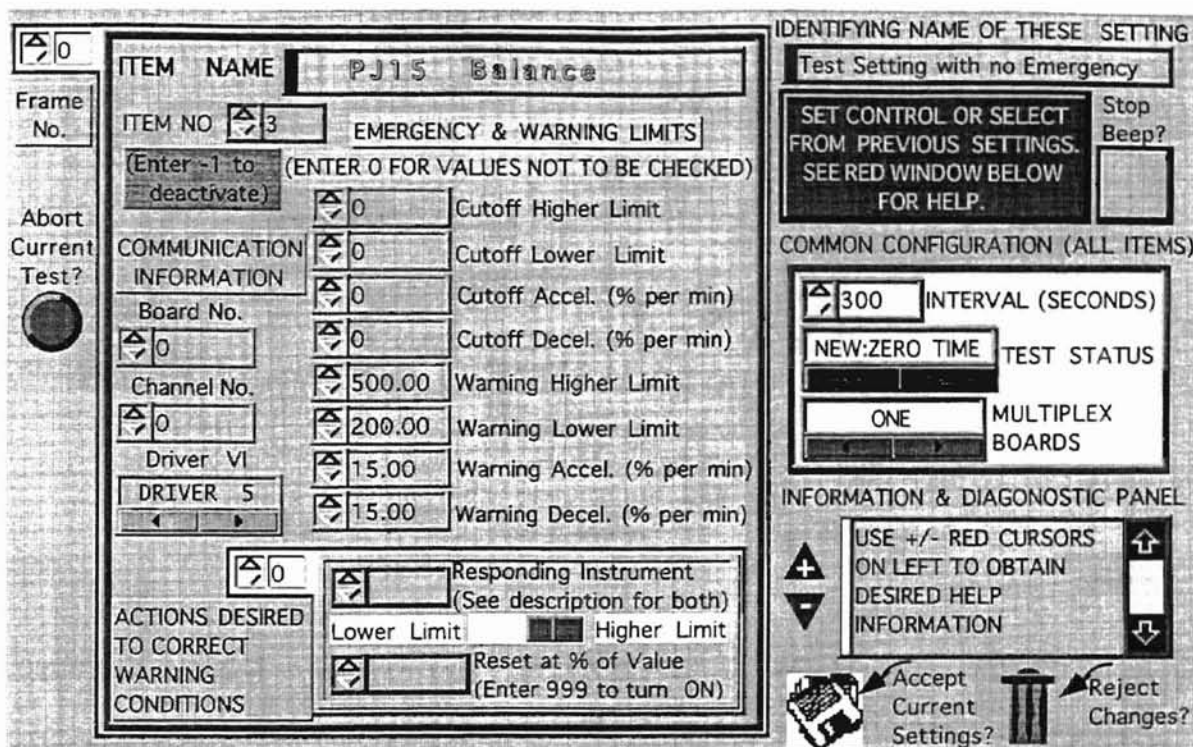


FIGURE 6.7 - Front panel of "NIPER Indicator" VI showing an appropriate set of entries for Problem 1 of this chapter.

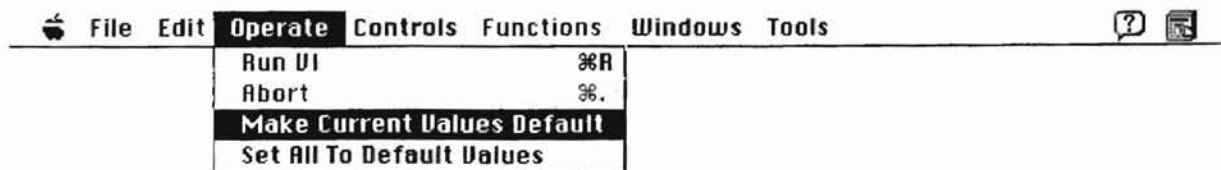


FIGURE 6.8 - How to set default values for the current run from the menu. These default settings will become permanent (till the next selection) if the VI is also saved. Otherwise, the selection will be effective for the current test only.

