TECHNICAL REPORT STANDARD PAGE

1. Report No. FHWA/LA-92/250	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle DESIGN AND DEVELOPMENT OF DATA ACQUISITION SYSTEM FOR THE LOUISIANA ACCELERATED LOADING DEVICE	5. Report Date SEPTEMBER 1992	
	6. Performing Organization Code	
7. Author(s) Louay N. Mohammad, Ph.D. Srinivas K. Kathavate	8. Performing Organization Report No. 250	
9. Performing Organization Name and Address LA. Department of Transportation and Development 4101 Gourrier Avenue Baton Rouge, LA 70808	10. Work Unit No.	
	11. Contract or Grant No. 92-6SS	
12. Sponsoring Agency Name and Address Louisiana Transportation Research Center P.O.Box 94245, Capitol Station Baton Rouge, LA 70804-9245	13. Type of Report and Period Covered Final Report March 1992 - June 1992	
	14. Sponsoring Agency Code	

15. Supplementary Notes

16. Abstract

The Louisiana Transportation Research Center has established a Pavement Research Facility (PRF). The core of the PRF is a testing machine that is capable of conducting full-scale simulated and accelerated load testing of pavements (better known as Accelerated Loading Device, ALD). The ALD provides a new approach to evaluate pavement materials, construction, and performance; and to verify mathematical models of mechanistic pavement design procedures.

Instrumentation and data acquisition plays an integral part in the PRF to provide a system for monitoring materials and systems responses due to traffic loads and environmental conditions. These responses are essential for pavement materials characterization and performance evaluation.

The data acquisition system developed for the Louisiana ALD includes a 486 based Personal Computer, plug-in distributed Input/Output board, interface circuit boards, signal conditioning modules, and Graphical User Interface (GUI) software development tools. Menu driven user friendly software, Accelerated Loading Device Instrumentation Software (ALDIS), was developed in C language under Labwindows environment. ALDIS offers attractive GUI features to acquire data from most common types of sensors at various sampling rates, data storage and management in ASCII/Binary format and real-time graphical presentation of the data.

17. Key Words Pavement Materials Characterization, Instrumentation, Distributed Input/Output, Sensors, A/D - D/A, Signal Conditioners, Instrumentation Software	18. Distribution Statement Unrestricted. This document is available to the public through the National Technical Information Service, SpringField, Virginia 22161.

19. Security Classif. (of this report) N/A	20. Security Classif. (of this page) N/A	21. No. of Pages 93	22. Price

Form DOT F1700.7 (1-92)

DESIGN AND DEVELOPMENT OF DATA ACQUISITION SYSTEM FOR THE LOUISIANA ACCELERATED LOADING DEVICE

FINAL REPORT

By

LOUAY N. MOHAMMAD, Ph.D.
ASSISTANT PROFESSOR - RESEARCH OF CIVIL ENGINEERING
COORDINATOR, ENGINEERING MATERIALS
CHARACTERIZATION RESEARCH FACILITY

And

SRINIVAS K. KATHAVATE
GRADUATE RESEARCH ASSISTANT

Research Report No. 250
Research Project No. 92-6SS(B)

Conducted By

LOUISIANA TRANSPORTATION RESEARCH CENTER
In cooperation with
U.S. Department of Transportation
Federal Highway Administration

August 1992

ABSTRACT

The Louisiana Transportation Research Center has established a Pavement Research Facility (PRF). The core of the PRF is a testing machine that is capable of conducting full-scale simulated and accelerated load testing of pavements (better known as Accelerated Loading Device, ALD). The ALD provides a new approach to evaluate pavement materials, construction, performance; and to verify mathematical models of mechanistic pavement design procedures.

Instrumentation and data acquisition plays an integral part in the PRF to provide a system for monitoring materials and systems responses due to traffic loads and environmental conditions. These responses are essential for pavement materials characterization and performance evaluation.

The data acquisition system developed for the Louisiana ALD includes a 486 based Personal Computer, plug-in distributed Input/Output board, interface circuit boards, signal conditioning modules, and Graphical User Interface (GUI) software development tools. Menu driven user friendly software, Accelerated Loading Device Instrumentation Software (ALDIS), was developed in C language under Labwindows environment. ALDIS offers attractive GUI features to acquire data from most common types of sensors at various sampling rates, data storage and management in ASCII/Binary format and real-time graphical presentation of the data.

IMPLEMENTATION STATEMENT

A data acquisition system was developed for the Louisiana ALD. This system provides with state-of-the-art tools to monitor and manage data from pavement responses of various research studies scheduled at the PRF. The results from the data collected will help researchers to evaluate materials, construction practices, and performance. In addition, these data can be used to verify mathematical models, develop new mechanistic design procedures, and evaluate in-situ material properties.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES
(CONTINUED)

1. INTRODUCTION

The Louisiana Transportation Research Center (LTRC) is in the process of establishing a Pavement Research Facility (PRF). The core of the PRF is a testing machine that is capable of conducting full-scale simulated and accelerated load testing of either in-service or test section pavement. The Louisiana ALD is an outdoor loading device with an all-weather, computer controlled operation and unassisted movement of the machine in a transverse direction. A photograph of the ALD is shown in Figure 1. The machine is 105 ft (32 m) long, 13 ft (4 m) wide, and 19 ft (5.8 m) high, and weighs 120,000 lb (537.6 kN) fully assembled. The ALD requires little external power to operate because gravity is used to accelerate and decelerate the trolley assembly.

The test section is 40 ft (12.2 m) long, and 13 ft (4 m) wide. Since the facility is outdoors, there are no restrictions on the thickness of the test section and normal highway construction equipment can be used. The ALD applies load to the pavement through one set of dual truck tires in one direction. The load is applied through dead weights of steel plates placed on top of the loading wheels. The load that can be applied to the tires ranges from 9,400 to 22,500 lb (42 to 10 kN). The test wheels travel at 12.5 mph (20.1 kmph) over the entire test section. Loads are applied to the pavement in only one direction of wheel travel. The transverse position of the wheels can be varied across a 48 inch (1.2 m) wheelpath.

The ALD testing system provides a new era in evaluating pavement materials, construction practices, and performance. Results of full scale pavement testing can be used to verify mathematical models, develop new mechanistic design procedures based on full-scale pavement response, and evaluate in-situ material properties under various loading and environmental factors. The Accelerated Loading Device provides researchers with the capability to study the effect of loading on the behavior of the pavement system.

Instrumentation in pavement test sections may be installed during the construction or retrofitted after the construction is completed. Test sections may be instrumented with strain and deflection gauges, pressure and load cells, Linear Voltage Differential Transformer (LVDT),

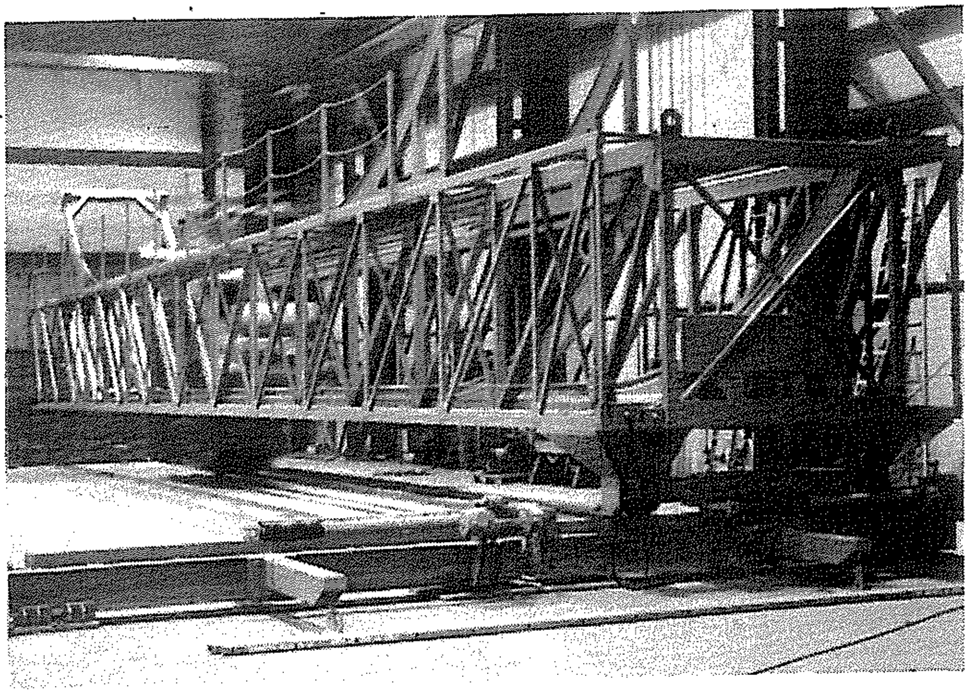


Figure 1. Accelerated Loading Device (ALD) [ref 2]

and temperature and moisture gauges to provide information about pavement materials responses due to traffic loads and environmental effects.

Instrumentation and data acquisition play a vital role in the evaluation of material responses that are used in the characterization of pavement materials under various traffic and environmental conditions.

In order to monitor pavement materials responses due to traffic and environmental factors, a data acquisition system was developed. This system was designed to have the capability of acquiring data from most common types of sensors at various sampling rates, data storage and management in a computer file format to be accessed either in real time or later for further analysis, and graphical presentation of the data using Graphical User Interface (GUI) environment.

2. BACKGROUND

As a result of the popularity of personal computers and low cost workstations, more of them are finding their way into laboratories and factories in data acquisition applications. A typical data acquisition system is presented in Figure 2. A growing array of hardware and software can be used with these computers to configure complete, easy to use, data acquisition and control systems. These products have varying levels of sophistication, performance, and cost. Solutions that look inexpensive are often incomplete. Choosing the right hardware and software is critical to the success of any application.

A variety of I/O products providing a set of building blocks for end users, system integrators, and original equipment manufacturers are available. These products are all designed to be compatible with current industry standard computing platforms, operating systems, and

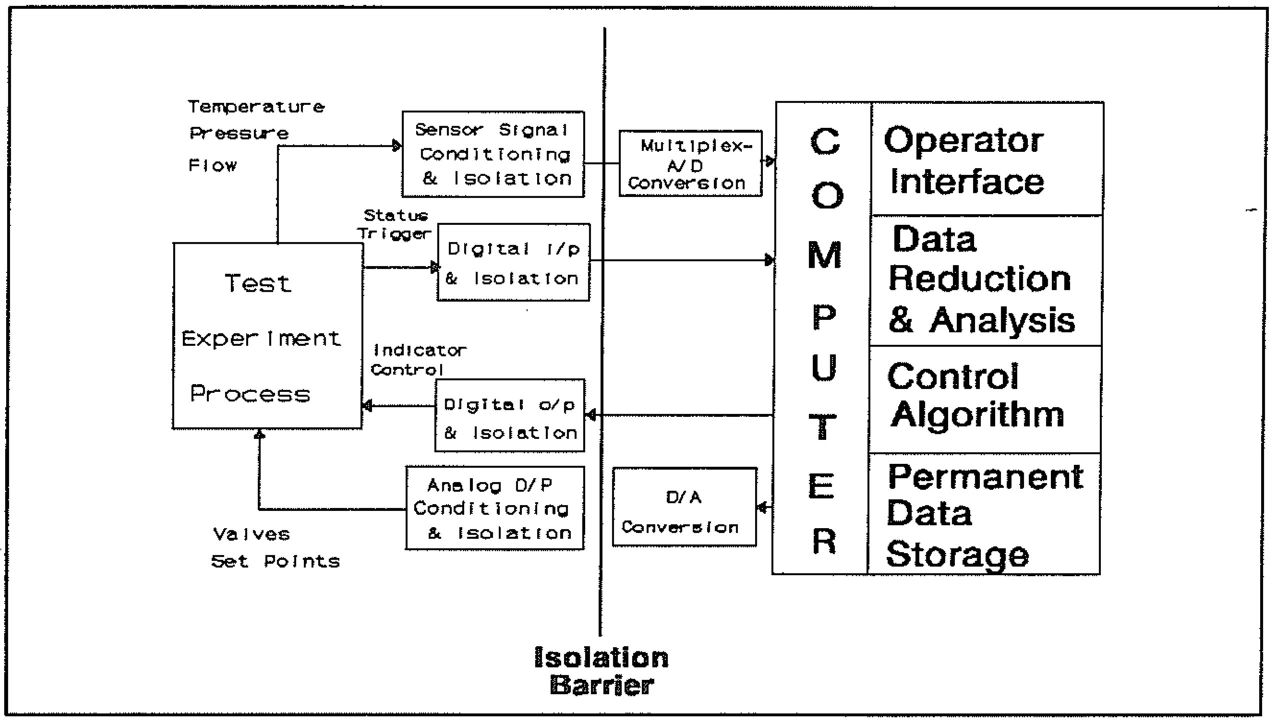


Figure 2. Typical Data Acquisition System [ref 3]

communication standards. A number of different architectures are supported enabling a system to be configured that best meets the needs of virtually any application from simple laboratory data acquisition to distributed process control.

The functions to look for in a data acquisition and control system include; but are not limited to:

- Sensor Signal conditioning
- Isolation
- Analog to Digital Conversion (A/D)
- Digital to Analog Conversion (D/A)
- Data Reduction and Analysis
- Control Algorithms
- Permanent Data Storage

The first four of these features are primarily functions of the signal conditioning and data acquisition hardware used. The last three of these are generally functions of the software, i.e., the program being used. Brief descriptions of these features are discussed below:

Sensor signal conditioning:

Describes the electronics required to convert the varying physical properties of sensors into standard high level signals that can be easily converted by standard A/D converters. These functions include: amplification, completion, excitation, filtering, common mode rejection, protection, linearization, and cold junction compensation.

Isolation:

Guarantees that there is no electrical (galvanic) connection between the sensors on one side of the isolation barrier and the measurement system on the other side. Isolation can eliminate ground loops and improve the safety and reliability of the system.

A/D - D/A Conversion:

Is the function to interface the outside world to a computer. The A/D conversion is done on the principle of successive approximations at different resolutions, while D/A conversion is by means of Staircase Generation Signal with different resolutions.

Data Reduction and Analysis:

Involves using the analytical capabilities of a computer to speed up the process of computation. This includes converting values to engineering units or transforming time based data to the frequency domain, comparing data to desired values, etc. Data manipulation of all types can be done either in "real-time" or as "off-line" analysis.

Control Algorithm:

The rules a system follows to automatically start and stop a process or to alter it by changing an output state. In this way automation can exist without human intervention.

Permanent Data Storage:

It is a record, typically on disk, of the test results or experiment that can be used for required record keeping.

Typically, pavement test sections are instrumented with different types of sensors to measure responses (load related and non-load related). The most critical responses are load related measurements (i.e., deflection, strain, etc.). A typical response curve for longitudinal surface strain is presented in Figure 3. Analysis of such response curves indicated that a sampling rate of 100 Hz would be adequate to capture the entire trace from the point of load application to the point of load removal on the test section.