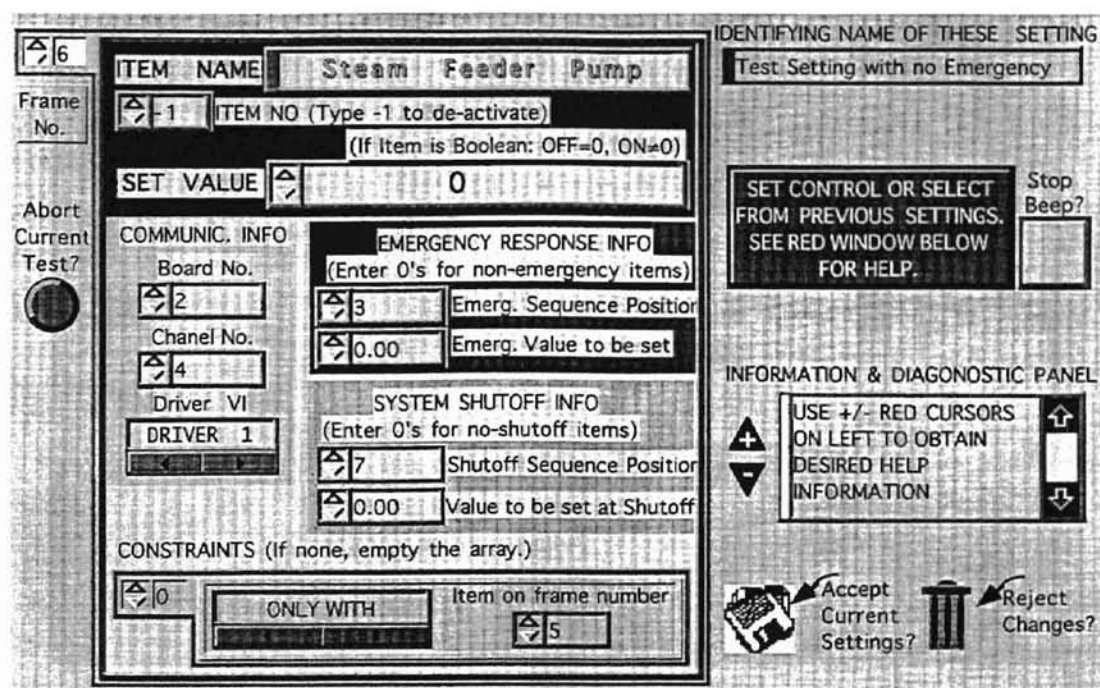


FIGURE 6.9 - Front panel of "NIPER Control" VI showing an appropriate set of entries for Problem 1 of this chapter.

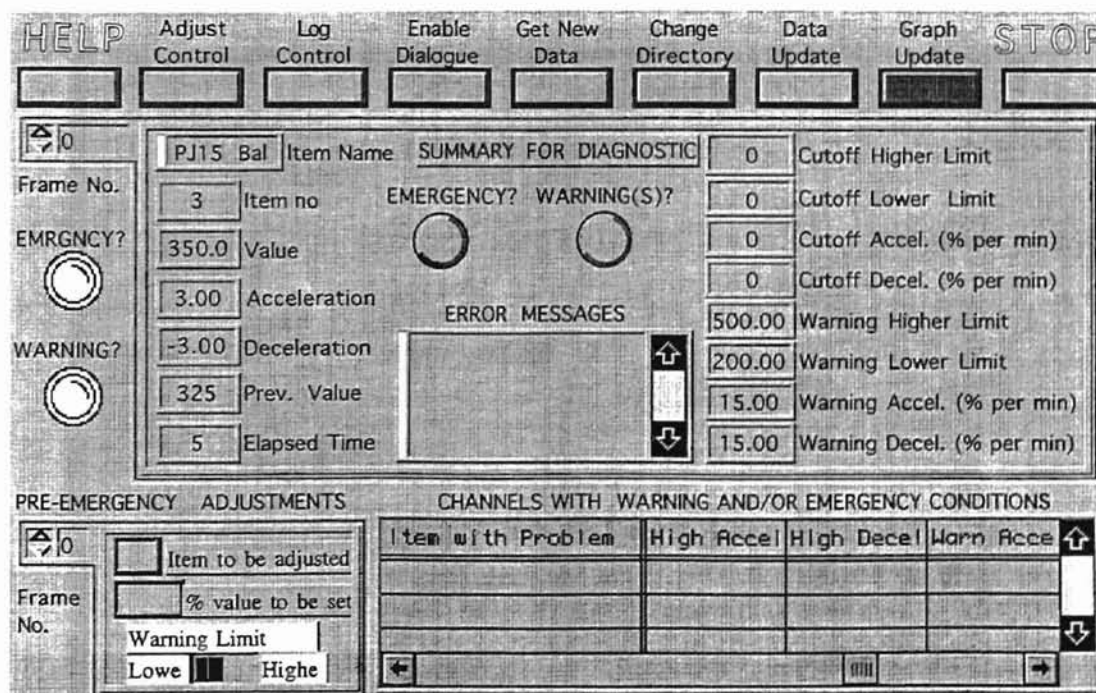


FIGURE 6.10 - Front panel of "NIPER MAIN FACILITY" VI showing the appropriate selection of the menu bar buttons and the type of data received if Problem 1 of this chapter has been properly configured.

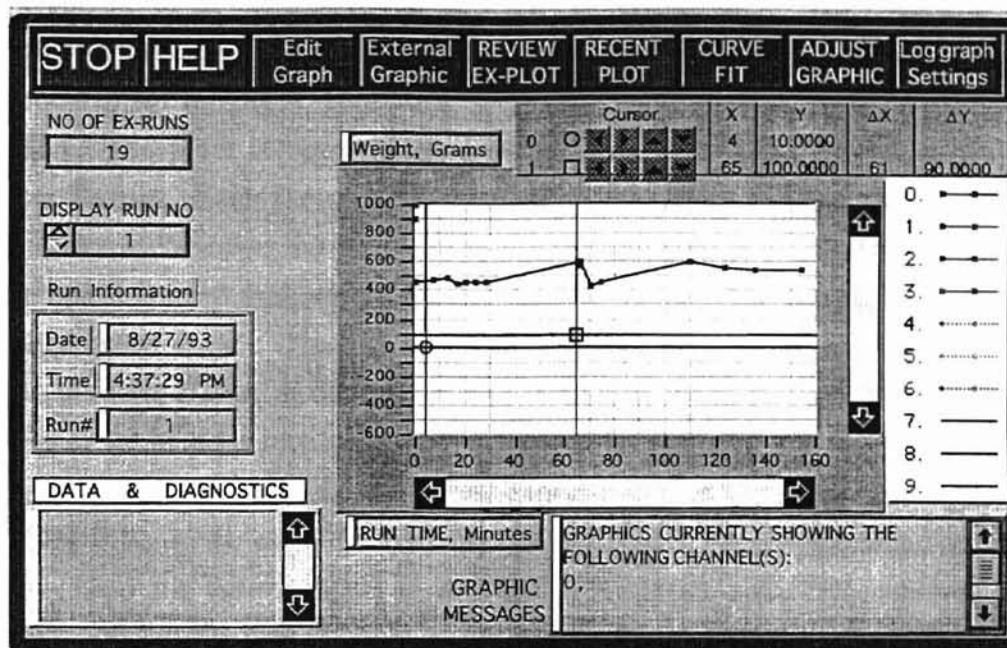


FIGURE 6.11 - Front panel of "NIPER GRAPHIC FACILITY" VI showing the data plot and run information. Note that "RECENT PLOT" button is pressed (not discernible in the figure because of lack of color) to receive data from "NIPER MAIN FACILITY" configured for Problem 1 of this chapter.