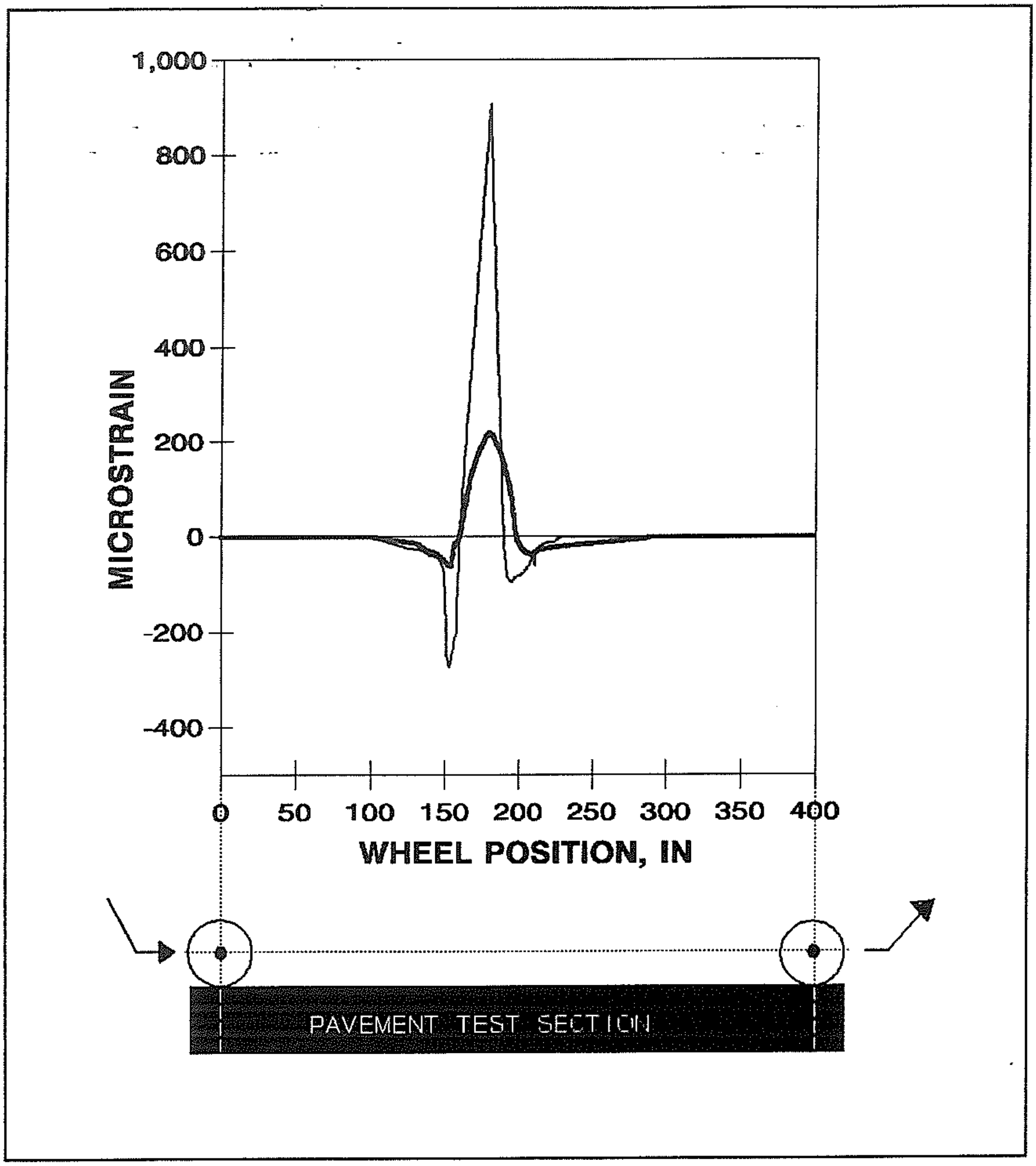


Figure 3. Typical response curve for longitudinal surface strain [ref 2]

3. SYSTEM CONFIGURATION AND SELECTION

3.1 ARCHITECTURAL ALTERNATIVES

Using the best architecture, hardware, and software for a particular application can speed up implementation and insure desired outcome of an application. Although at times it may seem like there are a perplexing array of products for data acquisition and control, there are three basic architectural alternatives for configuring data acquisition and control systems:

- Computer Plug-in I/O
- Distributed I/O
- Distributed Controllers

Computer Plug-in Input/Output (I/O)

Boards often represent the lowest cost alternative and can gather data at the highest speeds. These boards typically have low point counts and rely on the host computer for each operation. They are used in applications where the computer is close to the sensors (within the vicinity of the room) being measured. External signal conditioning is used in conjunction with these boards.

Distributed I/O

Figure 4 presents a block diagram of a Distributed I/O. This system is required in an application where the I/O is far away from the computer (Maximum 4000 Ft). It is an alternative to running long sensor wires which may be very expensive and may compromise the quality of the sensor signal. Different levels of modularity (single point to hundreds per location) and intelligence are available. Standard RS-232C or industrial RS-422/485 are used with these devices, making them compatible with any computer. Distributed I/O is best in higher point count or very distributed applications where high quality signal data is critical and moderate acquisition speeds are adequate.

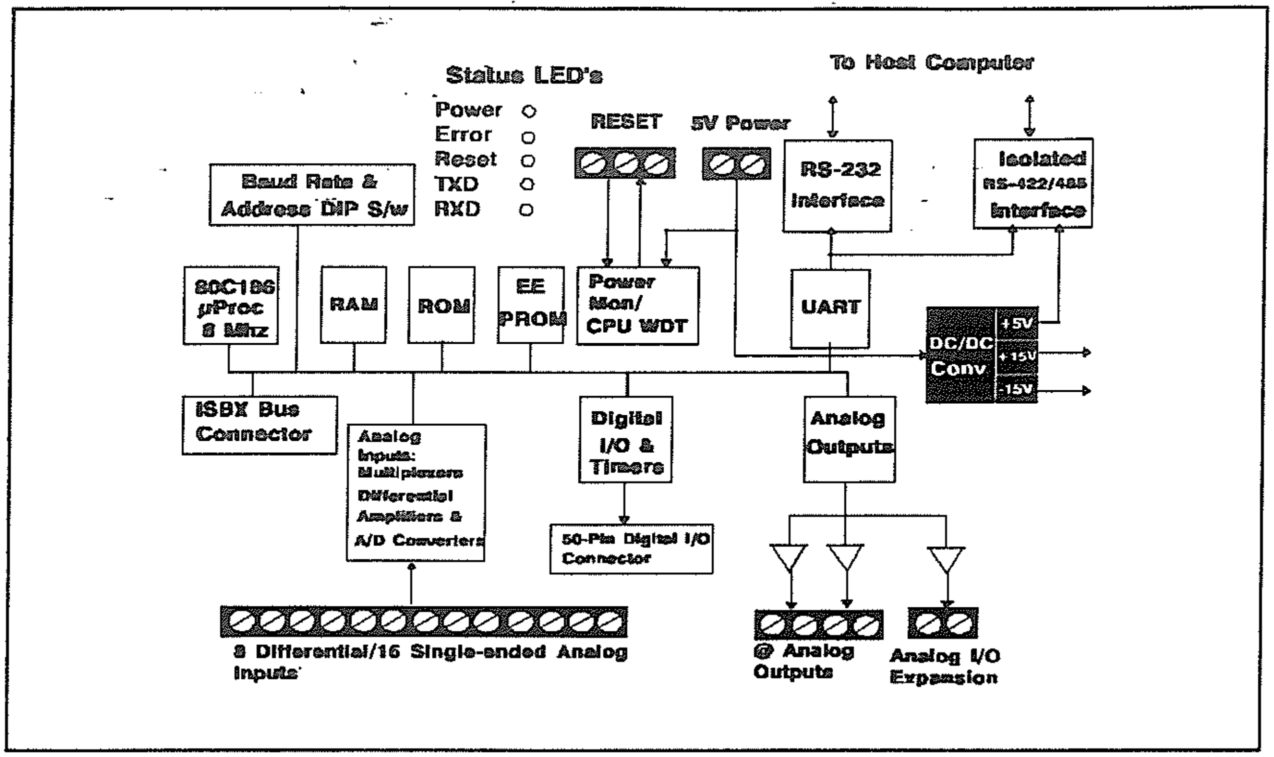


Figure 4. Distributed I/O Subsystem Architecture [ref 6]

Distributed Control

Carries the benefits of distributed I/O plus the ability to make decisions remotely. This increases system reliability because a distributed system can continue to operate even when the host computer is down and increases the overall system performance by distributing the control decisions, algorithms, and other analysis functions to local processors. Other benefits of the distributed control systems include local operator interface and modularity of implementation in multi-unit applications.

3.2 HARDWARE SELECTION

The main drawback of the stand alone distributed control system is in the speed of the data transfer with a host computer. A plug-in type distributed control system, also referred to as Data Acquisition Processor (DAP), was selected to provide high data transfer rate to a host computer while the system is still being used as a front-end processing unit. This combines the desired features of both the stand alone distributed I/O type and plug-in distributed I/O type. The DAP system has parallel bus communication to offer highest possible speeds through First In First Out (FIFO) buffer in data transfer while still providing excellent distributed I/O processing capabilities. The Data Acquisition Processor built around the Intel 80186 CPU running at 10 Mhz with ROM based Real-time Operating System on board offers excellent I/O sub commands for drivers in software development. This hardware platform has the functions of analog to digital conversion in simultaneous sampling and regular sampling mode, digital to analog conversion for control applications, and Digital I/O lines for digital interface between data acquisition systems and the real-world. Also, the DAP does meet the requirements for high speed data acquisition of load related responses as presented in Figure 3. In addition, the simultaneous sampling of a maximum of 16 channels makes DAP an ideal hardware setup. A block diagram of the DAP architecture is shown in Figure 5.

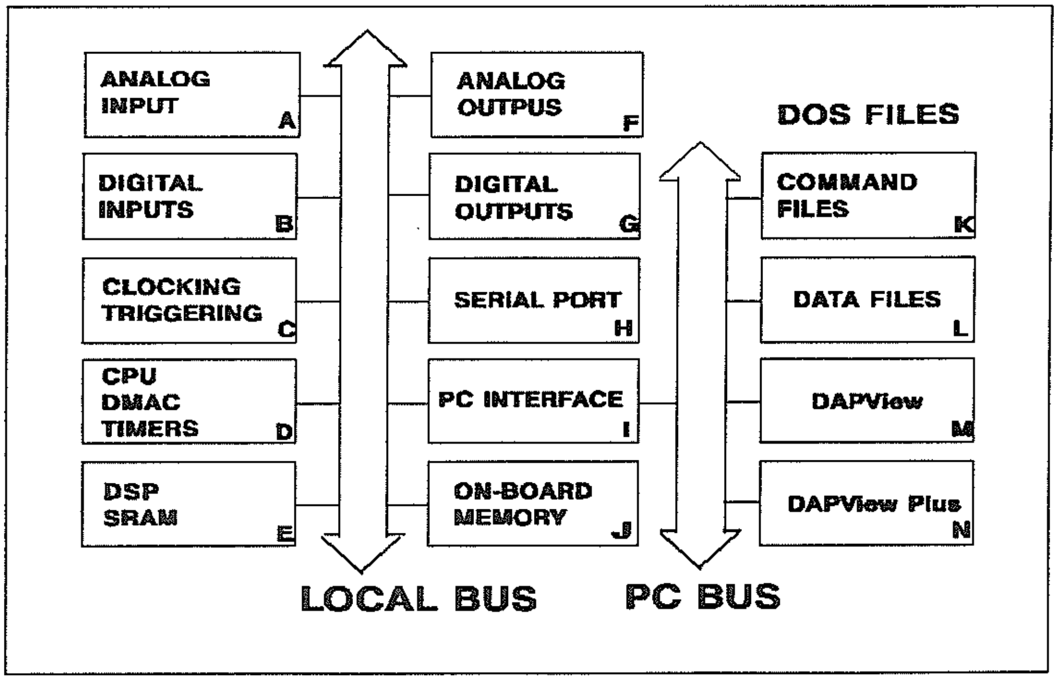


Figure 5. Data Acquisition Processor (DAP) Architecture [ref 7].

3.2.1 Hardware Description

A typical diagram of a pavement section instrumented with strain gauges interfaced to the DAP is presented in Figure 6. The Personal Computer (PC) selected for the plug-in DAP is a Compaq 486. The PC offers the necessary fixed storage and processor speed for optimum use of the data acquisition system.

The data acquisition board selected is a Microstar Laboratories model DAP 1200/4S. This 12 bit resolution board provides 16 analog input channels, single-ended or differential with software selectable by channel and expansion to a maximum of 512 input channels through the use of a multiplexer scheme. There are 16 digital inputs to this board that are synchronous and expandable to 64 inputs. In addition, the board has two 12 bit analog outputs, and 16 bit digital outputs. Programmable gains of 1, 10, 100, and 1000 are available for each channel. The DAP has high speed analog to digital converter (typically of 5 μseconds) and therefore offers a high sampling rate of 156 kHz. The DAP model 1200/4S can support multi-board synchronization among DAP-1200 models and this feature allows any number of analog and digital I/O to be multiplexed among DAP boards. The Operating System on DAP offers programming means in setting parameters such as sampling rate, individual channel gain, single ended/differential, etc.

The DAP combines analog data acquisition hardware with a 16 bit microprocessor, a large buffer memory, and a real time multi-tasking operating system called Data Acquisition Programming Language (DAPL). The DAP handles all the low level details of data acquisition while performing computations in real-time. This frees the PC for user interaction and file management. The high speed First In First Out (FIFO) buffer is the communication path with the host personal computer. The Direct Memory Access (DMA) device transfers data between the DAP system memory and the communication pipe.

The sensory data is fed into modular signal conditioners which are devices that are used to translate a sensor's physical properties to either a digital value or a high level analog output that can be easily digitized. These modules also output current loop in 0-20 ma/4-20 ma current standards. These current loop outputs are relatively immune to noise and can travel over several

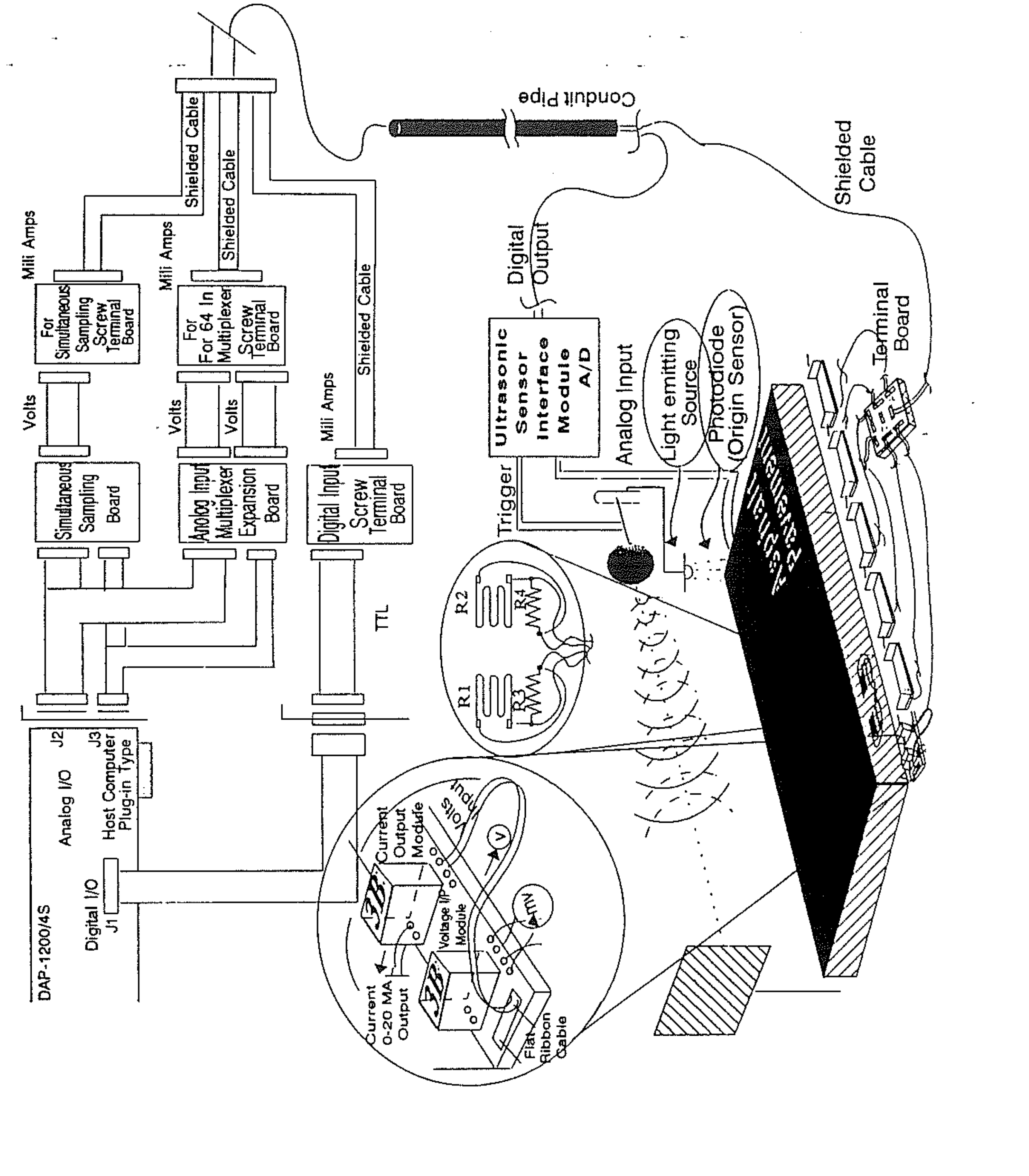


Figure 6. Instrumented Pavement section

thousand feet without signal degradation. This feature makes the high performance signal conditioning system perfect for long distance applications. All these isolated modules provide ±1500 volt isolation and individual channel modularity.

The 0-20 ma loop current is carried to the DAP accessory board for voltage conversion. A precision resistor is used to drop the current, and the voltage across it will be measured as analog input signal by the DAP.

Two types of interface circuit boards were utilized in the system development: analog input expansion board, and analog input screw termination board.

The fault-protected analog input expansion board multiplexes 64 analog inputs into a DAP. As many as 8 analog input expansion boards can be connected to a DAP to provide up to 512 analog inputs.

The analog screw termination board is a 40 point quick connect termination board for analog signals. It provides access to all connections on the DAP analog connector. The analog termination board provides a ground connection for each input and output signal, allowing easy connection to discrete devices. The analog termination board can also be used for differential inputs. A differential input is used to measure the difference between two voltages. The negative terminal voltage is subtracted from the positive terminal voltage.