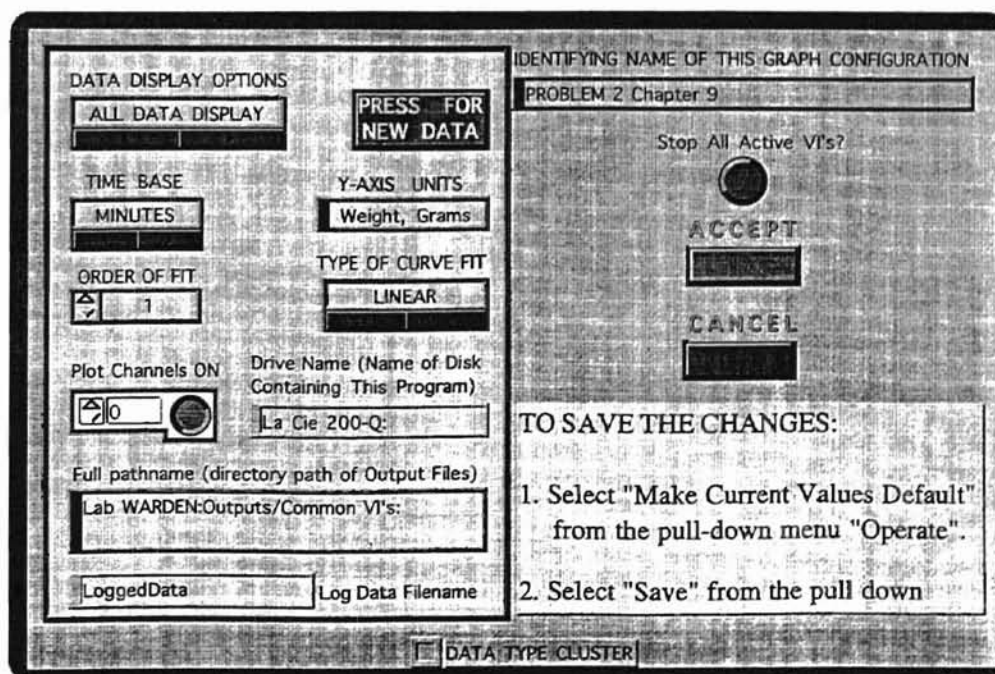


FIGURE 6.12 - Front panel of "NIPER Graph Configuration" VI. This panel opens automatically (or manually when operator desires) to allow the selection of graphic representation of data.

Problem 2—Operation of an Ice Cream Manufacturing Plant

Problem 2 is an example of the versatility of NIPER Lab WARDEN and the user is asked to configure the simplified ice cream plant shown in Fig. 6.13. The plant is a three component plant having a dispensing section, blending/mixing section, and a filling operation. The plant is operated on a batch basis with various batches switched to different flavors. Each flavor is a combination of three premixed liquids in separate flavor tanks each sitting on an electronic balance (Mettler PJ15). The pumps have manual speed control that are set for the specific recipe. The pumps have a computer controlled on/off switch. The blending tank is refrigerated and has two thermocouples to indicate the temperature at the bottom and to caution when the tank is overfilled. The viscosity of the blend is monitored by an ammeter connected to the motor that rotates the blades/paddles in the blending tank. Past experience with this equipment indicates that the mixing is complete when the viscosity of the ice cream causes the ammeter to exceed 9 amps. Only the bottom blade is attached to the shaft, the others are free-rotating. Therefore, even as the tank is depleted the amp load is not much reduced. The ammeter reading should be monitored by the computer and when it exceeds the preset limit of 9 amps, a solenoid valve should be activated and pump 5 should be started.