3.3 SELECTION OF SOFTWARE DEVELOPMENT TOOL

While the hardware used in data acquisition enables a computer to gather data from and control real-time events, it is the software that provides the instructions concerning what actions to take at any point in time.

Software transforms the PC and data acquisition hardware into a complete data acquisition, analysis, and display system. The development of such software involves application specific routines and interactive programs interfaced by system library functions. One such software development environment is shown in Figure 7.

The main criteria set for the selection of the development software was the ease of programming to take full advantage of the DAP board specifications in the GUI environment. Thus, National Instrument's LabWindows development software was selected to meet this objective.

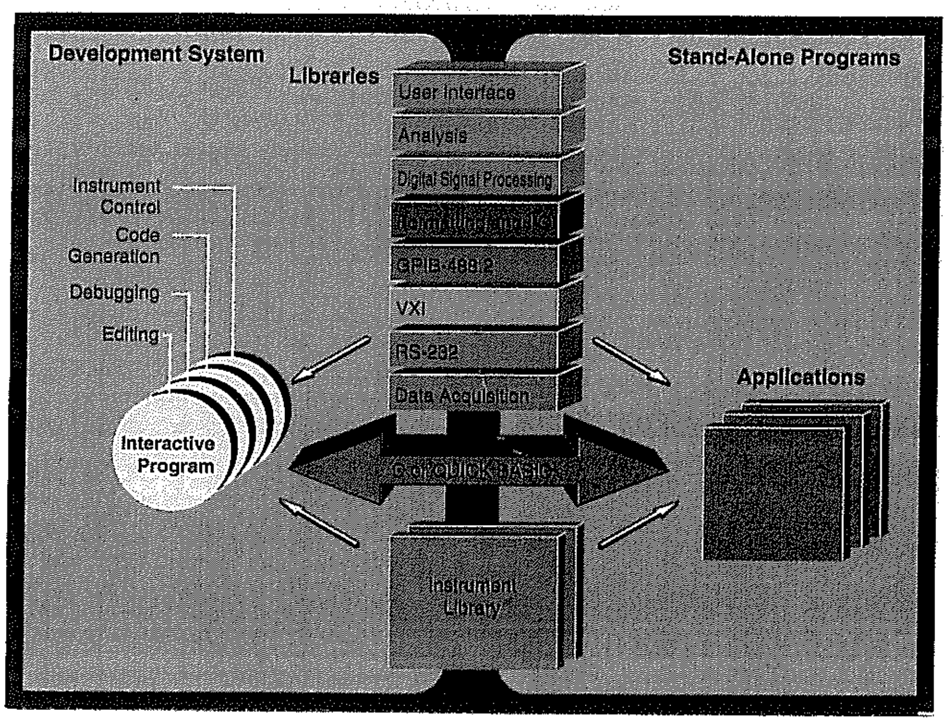
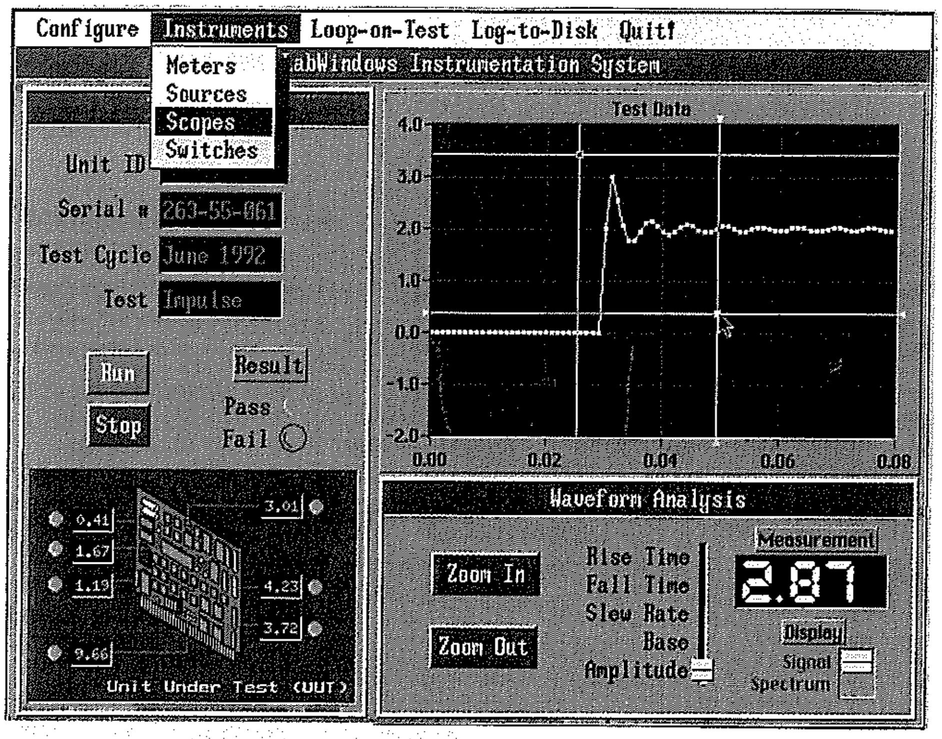


Figure 7. A typical Software Development System Environment [ref 5].

Included as part of the driver software are the instrument panel design routines. With usage of these routines one can be up and running with analog, digital, and counter/timer I/O monitoring routines under an intuitive menu-driven system. Application software packages add analysis and presentation capabilities to the driver software. These packages also integrate instrumentation control into data acquisition in designing multiple instrument panels like the one shown in Figure 8.

The Advanced Development Toolkit software provides the platform for GUI programs,



Unit ID
Serial # 263-55-061
Test Cycle June 1992
Test Impulse

Measurement 2.87

Figure 8. Instrument panel design software utilities [ref 5]

custom driver commands to meet a specific task, and many more features for window programming.

3.3.1 SOFTWARE DESCRIPTION

The data acquisition software developed in this study is the Accelerated Loading Device Instrumentation Software version 1.0 (ALDIS). ALDIS is a state-of-the-art integrated tool for data acquisition, data storage, and data presentation. The GUI associated with the ALDIS offers excellent display features with instrument panels like, strip charts, XY graphs, digital panel meters, rotary and binary switches, push buttons, etc.

The ALDIS supports individual and group channel operations with user programmable features such as sampling rate, single /differential input signal, channel type, scale factors and offset, and time delay. It also offers programmable data storage formats like file format in ASCII/Binary, converted data in US/SI units standards, and data storage file name.

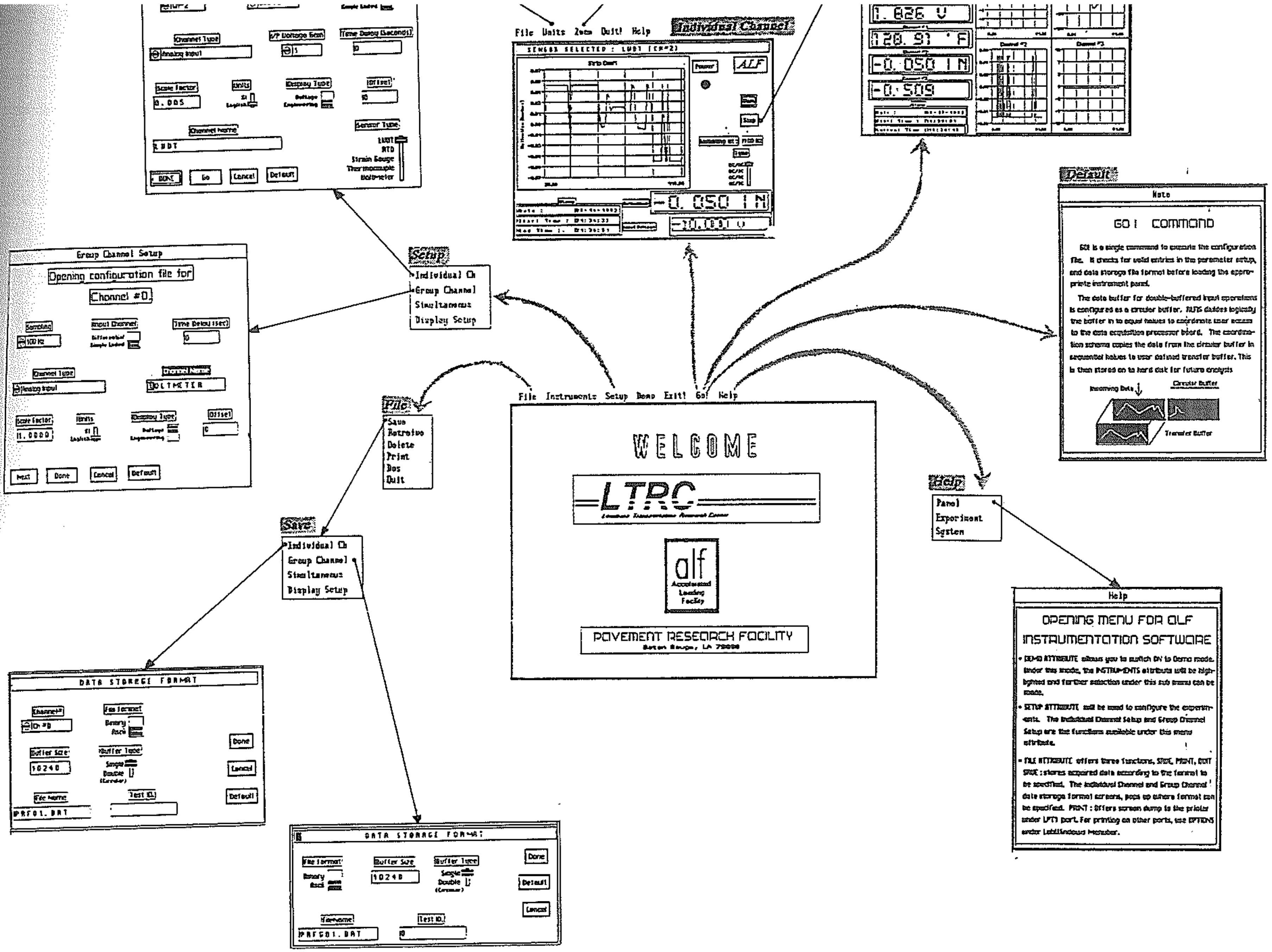
The ALDIS, developed in C language and compiled, runs on a personal computer under Labwindows environment. The highly user friendly graphical user interface takes the user through the software set-up features effortlessly and the online text help provides useful notes. Calibration and zeroing of the various sensors are performed in the individual channel mode. The organization of the ALDIS is presented in Figure 9.

The user needs to define the configuration of the experiment using the selection controls on setup panels. These configurations include Individual channel and Group channel Setup, Display Setup, and File Management Options. Figure 10 shows the different parameters than can be configured with each individual channel. The different parameters for channel configuration is summarized here below:

Configuration Parameters

- *Sampling Rate*

The programmable sampling rate can be set to be 1Hz, 2Hz, 5Hz, 10HZ, 20Hz, 50Hz, 100Hz, 200Hz, and 500Hz. The ALDIS compiles the program dynamically depending on these



Note

GO! COMMAND

GO! is a single command to execute the configuration file. It checks for valid entries in the parameter setup, and data storage file format before loading the appropriate instrument panel.

The data buffer for double-buffered input operations is configured as a circular buffer. ALFIS divides logically the buffer in to equal halves to coordinate user access to the data acquisition processor board. The coordination schema copies the data from the circular buffer in sequential halves to user defined transfer buffer. This is then stored on to hard disk for future analysis.

Help

OPENING MENU FOR ALF INSTRUMENTATION SOFTWARE

- DEMO ATTRIBUTE allows you to switch ON to Demo mode. Under this mode, the INSTRUMENTS attribute will be highlighted and further selection under this sub menu can be made.
- SETUP ATTRIBUTE will be used to configure the experiments. The Individual Channel Setup and Group Channel Setup are the functions available under this menu attribute.
- FILE ATTRIBUTE offers three functions, SAVE, PRINT, EXIT SAVE : stores acquired data according to the format to be specified. The Individual Channel and Group Channel data storage format screens, pops up where format can be specified. PRINT : Offers screen dump to the printer under LPT1 port. For printing on other ports, use OPTIONS under LabWindows Menubar.

Figure 1.2 ALDIS software organization

ALFIS OPERATION

parameters as an input procedure for real-time operating systems running on DAP.

• *Scale factor*

This parameter converts the raw data (volts) from the interface device to an appropriate engineering unit. The scale factor value defines the conversion for an input of 1.0 volt. The scale factor for LVDT with full scale of 0.05 inches for 10 volts would be 0.005.

• *Offset*

This entry defines the offset value for unconverted data and the default value is set to 0. The Scale Factor and the Offset entries together convert raw data to scientific or engineering units according to the formula:

Converted Data = ((Raw Data / Channel Gain) + Offset Constant) * Scale Factor

• *Channel Gain Factor*

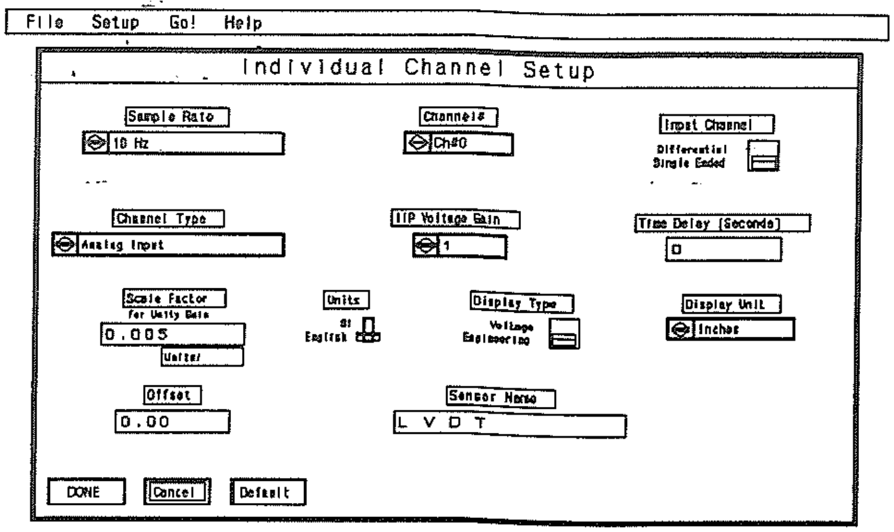
The input amplifier provides different gain factors of 1,10,100, and 1000 for each input channel. This parameter can be set to raise the level of the input signal to the desired value before it can be acquired and stored into a file. This parameter increases the resolution of input channels below 10 volts.

• *Time Delay*

This parameter defines the time delay in seconds for the start of acquisition for a particular channel.

• *Channel Name*

This entry is a name, up to 15 characters long, that appears in the individual channel setup. The channel name entry is present for convenience when documenting the setup and is optional.



Individual Channel Setup

Sample Rate: 10 Hz
Channel#: Ch#0
Input Channel: Differential / Single Ended
Channel Type: Analog Input
I/P Voltage Gain: 1
Time Delay (Seconds): 0
Scale Factor for Unity Gain: 0.005 Units/
Units: SI / English
Display Type: Voltage / Engineering
Display Unit: Inches
Offset: 0.00
Sensor Name: L V D T

Figure 10. Setup Option for Individual Channel

· *Channel Gain Factor*

The input amplifier provides different gain factors of 1,10,100, and 1000 for each input channel. This parameter can be set to raise the level of the input signal to the desired value before it can be acquired and stored into a file. This parameter increases the resolution of input channels below 10 volts.

· *Time Delay*

This parameter defines the time delay in seconds for the start of acquisition for a particular channel.

· *Channel Name*

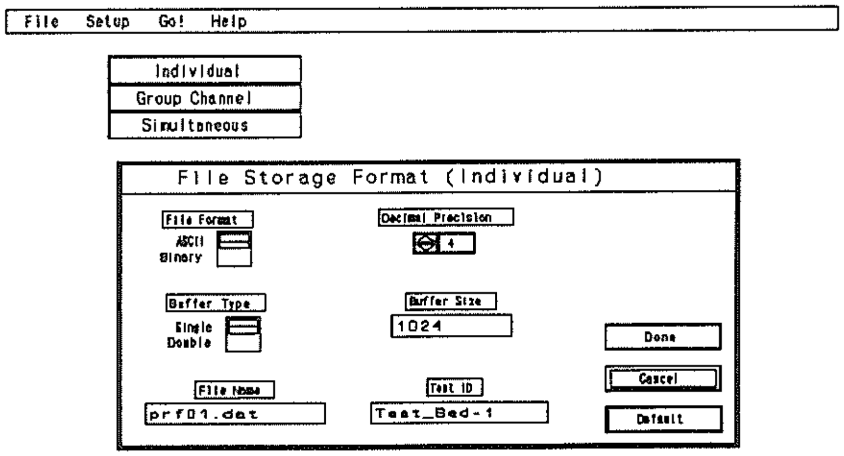
This entry is a name, up to 15 characters long, that appears in the individual channel setup. The channel name entry is present for convenience when documenting the setup and is optional.

• *Channel Type*

There are four channel types, two input and two output types. Analog input and digital input form the input section of the data acquisition system.

The group channel configuration is similar to individual channel setup except that ALDIS offers each channel in sequence to be configured. The user can setup each channel with similar parameters as individual channel setup and view all selected channels on one screen with a fixed color for each channel.

Once the data acquisition format is defined under setup the user needs to define the storage format. The Save option under File Menu takes the user into the individual channel Data Storage Format screen. Here the user can define the data file to be in ACSII/Binary mode, Engineering Units in US/SI standards, and the file name to save data. The GO command under Open Menu checks for valid configuration and entries before starting the test with the instrument panel. The final data file contains complete stamping of date, start time, end time, channel name and type, and time intervals with converted data. Figure 11 shows the Data Storage Format screen.



File Storage Format (Individual)

File Format: [✓ ASCII] Binary

Decimal Precision: 4

Buffer Type: [✓ Single] Double

Buffer Size: 1024

File Name: prf01.dat

Test ID: Test_Bed-1

Figure 11. Data Storage Format

The ALDIS application software at the host computer site has been designed to have a dual buffer data stream for video display and data storage simultaneously. The processed data can be loaded in to secondary memory device such as, steamer tape backup drives for analysis at a later date. The distributed processing power adds up to offering fault tolerant system to an extent possible. The DAP and the host computer have a communication protocol to determine the failure at any end. In the event of the host computer not responding to DAP, it can initiate data transfer through an additional on-board data link (RS232) with another stand-by computer for data storage or man-machine interface.

4. CONCLUSIONS

A data acquisition system was developed for the Louisiana ALD. This system includes both hardware and software. The hardware was a 486 based Personal Computer, plug-in Distributed I/O board, interfacing circuit boards, and signal conditioning modules. The current configuration of the board has the capability to acquire 64 input signals with 16 simultaneous sampling, two analog output signals, 16 digital input/outputs. The maximum possible analog input signals is 512.

Menu driven user friendly software, ALDIS was developed using C language under Labwindows environment. ALDIS has features to acquire data from most common types of sensors at various sampling rates in either the individual or grouped mode. Data storage and management was performed in ASCII or binary format. GUI was utilized for real-time graphical presentation of the data.

5. REFERENCES

1. Tabatabaee, N., Sebaaly, P., and Scullion, T., "Pavement Testing Facility, Instrumentation for Flexible Pavements", Report No FHWA/RD/89-084, Federal Highway Administration, Mclean, Virginia, 1989.
2. Bonaquist, R., Surdhal, R., and Mogawer, W., "Pavement Testing Facility - Effects of Tire Pressure on Flexible Pavements", Report No. FHWA/RD/89-084, Federal Highway Administration, McLean, VA, 1989.
3. Analog Devices, "Data Acquisition and Control Catalog", Analog Devices, Inc. Norwood, MA, 1990.
4. Analog Devices, "Supplementary Catalog on Data Acquisition and Control" Analog Devices, Inc. Norwood, MA, 1990.
5. National Instruments, "Catalog on IEEE-488 and VXI Bus Control, Data Acquisition and Analysis", National Instruments Corporation, Austin, TX, 1992.
6. Cyber Research - "PC Systems Handbook for Scientists and Engineers", Volume 8, Page 86, Ciber Research, Inc. Branford, CT, 1991.
7. "The Intelligent Solution for Data Acquisition", MicroStar Laboratories, Bellevue, WA, 1991.
8. "Data Acquisition Processor Hardware Manual", Analog Accelerated Series, MicroStar Laboratories, Bellevue, WA, 1991.

APPENDIX A

LOUISIANA TRANSPORTATION RESEARCH CENTER

A L D I S

Version 1.1

Accelerated Loading Device Instrumentation Software

USERS MANUAL

FEBRUARY, 1993

ALDIS 1.1

This document is a reference manual that contains detailed description of the Accelerated Loading Device Instrumentation Software (ALDIS) features and functionality. ALDIS is a state-of-the art integrated tool for data acquisition, data storage, data presentation. This software is user friendly and menu driven that utilizes Graphical User Interface (GUI) environment. The library functions of the Labwindows instrumentation software have been extensively used for experiment setup and data presentation. ALDIS 1.1 requires a minimum of 386 Personal Computer, with MS-DOS 3.0 or later version disk operating system, Labwindows 2.2, VGA color monitor for interactive data presentation, and a minimum of 20 MB hard disk space for data storage. A knowledge of the use of pointing device (mouse) is desired.

THE ORGANIZATION OF THE ALDIS USER MANUAL:

This user manual is organized as follows:

- Running the Software
- Experiment Setup
    - Individual channel setup
    - Group channel setup

Running the software:

- Type at *C:\LW\PROGRAMS>ALDIS <CR>*
- ALDIS function selection list will appear, Figure 12.

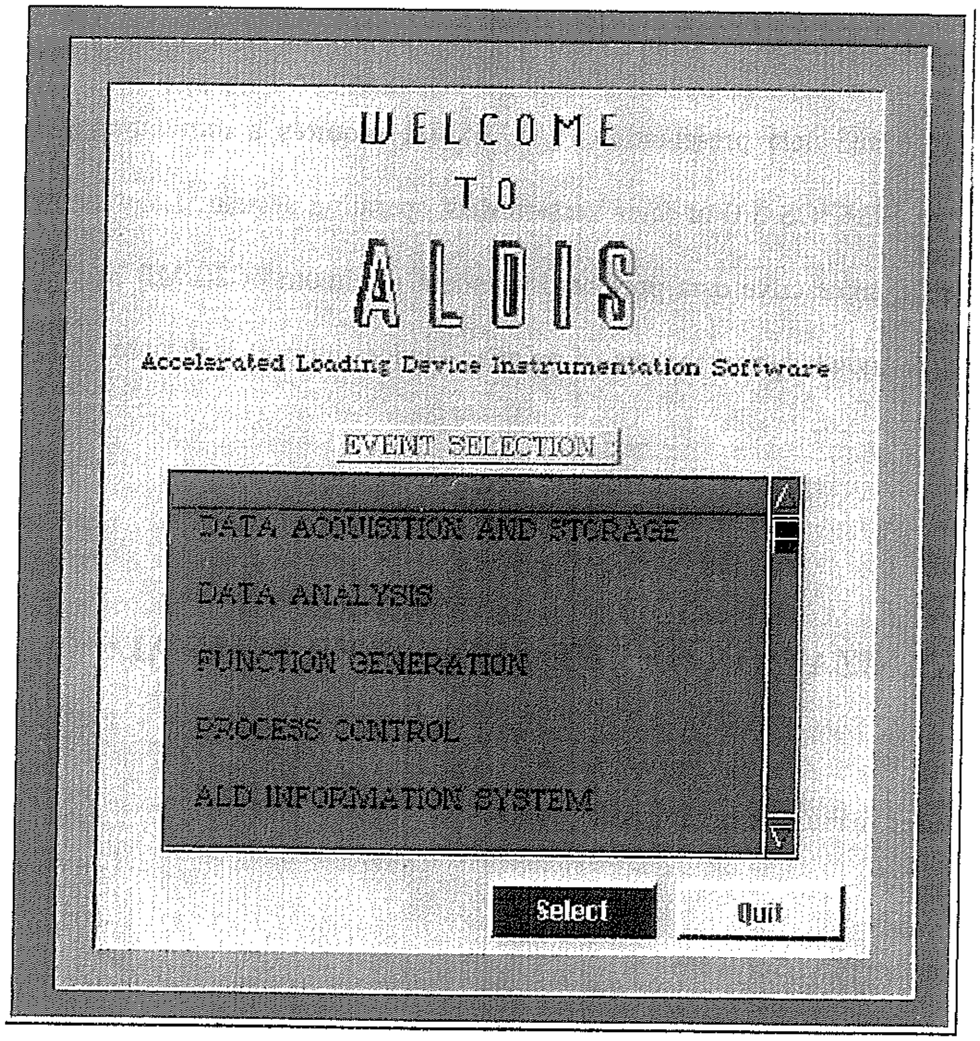


Figure 12. Function Selection Panel

- Select Data Acquisition and Storage function and press *Select*. The Opening Screen as shown in Figure 13 will appear. (Use Arrow keys for up and down cursor movement and Enter keys for selection. Point device; Mouse can also be used for function selection.)

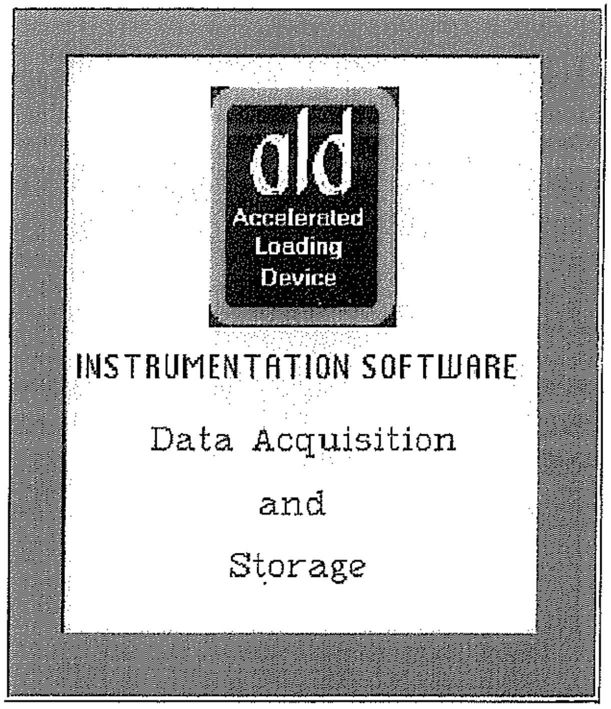


Figure 13. Data Acquisition and Storage Opening Panel

Setup Configuration:

The parameters *Sampling rate, Scale factor, Offset, Time Delay, Gain factor, Channel Type (single/differential), Analog/Digital, File format, and data presentation parameters* needs to be initialized. The value will be stored in a CH*.CFG file.

1.1 Individual channel setup :

Select **Setup** from the opening menu Figure 14, which opens the pop-up menu: **Individual Channel, Group Channel, Display: Individual Channel, and Display: Group Channel.**

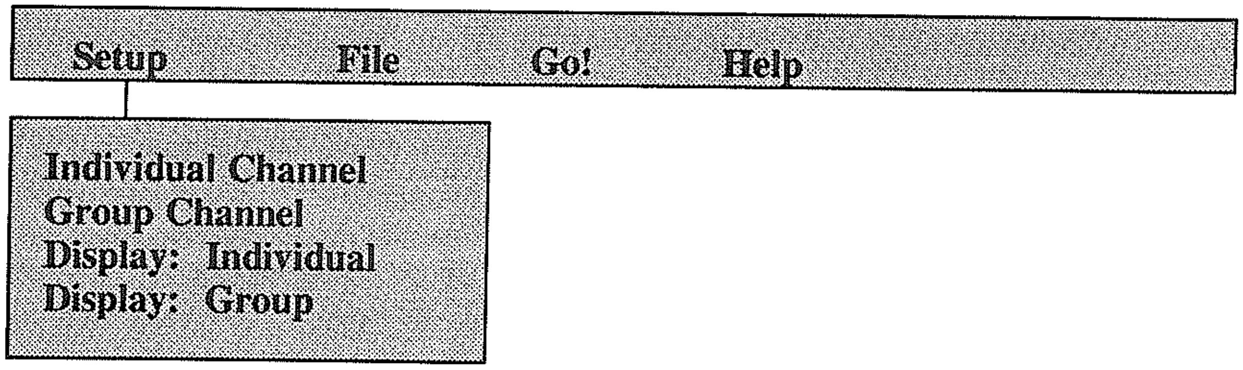
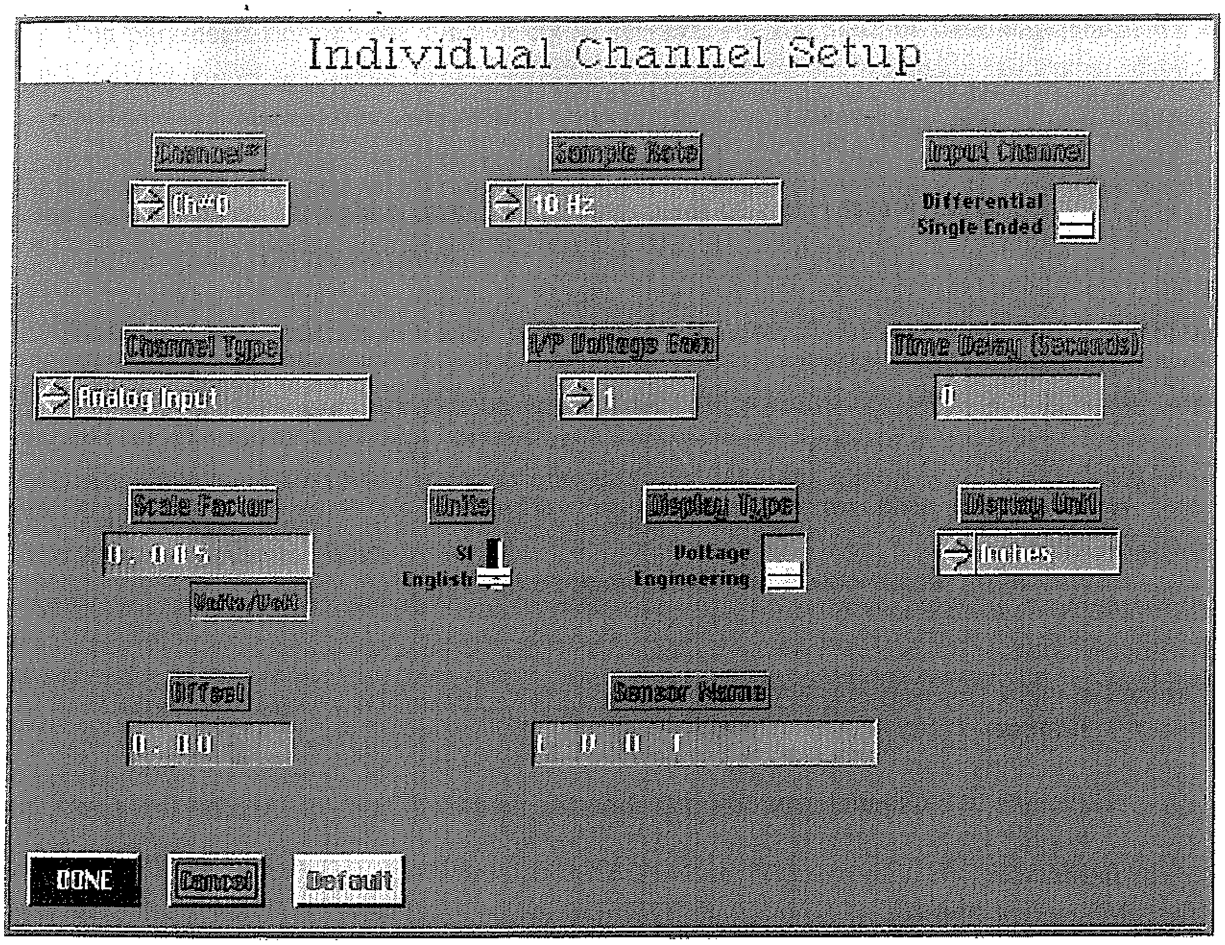


Figure 14. Opening Menu - Setup Pop-up (Individual Channel)

Select **individual channel** from this pop-up menu to enter into individual channel configuration mode. Figure 15 presents an individual channel configuration setup panel. The selection of different controls can be through the *TAB* key on keyboard or through the pointing device (mouse). The up and down arrow keys can be used to select each value in case of ring and slide switches or mouse on the control parameters directly. The numeric fields needs to be entered through keyboard. A description of the individual parameters are discussed below:


Individual Channel Setup

Channel#: Ch#0

Sample Rate: 10 Hz

Input Channel: Differential / Single Ended

Channel Type: Analog Input

I/P Voltage Gain: 1

Time Delay (Seconds): 0

Scale Factor: 0.005 Volts/Volt

Units: SI / English

Display Type: Voltage / Engineering

Display Unit: Inches

Offset: 0.00

Sensor Name: L V D T

Figure 15. Individual Channel Configuration Setup Panel

· *Sampling Rate:*

Sampling rate is the number of samples per second of A/D conversion for the channel selected. It can be selected from the ring switch to one of the following: 1Hz, 2Hz, 5Hz, 10HZ, 20Hz, 50Hz, 100Hz, 200Hz, 500Hz, and 1kHz.

• *Channel Number:*

A numeric entry that refers to the channel number to which the sensor is assigned.