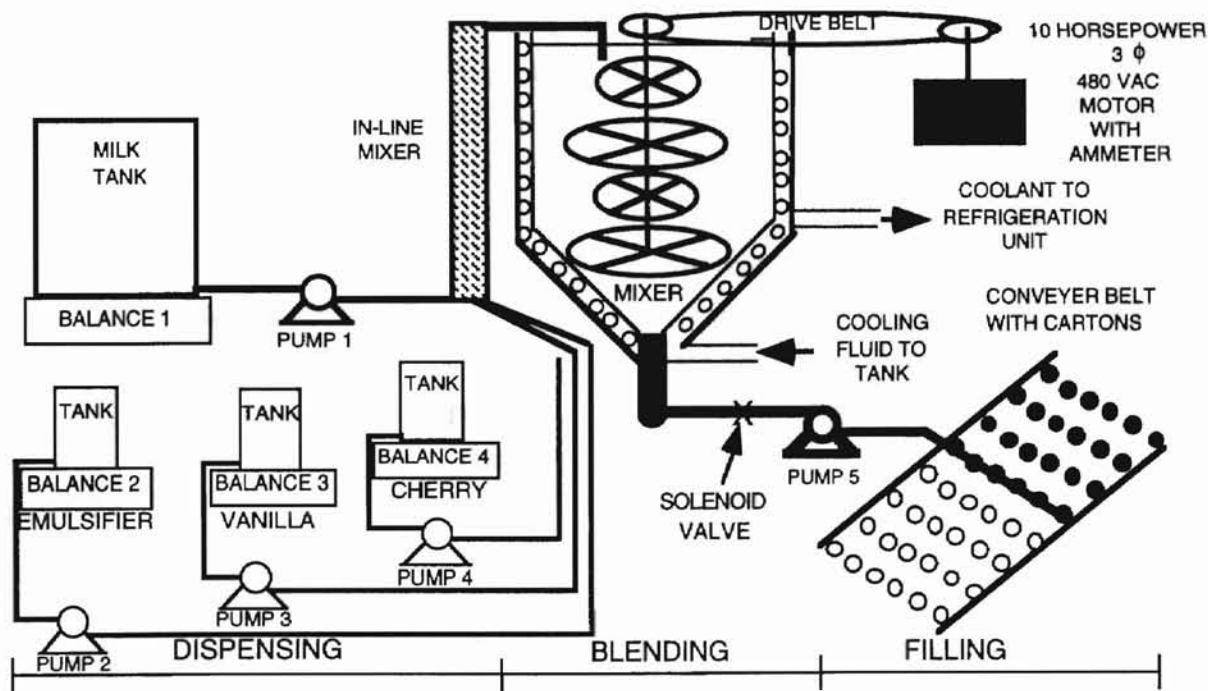


FIGURE 6.13 - Major units in ice cream plant.

During the filling operation, containers ride a conveyor belt under a dispensing head where the cartons are filled with ice cream. The operation is designed to fill six cartons by activating pumps for 10 seconds, then stopping and advancing the conveyor belt. This filling cycle continues until all of the ice cream in the mixer tank has been dispensed. The flavor chosen for this batch is cherry-vanilla and the appropriate pump rates have been manually set. The recipe calls for pump 1 to operate 15 minutes, pump 2 for 3 minutes, pump 3 for 5 minutes and pump 4 for 1 minute.

The user is to configure the plant.

Sample Solution for Problem 2

It is recommended that problem 1 be completed before starting this problem. Follow the step-by-step procedure described in Chapter 3, *To Configure NIPER MAIN FACILITY for Specific Automation Setups*. The resulting tables from steps 1 through 4 of this procedure are Tables 6.1 through 6.4, respectively.

In summary, there are four balances, five timers (computer clock), an ammeter, and two thermocouples that function as indicator instruments (i.e. send signals to the computer), and eight control instruments (the valve, motor, conveyer, and pumps 1 through 5) that receive commands from the computer and change their status. The problem requires, at startup, that any filling operation (Fig. 6.13) in progress be stopped (pump 5, valve, and conveyor be turned OFF), and the dispensing (pumps 1 through 4) and blending (motor) operation be started automatically.

As the dispensing is completed, the pumps 1 through 4 are turned off after 15, 3, 5, and 1 minutes, respectively. These timing correspond to each ingredients quota in the ice cream recipe. The timers 1 through 4 (for pump 1 through 4) perform the function of turning them ON/OFF. As each timer exceeds its warning limit, the corresponding pump is turned OFF as a consequence. The four timers use the same driver because they exactly have the same functionality. (An alternate solution for this task may be to manually set the rate of each pump in the right proportion and use only one timer to shut all four pumps simultaneously. The more involved route was chosen for illustrative purposes).

TABLE 6.1
Problem Analysis Work Sheet

STEP 1: Enlist All Automation Instruments In Three Categories (See *To Classify Instruments On The Basis Of Their Functionality* for more details):

Indicator instruments:	Balance 1, Balance 2, Balance 3, Balance 4, Ampmeter, Timer 1 through 5, Thermocouple 1 and 2
Control instruments:	Valve, Motor, Conveyer, Pump 1 through 5
Dual-Function instruments:	None

STEP 2: List all automation instruments to be reset at start-up (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above): Pump 5 (OFF), Valve (OFF), Conveyer (OFF), Pumps 1 through 4 (ON), Motor (ON)

STEP 3: List all automation instruments to be reset in case of emergency (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above): Pump 5 (OFF), Valve (OFF), Conveyer (OFF), Pumps 1 through 4 (OFF), Motor (OFF)