• *Input Channel:*

Two input channel types are available; single-ended and differential-ended. Differential type can be used for noise reduction. This can be S or D (slide/toggle switch) representing single-ended and differential respectively.

• *Channel Type:*

There are two channel types on this slide/toggle switch control; Analog input and Digital input.

• *Input Voltage Gain:*

The input amplifier provides different gain factors of 1,10,100, and 1000 for each input channel. This parameter can be set on this ring switch to raise the level of the input signal to the desired value. For example, a gain of 10 for an input voltage range of 0 to 1 volt, yields an output in the range of 0 to 10 volts and thereby increasing the input resolution.

• *Time Delay:*

This parameter defines the time delay in seconds for the start of data acquisition for a particular input channel. Use keyboard to enter the numeric value on this control.

• *Scale factor, Offset constant:*

These parameter entries are used with almost all types of input channels. It converts raw data from the interface device into appropriate scientific or engineering units. The Scale Factor and the Offset together converts raw data into scientific or engineering units according to the formula:

$$\text{Converted Data} = ((\text{Raw data} / \text{Channel Gain}) + \text{Offset constant}) \times \text{Scale Factor}$$

Example:

Consider a LVDT sensor with full scale deflection of ±0.005 inches/volt, with zero offset, input channel gain of 10, and raw voltage input of 1.0 volts. Then for a full scale input voltage of ±10 volts;

$$\text{Converted Data} = ((1.0 / 10) + 0) \times 0.005$$
$$= 0.0005 \text{ inches.}$$

Note: In the above example the input voltage range with gain=10 can be ±1 volt, i.e. ±0.005 inches of deflection.

• *Display Type:*

Display type is a slide/toggle switch for Voltage or Engineering unit selection. Voltage selection disables Display Unit control and the input channel will be configured as a voltmeter in ± 10 volts range.

• *Display Unit:*

Select either SI variation or English variation for the display of the units, for the channel selected.

• *Sensor Name:*

Provides an identification for the interface channel. It is a string entry with a maximum of 15 characters.

• *Push buttons at the bottom left corner of the panel:*

Done initializes all the parameter variables, and stores on to a configuration file; *CH#*.CFG* for future retrieval.

Cancel terminates the setup operations and returns to panel in Figure 13.

Default resets the control values on the current panel with its initial values for fresh entry.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons in case Tab key is used for selection.

1.2. File Storage Format (individual channel):

Select **File** from the opening menu Figure 16, which opens the pop-up menu: **Load, Save, Print, Dos, and Quit.**

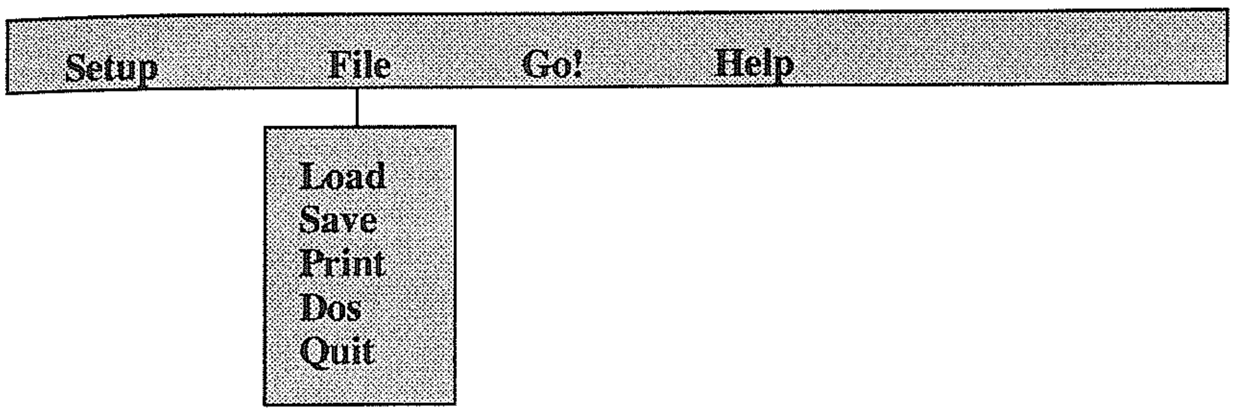
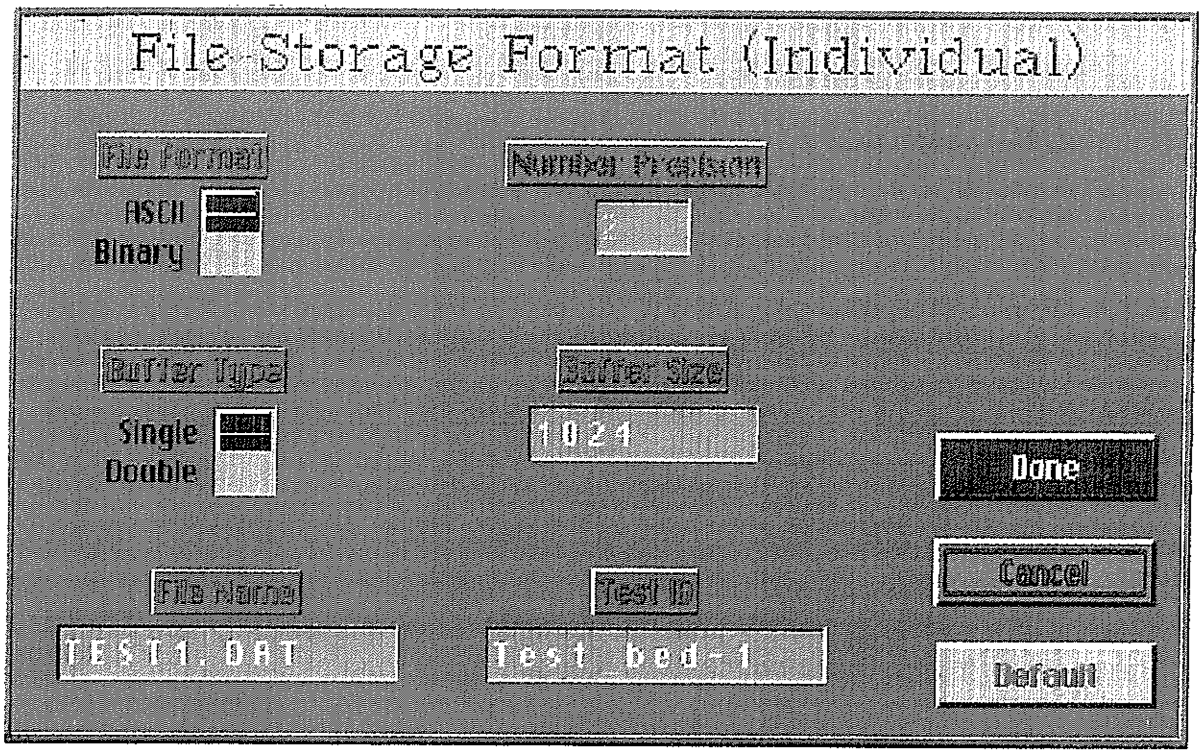


Figure 16. Opening Menu - File Pop-up (Individual Channel)

Select **Save** to define the parameters of File Save. Figure 17 presents an individual file storage format setup panel. The selection of different controls can be through the *TAB* key on keyboard or through the pointing device (mouse). The arrow keys can be used to select each value in case of ring and slide switches, or point with the mouse on the control parameters directly, while numeric fields needs to be entered through keyboard. A description of the individual parameters



Buffer Size: 1024

File Name: TEST1.DAT

Test ID: Test_bed-1

Figure 17. Individual Channel File Format Setup Panel

are discussed below:

· *File Format:*

The slide/toggle switch selects either ASCII or Binary file format. Binary format can be selected if data compaction is required during storage.

• *Decimal Precision:*

Decimal precision is the number of decimals that is desired for storage. The decimal precision numbers provided on this ring switch are 1,2,3,4,5, and 6.

• *File Name:*

This entry is a file name of 15 characters long. The data is stored on the hard disk under this File Name. The default is set to *prf01.dat*. Use the keyboard to enter the string variable.

• *Test ID:*

This entry is used for experiment identification and is of 25 characters long.

Figure 18 shows a typical format of the stored data file.

• *Push buttons at the bottom left corner of the panel:*

Done: initializes all the parameter variables, and stores on to a configuration file; *CH#.CFG* for future retrieval.

Cancel: terminates the setup operations and returns to panel in Figure 13.

Default: resets the control values on the current panel with its initial values.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons.

Date : 02-01-1993

Sensor Name: VOLTMETER
Sampling Time Interval: 0.1 Sec (Frequency: 10 Hz)
Test ID: TEST ID NOT SPECIFIED
Channel Type: Analog Input
P r o c e s s D a t a

Test Started Time(h:m:s) : 12:07:54

Time Seconds	Units Volt
0.0	2.0410
0.1	2.1387
0.2	2.2266
0.3	2.3096
0.4	2.3828
0.5	2.4512
0.6	2.5195
0.7	2.5879
0.8	2.6514
0.9	2.7100
1.0	2.7734

Test Started Time(h:m:s) : 12:07:54
Test Ended Time(h:m:s) : 12:09:01

Operator Note : NO NOTE

Figure 18. Stored Data File (Individual Channel)

1.3. Display Setup (individual channel):

Select **Setup** from the opening menu Figure 19, which opens the pop-up menu: **Individual Channel, Group Channel, Display: Individual Channel, and Display: Group Channel.**

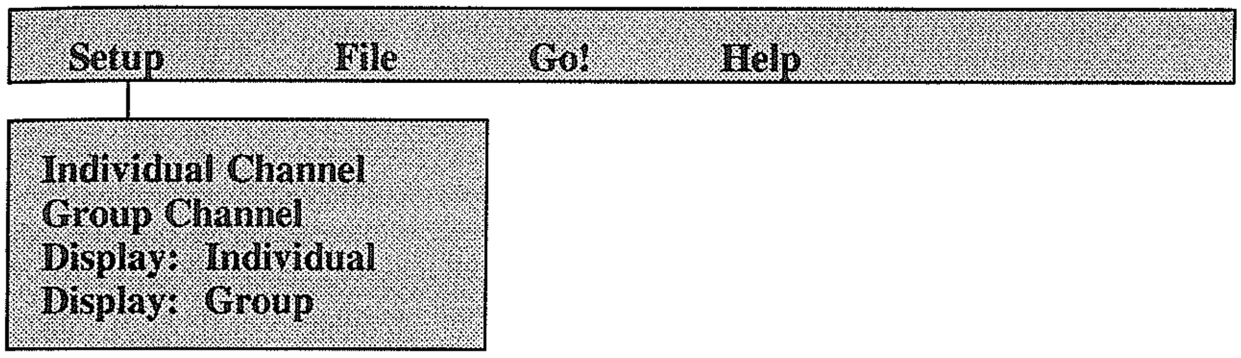


Figure 19. Opening Menu - Setup Pop-up (Individual Channel)

Select **Display Individual Channel** from this pop-up menu. The real-time data presentation format for individual channel setup is shown in Figure 20. There are two instrumentation panel support for data presentation: Strip chart and Digital Panel Meter (DPM). The selection of different controls can be through the *TAB* key on keyboard or through the pointing device (mouse). The arrow keys can be used to select each value in case of ring and slide switches, or mouse on the control parameters directly. The numeric fields needs to be entered through

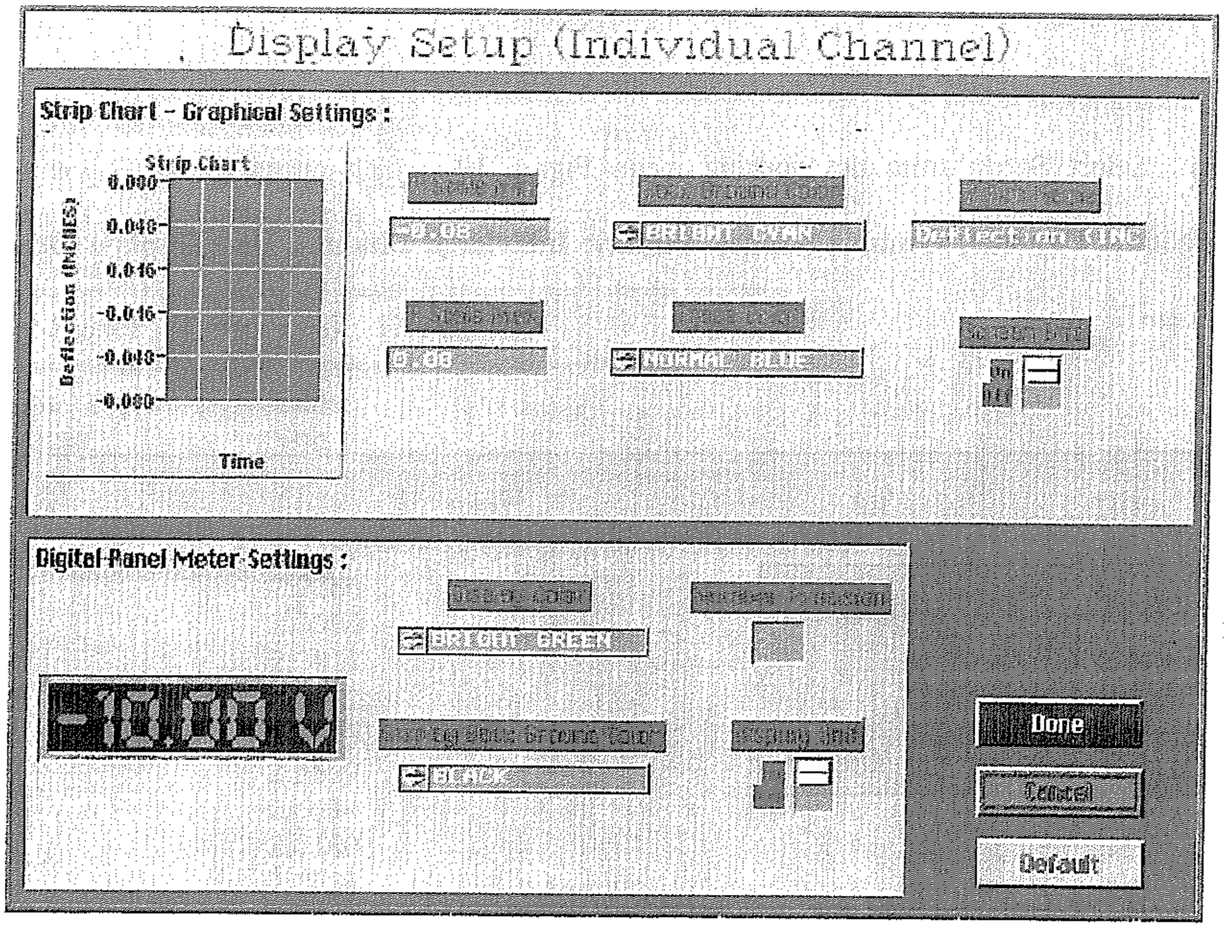


Figure 20. Individual Channel Display Format Panel

the keyboard. A description of the individual parameters are discussed below:

Strip Chart:

- *Y Axis Scale (Minimum, Maximum):*

This parameter control the strip chart's Y-axis scale with the minimum and Maximum value that can be presented. These values can also be changed in real-time during experiment.

• *Background Color:*

The strip chart background color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample strip chart.

• *Y Axis Name*

This entry is Y axis name of 25 characters long. The axis name entry is presented for presentation convenience and is set to default *Voltage*.

• *Trace Color*

The strip chart trace color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample strip chart.

• *Screen Grid On/Off:*

This parameter can be used to toggle the Screen Grid to On or Off.

Digital Panel Meter (DPM):

• *Background Color:*

The DPM background color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample DPM.

• *Text Color:*

The DPM text color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample DPM.

• *Decimal Digit Precision:*

This numeric value (1,2,3,4,5, and 6) sets the decimal precision on the 7-segment digital display.

• *Display Unit:*

This can be used to toggle the display of engineering unit On or Off on DPM.

• *Push button at the bottom left corner of the panel:*

Done initializes all the parameter variables, and stores on to a configuration file; *CH#*.CFG* for future retrieval.

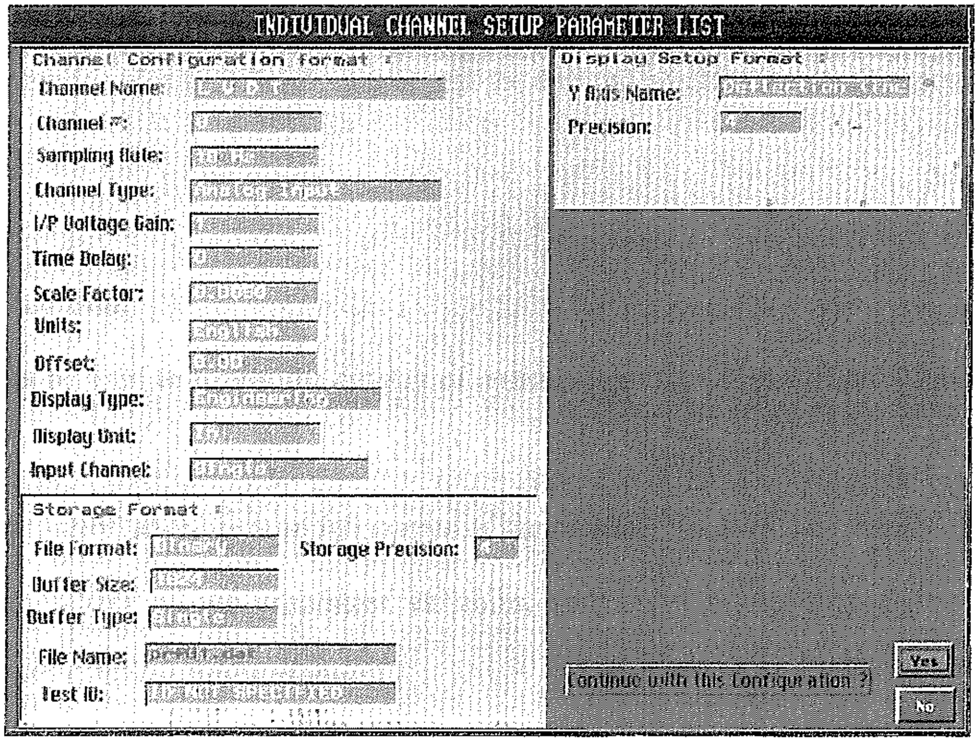
Cancel terminates the setup operations and returns to panel in Figure 13.

Default resets the control values on the current panel with its initial values.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons.

1.4. The *'GO'* Command:

The command *GO!* on opening menu (Figure 13) starts the single channel data acquisition process as configured. The *'GO'* command checks for completeness of the parameters initialized and lists the settings on the screen as shown in Figure 21. It prompts with the useful messages if the setup is incomplete. If no changes are to be made, then press *Yes* to continue. An instrument panel will appear, Figure 22. Press *Power* push button to illuminate the instrument panel which indicates the power-on (ready-to-run) state. Then press *Run* push button to start the data acquisition, storage, and presentation processes. An active


INDIVIDUAL CHANNEL SETUP PARAMETER LIST

Channel Configuration format :

Channel Name:	[illegible]
Channel #:	[illegible]
Sampling Rate:	[illegible]
Channel Type:	Analog Input
I/P Voltage Gain:	[illegible]
Time Delay:	[illegible]
Scale Factor:	[illegible]
Units:	[illegible]
Offset:	0.00
Display Type:	[illegible]
Display Unit:	[illegible]
Input Channel:	[illegible]

Storage Format :

File Format:	[illegible]
Storage Precision:	[illegible]
Buffer Size:	[illegible]
Buffer Type:	[illegible]
File Name:	[illegible]
Test ID:	NOT SPECIFIED

Display Setup Format :

Y Axis Name:	[illegible]
Precision:	[illegible]

Continue with this Configuration ?

Figure 21. Individual Channel Parameter List

instrument panel is shown in Figure 22.

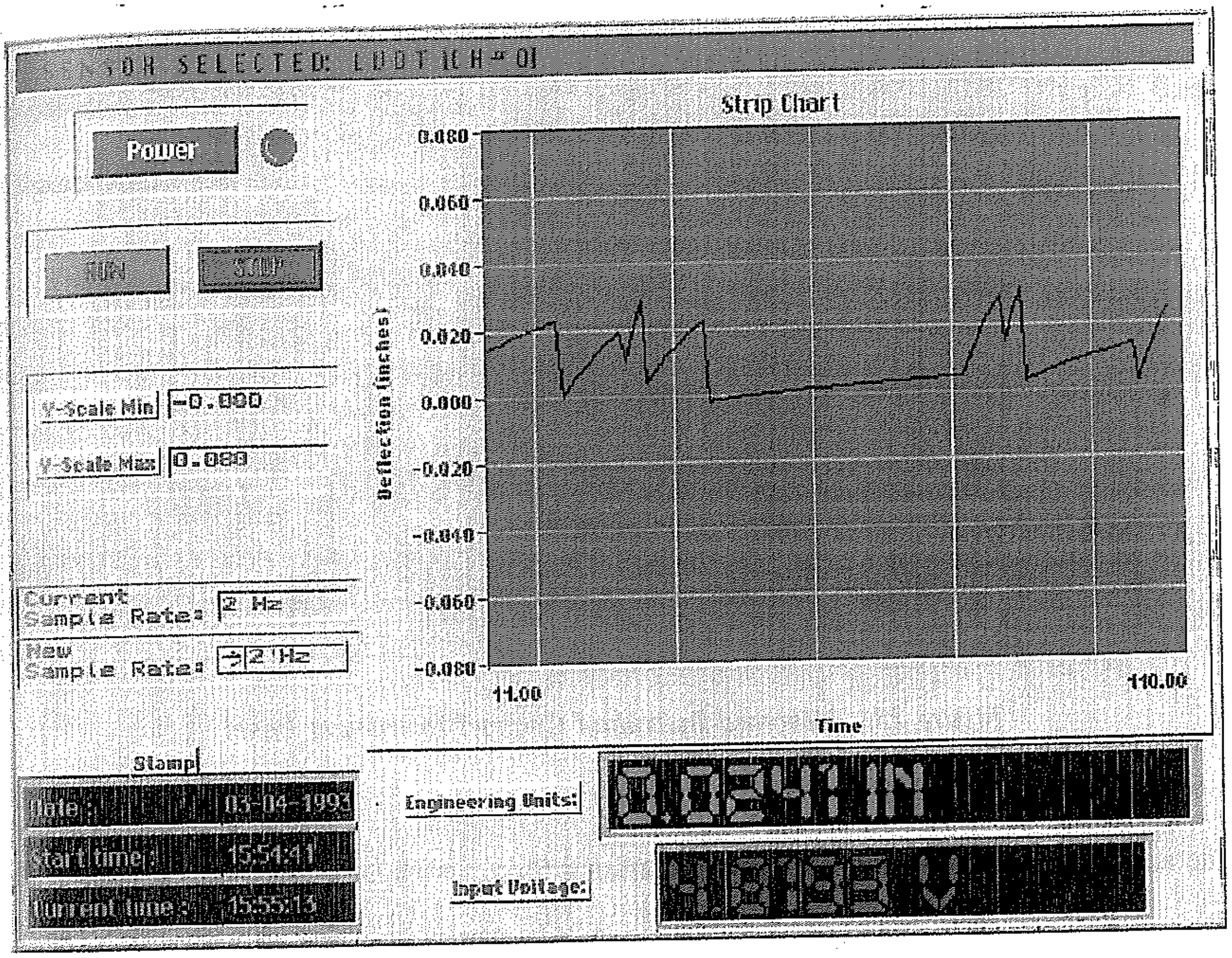


Figure 22. *Individual Channel Instrument Panel*

The Zoom option on the command line can be used to view the graph on a full screen as shown in Figure 23, which include a DPM.

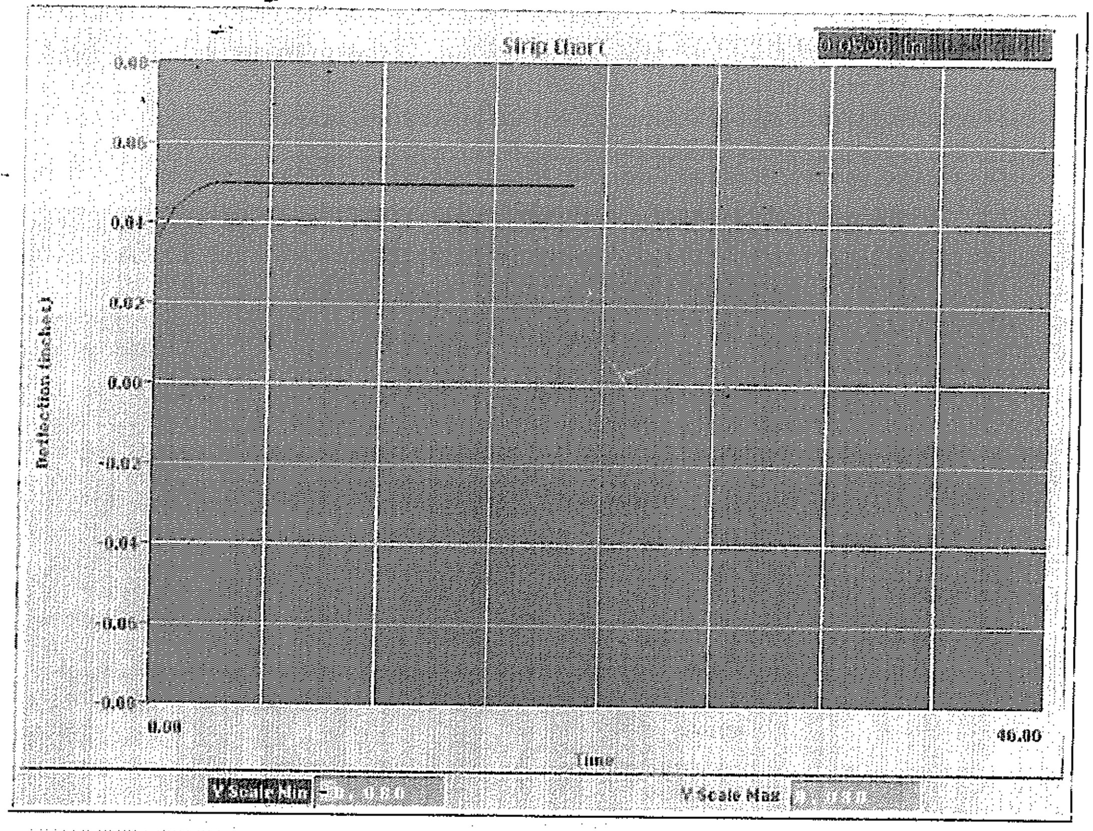
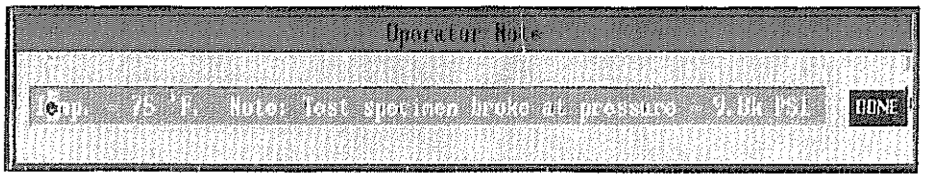


Figure 23.   Zoomed Individual Channel Instrument Panel

The acquisition of data can be stopped by pressing Stop push button. On termination of the experiment using *File - Quit* an operator note window will appear, Figure 24. It accepts 256 character note which is appended to the data file.



Temp. = 75 °F. Note: Test specimen broke at pressure = 9.0k PSI

Figure 24.   Operator Note Window

2.1 Group Channel Setup :

Select **Setup** from the opening menu, Figure 25, which opens the pop-up menu: **Individual Channel, Group Channel, Display: Individual Channel, and Display: Group Channel.**

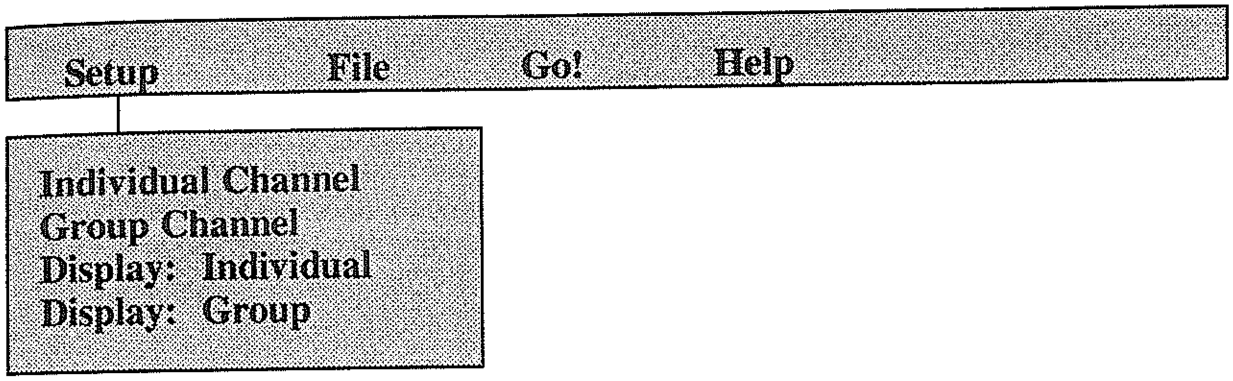
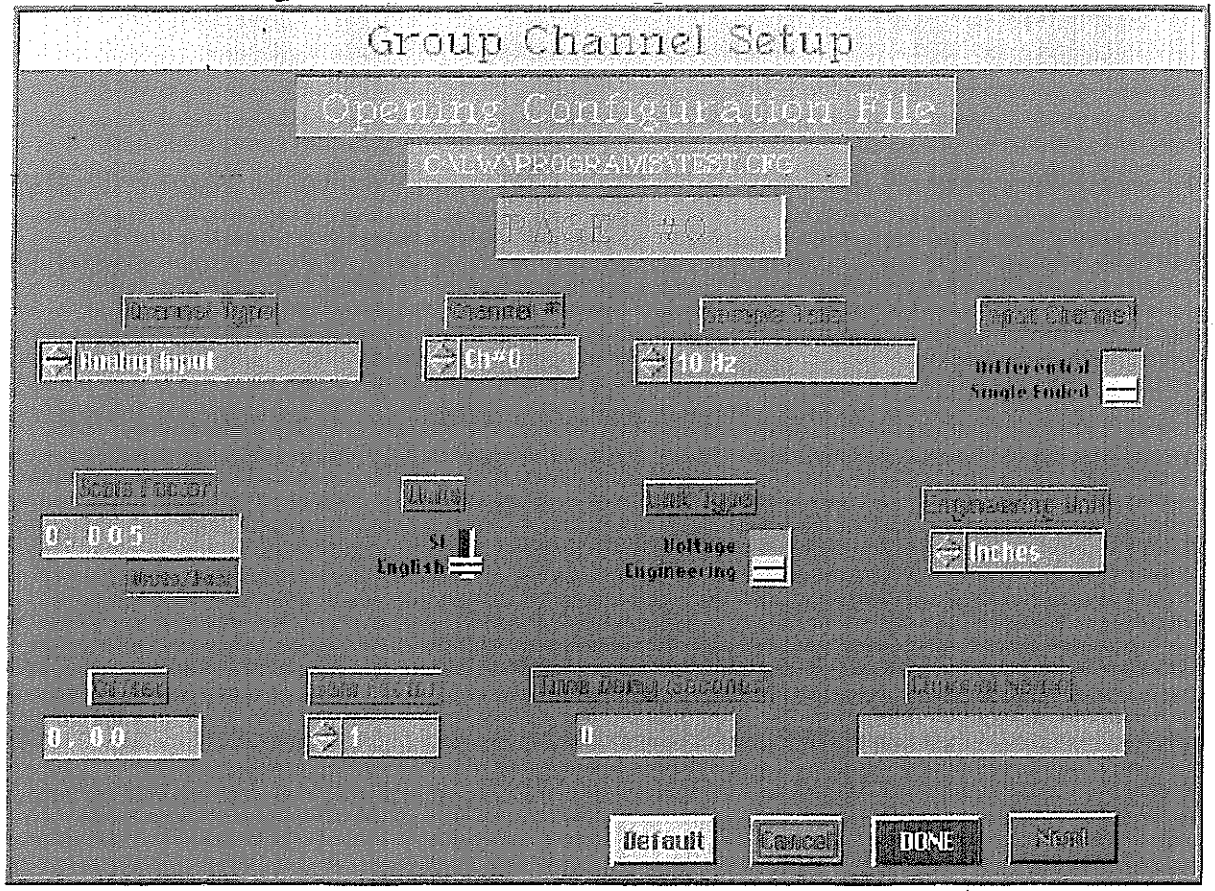


Figure 25. Opening Menu - Setup Pop-up (Group Channel)

Select **group channel** from this pop-up menu to enter into group channel configuration mode. The configuration file name *.CFG is entered in the prompt window presented. The configuration file name can also be picked-up from the listed ones. Press *Select* to confirm the configuration file name entry. *Cancel* returns to command level. Figure 26 presents a group channel configuration setup panel. The selection of different controls can be through the *TAB* key on keyboard or through the pointing device (mouse). The arrow keys can be used to select


Group Channel Setup

Opening Configuration File

C:\LW\PROGRAMS\TEST.CFG

PAGE #0

Channel Type: Analog Input

Channel #: Ch#0

Sample Rate: 10 Hz

Input Channel: Differential / Single Ended

Scale Factor: 0.005

Units: SI / English

Unit Type: Voltage / Engineering

Engineering Unit: Inches

Offset: 0.00

Gain Factor: 1

Time Delay (Seconds): 0

Figure 26. Group Channel Configuration Setup Panel

each value in case of ring and slide switches, or mouse on the control parameters directly. The numeric fields needs to be entered through keyboard. A description of the individual parameters are discussed below:

• *Channel Type:*

There are two channel types on this slide/toggle switch control; Analog and Digital inputs.