STEP 4: List all automation instruments to be reset at run shut-off (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above): Same as in STEP 3 above

STEP 5: List all automation instruments to be reset in case of warning (These instruments have to be Control or Dual-Function Instruments and must be included in the listing in STEP 1 above): Pump 1 through 4 (OFF), Pump 5 (ON or OFF), Valve (ON or OFF), Conveyer (ON or OFF)

WARNING CONDITIONS	CONTROL INSTRUMENTS	RESPONSE
Timer 1 \geq 15 min	Pump 1	Turn OFF (Reset to 0)
Timer 2 \geq 3 min	Pump 2	Turn OFF (Reset to 0)
Timer 3 \geq 5 min	Pump 3	Turn OFF (Reset to 0)
Timer 4 \geq 1 min	Pump 4	Turn OFF (Reset to 0)
Ampmeter $>$ 9 amp	Valve, Pump 5, Conveyer	Turn ON (Reset to 999)
Timer 5 \geq 20 min	Pump 5	Turn OFF (Reset to 0)
Wt. Balance 1 \leq 1000 kgms	None	Audio/Video Warning
Wt. Balance 2 \leq 100 kgms	None	Audio/Video Warning
Wt. Balance 3 \leq 100 kgms	None	Audio/Video Warning
Wt. Balance 4 \leq 100 kgms	None	Audio/Video Warning
Thermocouple 1 \leq 35° F	Pump 1 through 4	Turn OFF (Reset to 0)

STEP 6: List all automation instruments whose operation is subjected to meeting constraints with object instrument(s) (i.e., the control instruments which can be operated only when the status of target instrument(s) meets the constraint conditions):

CONSTRAINED INSTRUMENT	OBJECT INSTRUMENT	LOGICAL CONSTRAINT
Pump 5	Valve Conveyer	Pump Can turn ON "ONLY WITH" Valve ON Pump Can turn ON "ONLY WITH" Conveyer ON

TABLE 6.2
Settings for Indicator Instruments

Box Name	Entry For Set #0	Entry For Set #1	Entry For Set #2	Entry For Set #3
Frame No	0	1	2	3

ITEM NAME	Balance 1 (PJ15)	Balance 2 (PJ6)	Balance 3 (PJ15)	Press. Transd.
Item No.	0	1	2	3

COMMUNICATION INFORMATION				
Board No.	0	2	2	2
Channel No.	0	0	1	3
Driver VI	Driver 0	Driver 1	Driver 0	Driver 2

EMERGENCY & WARNING LIMITS				
Cutoff Higher Limit	0	120	0	30
Cutoff Lower Limit	0	0	0	0
Cutoff Accel. (% per min)	0	0	0	0
Cutoff Decel. (% per min)	0	0	0	0
Warning Higher Limit	0	115	450	25
Warning Lower Limit	100	100	0	0
Warning Accel. (% per min)	0	0	0	0
Warning Decel. (% per min)	0	0	0	0

IDENTIFYING NAME OF THESE SETTINGS	Prob 4 of Chapter 9
---	---------------------

COMMON CONFIGURATION (ALL ITEMS)	
Interval (Seconds)	60
Test Status	New:Zero Time
Multiplex Boards	Select Any

ACTIONS DESIRED TO CORRECT WARN. COND.				
---	--	--	--	--

FRAME NO = 0				
Responding Instrument	2 (Light 1)	3 (Light 2)	4 (Light 3)	5 (Light 4)
Lower or Higher Limit Status	Lower Limit	Lower Limit	Higher Limit	Higher Limit
Reset at % of Value	999	999	999	999
FRAME NO = 1				
Responding Instrument	Empty	1 (Pump)	Empty	Empty
Lower or Higher Limit Status	Empty	Lower Limit	Empty	Empty
Reset at % of Value	Empty	999	Empty	Empty
Frame No = 2				
Responding Instrument	Empty	3 (Light 2)	Empty	Empty
Lower or Higher Limit Status	Empty	Higher Limit	Empty	Empty
Reset at % of Value	Empty	999	Empty	Empty

TABLE 6.3
Settings for control instruments

Box Name	Frame 0 Entry	Frame 1 Entry	Frame 2 Entry	Frame 3 Entry	Frame 4 Entry	Frame 5 Entry
Frame No	0	1	2	3	4	5
ITEM NAME	Valve	Pump	Light 1	Light 2	Light 3	Light 4
Item No.	5	6	7	8	9	10
Set Value	999	0	0	0	0	0
COMMUNICATION INFORMATION						
Board No.	3	3	3	3	3	3
Channel No.	0	1	2	3	4	5
Driver VI	Driver 0	Driver 0	Driver 0	Driver 0	Driver 0	Driver 0
EMERGENCY RESPONSE INFO						
Emerg. Sequence Position	1	2	0	0	0	0
Emerg. Value to be set	1 (Drain)	0	0	0	0	0
SYSTEM SHUTOFF INFO						
Shutoff Sequence Position	1	2	3	4	5	6
Value to be set at Shutoff	0	0	0	0	0	0
IDENTIFYING NAME OF THESE SETTINGS		Prob 4 of Chapter 9				
CONSTRAINTS (If none, empty array)						
FRAME NO = 0						
Logical Condition (No name on box)	Empty	Only With	Empty	Empty	Empty	Empty
Item on frame number	Empty	5	Empty	Empty	Empty	Empty

TABLE 6.4
List of Drivers and Directory Location(s)

INSTRUMENT	DRIVER NAME	DIRECTORY LOCATION
Balance 2 Balance 3 Balance 4	Driver RS232 Mettler PJ6	La Cie 200-Q:NIPER Lab WARDEN: Readout & Control Facility: Instrument Drivers:
Balance 1	Driver RS232 Mettler PJ15	ditto
Timer 1 through 5	Driver Timer Example	
Ammeter	Driver Analog Input Example	ditto
Pump 5	Driver Timed ON/OFF EXAMPLE	ditto
Pump 1 through 4, Valve, Motor, Conveyer	Driver Solenoid ON/OFF EXAMPL	ditto

As the ingredients in the blending are whipped by the mixer and cooled down, the viscosity of the ice cream begins to increase causing the motor to draw more power. When the ammeter shows more than 9 amps, a warning is issued that ice cream is ready to be dispensed, and as a result, the dispensing unit is activated in the following sequence: the valve is turned ON (open), pump 5 is turned ON, and the conveyer is turned ON.

Pump 5 requires a different driver than the one for pumps 1 through 4 so that it can dispense in a ON-10-seconds/OFF-2-seconds cycles when it is turned ON. The driver is a simple modification of the driver for other pumps and other ON/OFF instruments. The choice to handle the timed-cycle sequence of pump 5 at the driver level was made because of the small time intervals (10 and 2 seconds). For imprecise operation, it can be handled by the same driver. Timer 5 is used to turn this cyclic operation of pump 5 OFF at appropriate time (see next paragraph).

The conveyer is assumed to have an adjustable feed rate, where it can be manually set to advance one step in 2 seconds and stay there for 10 seconds (like in most stepper-motor operations). It also has a manual synch mechanism, where an operator can synchronize it with pump 5 by pressing the "GO" button exactly when the pump has stopped dispensing.

With previous experience, the operator has determined the most efficient batch time to be 20 minutes, i.e. the time it takes to dispense, blend, and fill the entire ice cream batch. The software has to be configured such that the timer 5 issues a warning every 20 minutes that the batch time has ended, and as a result of this warning, pump 5, valve, and the conveyer are turned OFF sequentially. (Alternately, a warning condition such as ammeter ≤ 5 can be used to turn these three instruments OFF because the load on the motor is drastically reduced when the mixer tank is emptied. This drastic reduction is

due to the fact that most high efficiency mixers have fixed blades only at the tip of the shaft—other blades are freely rotating).

To start a new batch every 20 minutes, operator selects (from the driver panels) that all the timers (timer 1 through 5) be reset to 0 every 20 minutes, which removes warning conditions from all the control instruments, and resets them to their startup values (the NIPER Lab WARDEN program code automatically resets instruments to their original values when a warning situation is corrected), i.e., pump 5, valve, and conveyor are turned OFF; and pump 1 through 4 and the motor is turned ON. The batch process automatically continues.