• *Channel Number:*

A numeric entry refers to the channel number to which the sensor is assigned.

• *Sampling Rate:*

Sampling rate is the number of samples per second of A/D conversion for the channel selected. It can be selected from the ring switch to one of the following: 1Hz, 2Hz, 5Hz, 10HZ, 20Hz, 50Hz, 100Hz, and 200Hz

• *Input Channel:*

Two input channel types are available; single-ended and differential-ended. Differential type can be used for noise reduction. This can be S or D slide/toggle switch representing single-ended and differential respectively.

• *Scale factor, Offset constant:*

These parameter entries are used with almost all types of input channels. It converts raw data from the interface device into appropriate scientific or engineering units. The Scale Factor and the Offset together converts raw data into scientific or engineering units according to the formula:

$$\text{Converted Data} = ((\text{Raw data} / \text{Channel Gain}) + \text{Offset constant}) \times \text{Scale Factor}$$

Example:

Consider a LVDT sensor among the group with full scale deflection of ±0.005 inches/volt at unity gain, with zero offset, input channel gain of 100, and raw voltage input of 1.0 volts. Then for a full scale input voltage of ±10 volts;

$$\text{Converted Data} = ((1.0/100) + 0) \times 0.005$$
$$= 0.00005 \text{ inches.}$$

Note: In the above example the input voltage range with gain=100 can be ±0.1 volt, i.e. ±0.0005 inches of full scale deflection.

• *Display Unit:*

Select either SI variation or English variation for the display of the units, for the channel selected.

• *Display Type:*

Display type is a slide/toggle switch for Voltage or Engineering unit selection. Voltage selection disables Display Unit control then the input channel will be configured as a voltmeter in ±10 volts range.

• *Input Voltage Gain:*

The input amplifier provides different gain factors of 1,10,100, and 1000 for each input channel. This parameter can be set on this ring switch to raise the level of the input signal to the desired value. For example, a gain of 10 for an input voltage range of 0 to 1 volt yields an output in the range of 0 to 10 volts and thereby increasing the input voltage resolution.

• *Time Delay:*

This parameter defines the time delay in seconds for the start of acquisition for a particular input channel. Use keyboard to enter the numeric value on this control.

• *Sensor Name:*

Provides an identification for the interface channel. It is a string entry with a maximum of 15 characters long.

• *Push button at the bottom left corner of the panel:*

Next: advances sequentially to the next panel referred to as Page #. There are 32 pages which can configure 32 channels as a group.

Done: initializes all the parameter variables, and stores in to a configuration file defined a priori; **.CFG* for future retrieval.

Cancel: terminates the setup operations and returns to panel in Figure 13.

Default: resets the control values on the current page panel the with its initial values.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons if the selection is made through the Tab key.

2.2. File Storage Format (group channel):

Select **File** from the opening menu, Figure 27, which opens the pop-up menu: **Load, Save, Print, Dos, and Quit.**

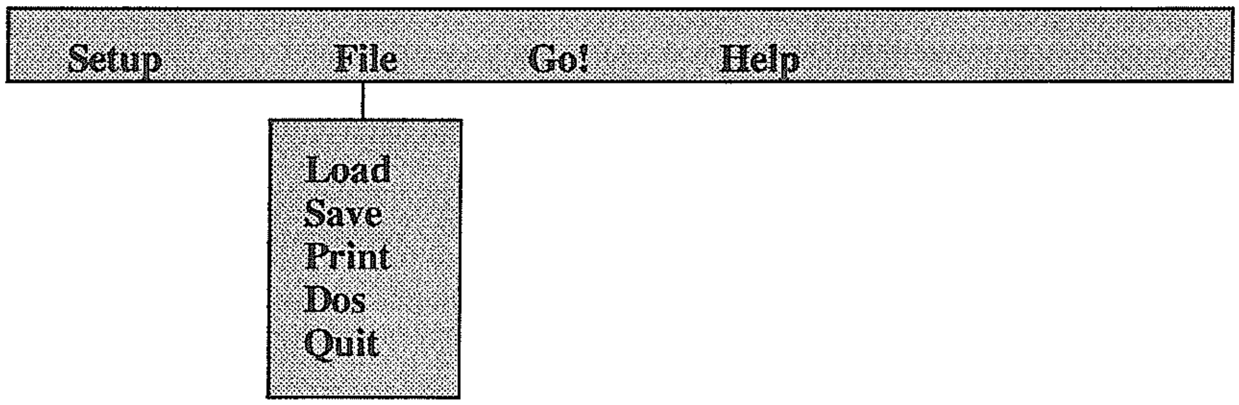
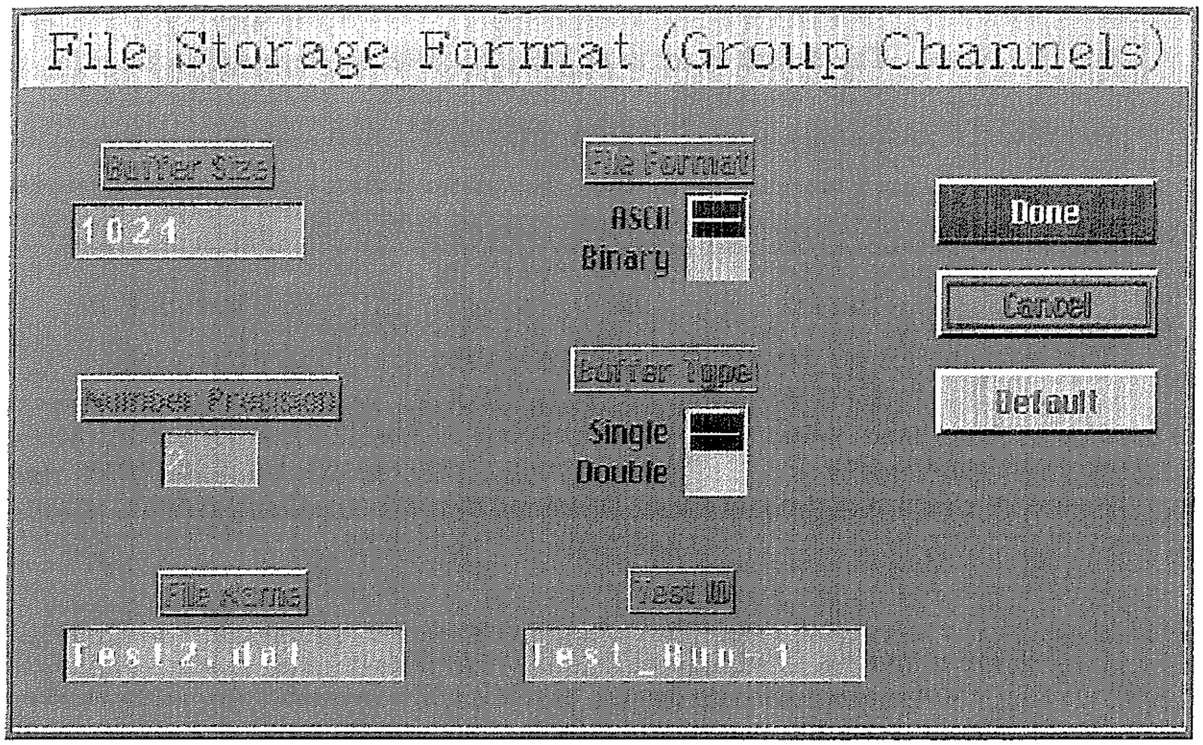


Figure 27. Opening Menu - File Pop-up (Group Channel)

Select **Save** to define the parameters of File Save. Figure 28 presents a group file storage format setup panel. The selection of different controls can be through the *TAB* key on keyboard or through the pointing device (mouse). The arrow keys can be used to select each value in case of ring and slide switches, or mouse on the control parameters directly. The numeric fields needs to be entered through keyboard. A description of the individual parameters are discussed below:

• *File Format:*

The slide/toggle switch selects either ASCII or Binary file format. Binary format can be


File Storage Format (Group Channels)

Buffer Size: 1024

File Format: [✓ ASCII] Binary

Number Precision: 2

Buffer Type: [✓ Single] Double

File Name: Test2.dat

Test ID: Test_Run-1

Figure 28. Group Channel File Format Setup Panel

selected if data compaction is required during storage.

• *Decimal Precision:*

Decimal precision is the number of decimals that is desired for storage. The decimal precision numbers provided on this ring switch are 1,2,3,4,5, and 6.

• *File Name:*

This entry is a file name of 15 characters long. The data is stored on the hard disk under this File Name. The default is set to *prfg01.dat*. Use the keyboard to enter the string variable.

• *Test ID:*

This entry is used for experiment identification and is of 25 characters long.

Figure 29 shows a typical format of the stored data file.

• *Push buttons at the bottom left corner of the panel:*

Done: initializes all the parameter variables, and stores on to a configuration file; *.*CFG* for future retrieval.

Cancel: terminates the setup operations and returns to panel in Figure 13.

Default: resets the control values on the current panel with its initial values.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons.

```
DATE :                         01-19-1993
G R O U P  S E N S O R  R E A D I N G S : for 3 Channels
CHANNEL NAMES :
          Ch #0 :      Voltmeter    Sampled at 0.125 Hz  (Analog Input)
          Ch #1 :      LVDT         Sampling at 0.25 Hz  (Analog Input)
          Ch #2 :      Voltmeter    Sampling at 0.5 Hz   (Analog Input)

P R O C E S S   D A T A
Test Started    Time(h:m:s) : 11:48:27

=====================|| CHANNEL #0 ||=====================
Seconds                   Unit
 Time                     Volts
==========================================================
0.0                     10.0000
8.0                      1.0000

=====================|| CHANNEL #1 ||=====================
Seconds                   Unit
 Time                      In
==========================================================
0.0                     -0.5000
4.0                      0.5000
8.0                     -0.5000

=====================|| CHANNEL #2 ||=====================
Seconds                   Unit
 Time                     Volts
==========================================================
0.0                     10.0000
2.0                      7.0000
4.0                      5.4560
6.0                      2.3455
8.0                      1.8900
==========================================================
Test Started    Time(h:m:s) : 11:48:27
Test Ended      Time(h:m:s) : 11:48:37
==========================================================
Operator Note : NO NOTE
```

Figure 29.   Stored Data File (Group Channel)

2.3. Display Setup (Group Channel):

Select **Setup** from the opening menu Figure 30, which opens the pop-up menu: **Individual Channel, Group Channel, Display: Individual Channel, and Display: Group Channel.**

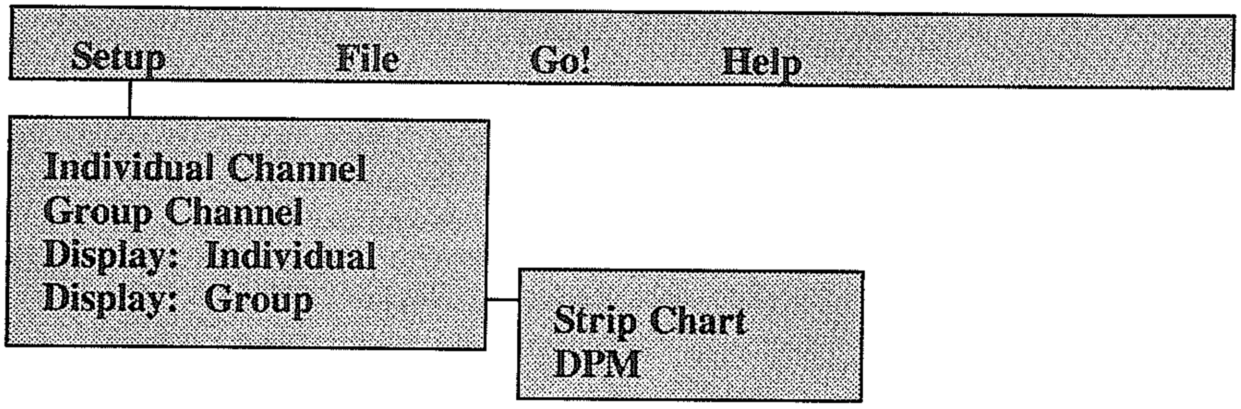


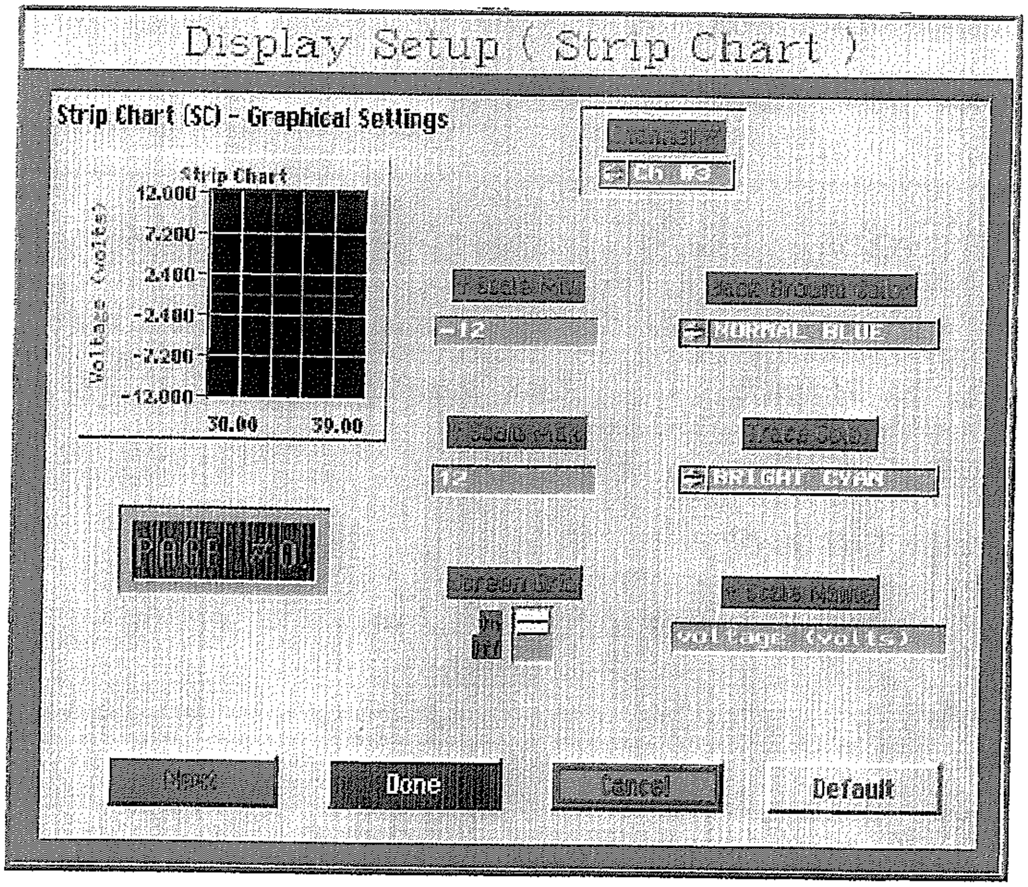
Figure 30. Opening Menu - Setup Pop-up (Group Channel)

Select **Display Group Channel** from this pop-up menu. The next level pop-up menu for selection of strip chart and DPM is offered. The real-time data presentation formats of strip chart and DPM for group channel setup are shown in Figure 31 and Figure 32 respectively. The selection of different controls can be through the *TAB* key on keyboard or through the point device mouse. The arrow keys can be used to select each value in case of ring and slide switches, or mouse on the control parameters directly. The numeric fields needs to be entered through keyboard. A description of the individual parameters are discussed below:

Strip Chart:

- *Channel Number:*

A numeric value selection on this control refers to the channel number which is presented in real-time X-Y graph on one of the four available strip charts.



Ch #3

Y Scale Min: -12

Back Ground Color: NORMAL BLUE

Y Scale Max: 12

Trace Color: BRIGHT CYAN

Y Scale Name: voltage (volts)

Figure 31. Group Channel Display Format Panel (Strip Chart)

• *Y Axis Scale (Minimum, Maximum):*

This parameter control the strip chart's Y-axis scale with the minimum and Maximum value that can be presented. These values can also be changed in real-time during experiment.

• *Background Color:*

The strip chart background color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample strip chart.

• *Trace Color*

The strip chart trace color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample strip chart.

• *Screen Grid On/Off:*

This parameter can be used to toggle the Screen Grid to On or Off.

• *Y Axis Name*

This entry is Y axis name of 25 characters long. The axis name entry is presented for presentation convenience and is set to default *Voltage (Volts)*.