Other warning, emergency, and constraints for the process are as follows. When the weight of Milk Tank falls below 1,000 kilograms or the weight of other tanks falls below 100 kilograms, audio/video warning is issued; if no action is taken by the operator and the weight falls below 500 or 50 kilograms, respectively, the instruments are shut down in this sequence: pump 5, valve, and conveyor, pump 1 through 4, and motor. If the cooling in the blending tank is not complete (thermocouple 2 > 32° F), the emergency shut down sequence is initiated. If the tank is overfilled (thermocouple 1 ≤ 35° F, indicating that the level has exceeded the thermocouple level), a warning is issued, and if not overridden by the operator, pumps 1 through 4 are turned OFF. The operator may shut the system down at any time. The selected sequence for normal shut down is the same as the emergency shut down sequence. The only safety constraint in the process is that the pump can not be turned ON unless both the valve and the conveyer are ON.

BIBLIOGRAPHY

- Circle Seal Controls, 1988. *Seven SV10-40 and SV400 Series, 2- and 3-Way Leakproof Solenoid Valves*, Circle Seal Controls, 1111 N. Brookhurst, Anaheim, CA, 92803, January.
- Eurotherm Corp., 1987. *Products Catalog 1027-A*, Eurotherm Corp., 11485 Sunset Hills Road, Reston, VA, 22090, January.
- Eurotherm Corp., 1982. *Eurotherm Operation and Maintenance Manual For Digital Temperature Controller Model 918*, Eurotherm Corp., 11485 Sunset Hills Road, Reston, VA, 22090-5286, July.
- Gordos, 1993. *Specifications for PB24 Relay Board*. Newark Electronic, Part No. 21F858 Mounting Board PB-24, Newark Electronics, Tulsa, OK, August.
- Greensprings Computer Inc., 1990. *Operation Manual for Multiport (4) Serial Board, P/N M51001A*, Greensprings Computer Inc., 1204 O'Brian Drive, Menlo Park, CA, 94025, January.

- Mettler, 1990. *Mettler Balance PJ15 User's Manual*, Mettler Instrument Corp., Box 71, Hightstown, NJ, 08520-9944, April
- National Instruments Corp., 1991a. *LabVIEW 2—Getting Started Manual*, Part No. 320246-01, Austin, TX, April.
- National Instruments Corp., 1991b. *Training In-Depth Course on LabVIEW 2*, Version 1.4, Part No. 776393-01, Austin, TX, April.
- National Instruments Corp., 1991c. *LabVIEW 2—User Manual*, Part No. 320244-01, Austin, TX, September.
- National Instruments Corp., 1992. *NB-DIO-24—User Manual*, Part No. 320094-01, Austin, TX, October.
- Paroscientific, Inc., 1990. *Digiquartz Precision Pressure Instruments, Programming and Operation Manual*, Document No. 8107-001, 4500 148th Avenue N.E., Redmond, WA 98052, August.

2

VITA

Syed Mohammad Mahmood

Candidate for the Degree of

Master of Science

Thesis: GENERAL-PURPOSE AUTOMATION PROGRAMMING USING A
GRAPHIC LANGUAGE

Major Field: Computer Science

Biographical:

Personal Data: Born in Karachi, Pakistan, On July 23, 1952, the son of Mr.
and Mrs. Farooq.

Education: Received Bachelor of Science degree with a major in Mechanical
Engineering at Peshawar University, Pakistan, in 1975; Received Master of
Science and Doctor of Philosophy degrees with a major in Petroleum
Engineering at Stanford University in January 1987; Completed the
requirements for the Master of Science degree with a major in Computer
Science at Oklahoma State University in May 1996.

Experience: Worked as Mechanical Engineer (1975-1978) in a Cement
Manufacturing Factory; employed by Stanford University, Department of
Petroleum Engineering as a graduate research and teaching assistant (1978-
1984); Provided Engineering Consultancy Services to Schlumberger (1985);
employed by National Institute for Petroleum and Energy Research as a
Research Engineer (1986-1993); currently employed by BDM-
OKLAHOMA as Senior Petroleum Engineer (Starting January 1, 1994).

Professional Memberships: Society of Petroleum Engineers (SPE), Society of
Core Analysts (SCA), Instrument Society of America (ISA), Marquis
Who's Who in Science and Engineering, Strathmore's Who's Who in
America.