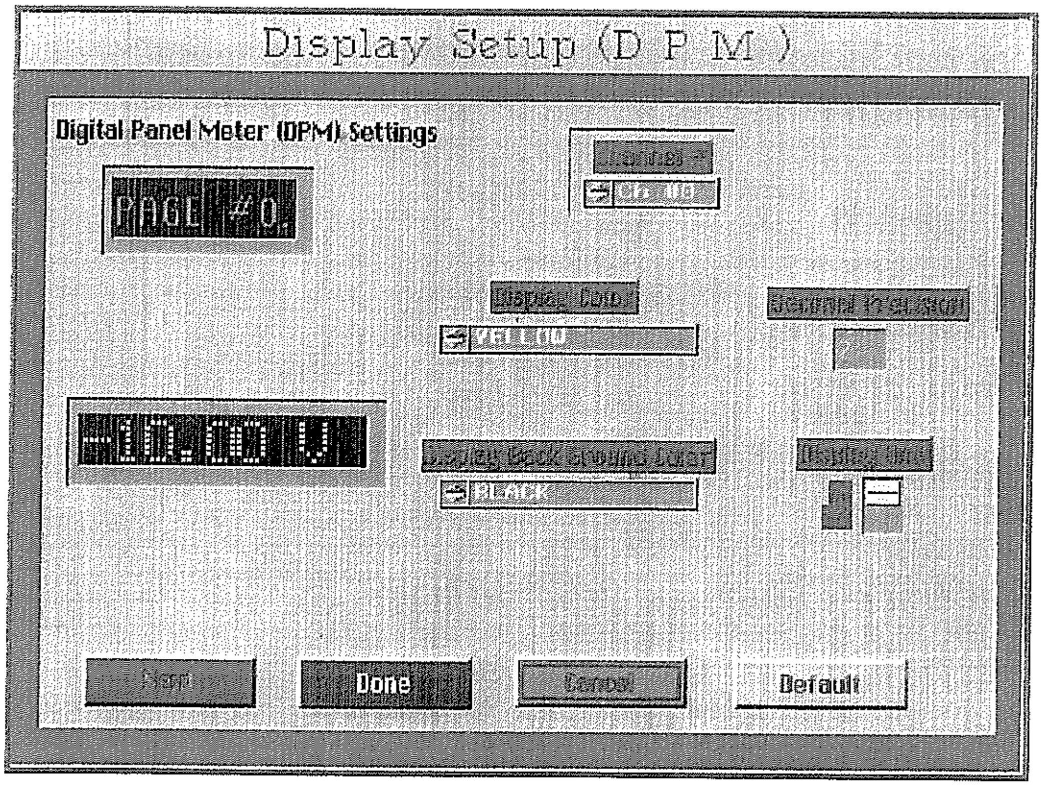
<u>Digital Panel Meter (DPM):</u>

• *Channel Number:*

A numeric value selection on this control refers to the channel number which is presented in digital form on one of the four available digital panel meters.

• *Text Color:*

The DPM text color can be selected from the ring switch to one of the available


Display Setup (D P M)

Digital Panel Meter (DPM) Settings

PAGE #0.

Ch. 00

YELLOW

BLACK

Figure 32. Group Channel Display Format Panel (DPM)

sixteen colors. The setup panel simulates this parameter on the sample DPM.

• *Decimal Digit Precision:*

This numeric value (1,2,3,4,5, and 6) sets the decimal precision on the 7-segment digital display.

• *Background Color:*

The DPM background color can be selected from the ring switch to one of the available sixteen colors. The setup panel simulates this parameter on the sample DPM.

• *Display Unit:*

This can be used to toggle the display of engineering unit On or Off on DPM.

• *Push button at the bottom left corner of the panel:*

Next: rolls over to the next group setting panel referred to as Page #. There are 4 pages which can configure up to 4 channels as a group.

Done: initializes the last channel parameter variables, and stores on to a configuration file defined a priori; **.CFG* for future retrieval.

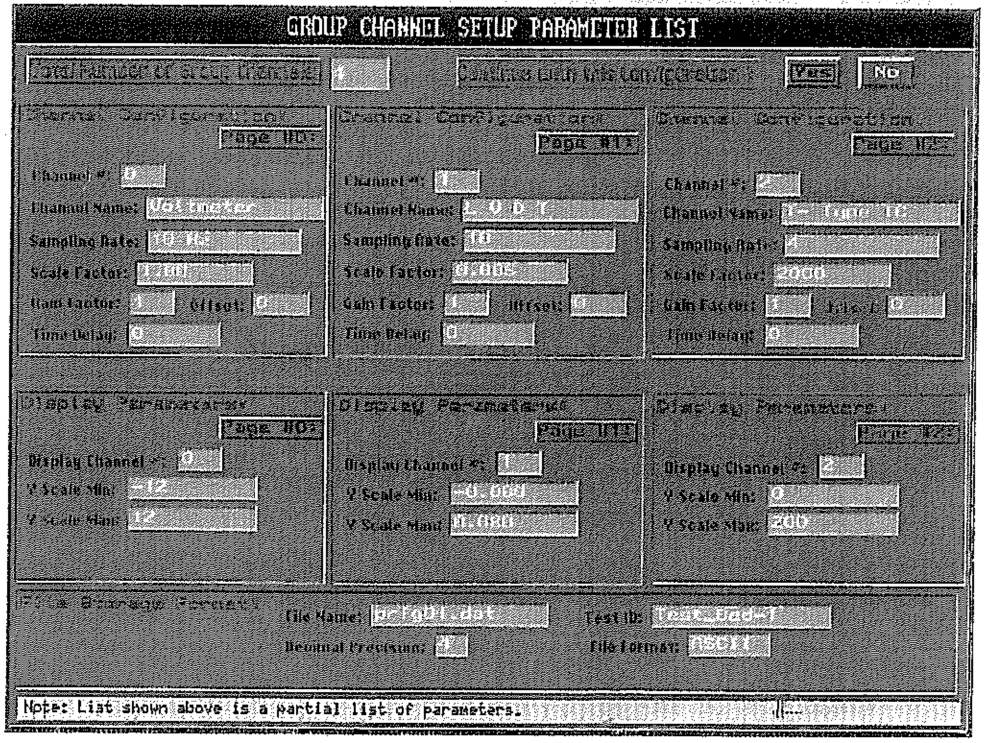
Cancel: terminates the setup operations and returns to panel in Figure 13.

Default: resets the control values on the current page panel with its initial values.

These push buttons can be selected through a Mouse or Tab key on the keyboard. Press Enter to activate these push buttons if the selection is made through the Tab key.

2.4. The *'GO'* Command:

The command *GO!* on opening menu (Figure 13) starts the group channel data acquisition process as configured. The *'GO'* command checks for completeness of the parameters initialized and lists the settings on the screen as shown in Figure 33. It prompts with the useful messages if the setup is incomplete. If no changes are deemed necessary, then press *Yes* to continue. An instrument panel will appear, Figure 34. Press *Power* push button to illuminate the instrument panel which indicates the power-on (ready-to-run) state. Then press



Total Number of Group Channels: [illegible]
Continue with this configuration? Yes No

Channel Configuration — Page #0:
Channel #: 0
Channel Name: Voltmeter
Sampling Rate: 10 Hz
Scale Factor: 1.000
Gain Factor: 1 Offset: 0
Time Delay: 0

Channel Configuration — Page #1:
Channel #: 1
Channel Name: L V D T
Sampling Rate: 10
Scale Factor: 0.005
Gain Factor: 1 Offset: 0
Time Delay: 0

Channel Configuration — Page #2:
Channel #: 2
Channel Name: [illegible]
Sampling Rate: 4
Scale Factor: 2000
Gain Factor: 1 Offset: 0
Time Delay: 0

Display Parameters — Page #0:
Display Channel #: 0
Y Scale Min: -12
Y Scale Max: 12

Display Parameters — Page #1:
Display Channel #: 1
Y Scale Min: -0.080
Y Scale Max: 0.080

Display Parameters — Page #2:
Display Channel #: 2
Y Scale Min: 0
Y Scale Max: 200

File Storage Format:
File Name: brfg01.dat
Test ID: Test_Bed-1
Decimal Precision: 4
File Format: ASCII

Note: List shown above is a partial list of parameters.

Figure 33. Group Channel Parameter Check List

Run push button to start the data acquisition, storage, and presentation processes. An active instrument panel is shown in Figure 34.

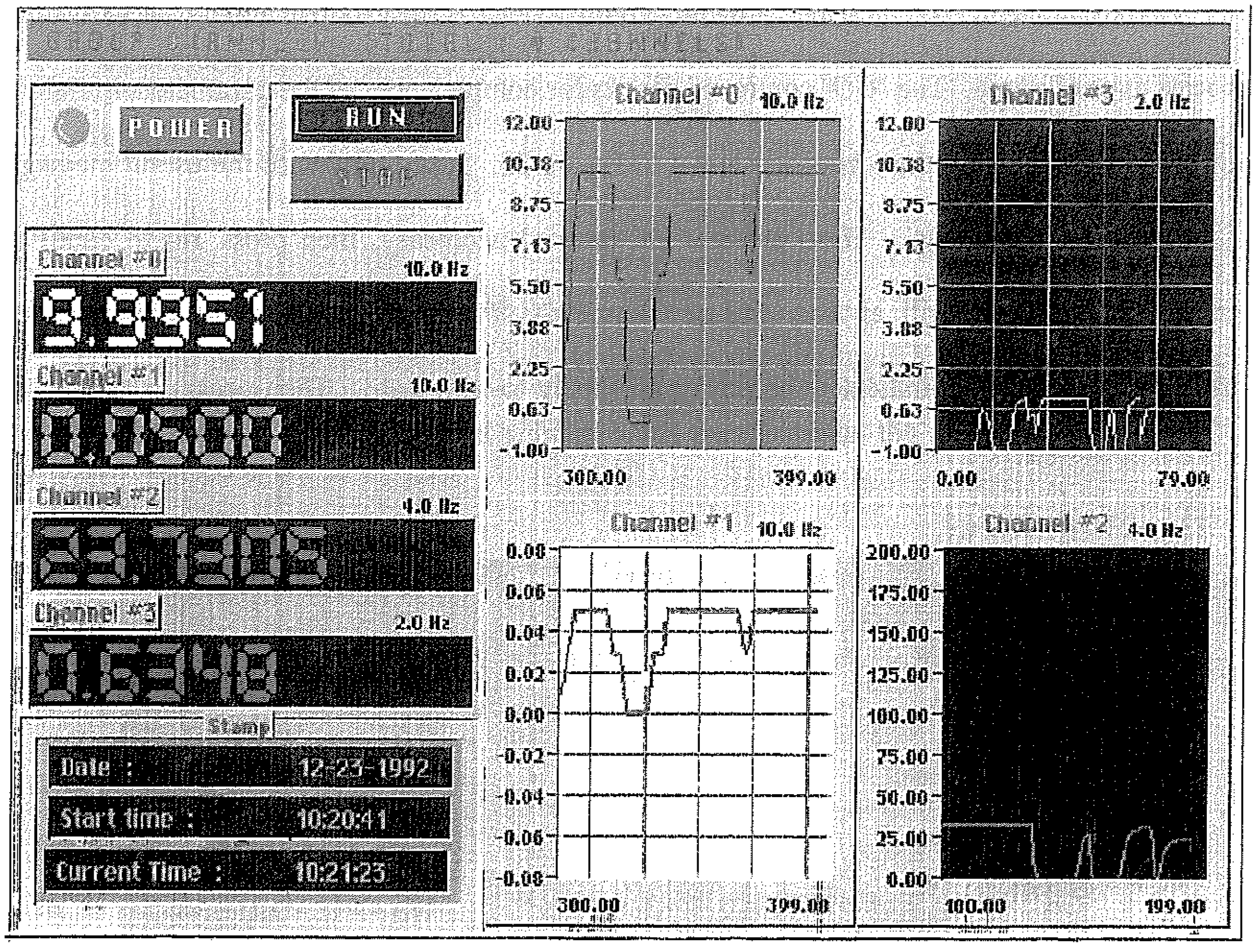


Figure 34. Group Channel Instrument Panel

The acquisition of data can be stopped by pressing *Stop* push button. On termination of the experiment using *File - Quit* an operator note window will appear. It accepts 256 character note which is appended to the data file at the end.

APPENDIX B

SOFTWARE LISTING

```
/*
                              A L D I S   VER 1.1

            Accelerated Loading Device Instrumentation Software


                    Designed by: Srinivas K. Kathavate
                  Supervised by: Dr. Louay N. Mohammad

*/
#include "ald.h"
#include "def.h"
#include "help.h"
#include "setup.h"
#include "indiv.h"
#include "group.h"
#include "dapio.h"

/*
            ALDIS Main Program - Logo Screen - Function Selection -

*/
main()
{
/* ------------------- ALDIS 1.1 Start ------------------*/
   AldHandle=LoadPanel("Ald.uir",Logo);SetMouseCursor(7);
   if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
   DisplayPanel(AldHandle);i=0;while(!KeyHit() && i<6){Delay(1);i++;}UnloadPanel(-1);quit=0;
   while(!quit){
      AldHandle=LoadPanel("Ald.uir",OpenAld);SetMouseCursor(0);
      if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
      DisplayPanel(AldHandle);SetActivePanel(AldHandle);
      GetUserEvent(TRUE,&handle,&id);
         if(id==OpenAld_Select){
         GetListItemIndex(AldHandle,OpenAld_Selection,&ndx);
            switch(ndx){
               case 1 :
                  DatacqNstore();break;
               case 3 :
                  UnloadPanel(-1);AldHandle=LoadPanel("Ald.uir",Ald2);SetMouseCursor(0);
```

```
                if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                DisplayPanel(AldHandle);
                MessagePopup("Sorry, Data Analysis not available at this time");break;
             case 5 :
                UnloadPanel(-1);AldHandle=LoadPanel("Ald.uir",Ald3);SetMouseCursor(0);
                if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                DisplayPanel(AldHandle);
                MessagePopup("Sorry, Function Generation not available now");break;
             case 7 :
                UnloadPanel(-1);AldHandle=LoadPanel("Ald.uir",Ald4);SetMouseCursor(0);
                if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                DisplayPanel(AldHandle);
                MessagePopup("Sorry, Process Control not available at this time");break;
             case 9 :
                UnloadPanel(-1);AldHandle=LoadPanel("Ald.uir",Ald5);SetMouseCursor(0);
                if(AldHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                DisplayPanel(AldHandle);
                MessagePopup("Sorry, Ald Info System not available at this time");break;
             default : break;}}
       if(id==OpenAld_Quit){chosen=ConfirmPopup("Are you sure ?");if(chosen)quit=TRUE;}}
}

/*
```

Data Acquisition and Storage Routine

This routine opens the Menu to data acquisition for channel Configuration Setup, File format definition, and display formats. It provides context sensitive help at every stage. File Menu offers functions like Load a config file, Save for file format definition, Print for screen dump, and DOS for prompt under TSR.

Input : None
Output : Defines for both Individual and Group channels, the channel configuration, file format, and display

```
*/

void DatacqNstore()
{
   UnloadPanel(-1);Dap_Init();OpenHandle=LoadPanel("Ald.uir",Ald1);Dap1File="INDIV";
   if(OpenHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
   DisplayPanel(OpenHandle);OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu);getout=TRUE;
   while(getout){SetInputMode(OpenMenuBar,OpenMenu_Help_Control,0);
      GetUserEvent(TRUE,&OpenHandle,&id);
      switch(id){
         case OpenMenu_File_Print :
            chosen=ConfirmPopup("Screen Print selected, Continue Printing");
            if(chosen==TRUE)OutputScreen(0, "");
            break;
         case OpenMenu_File_Quit :
            getout=FALSE;UnloadPanel(-1);
      count=GetFileInfo(IConfig,&FLength);if(count)CloseFile(InfoChHndl[ChNo]);
      count=GetFileInfo("Group.cfg",&FLength);if(count)CloseFile(InfoGrpChHndl);
      break;
   case OpenMenu_Setup_ICh :
      if(!ReLoad){RestoreFlags();SetupIndCh();}
      else {chosen=ConfirmPopup("ConfigFile loaded, Reload ?");
         if(chosen){RestoreFlags();SetupIndCh();}}
      break;
   case OpenMenu_Setup_GCh :
      if(!ReLoad){RestoreFlags();SetupGrpCh();}
      else {chosen=ConfirmPopup("ConfigFile loaded, Reload ?");
         if(chosen){RestoreFlags();SetupGrpCh();}}
      break;
   case OpenMenu_Setup_SCh :
      UnloadPanel(-1);SetupSimulCh();
      break;
   case OpenMenu_Setup_IndDisp :
      if(!ReLoad){
         if(IFlag)SetupIndDisplay();
         else MessagePopup("Run channel configuration first");}
      else MessagePopup("ConfigFile already loaded");
      break;
   case OpenMenu_Setup_GrpDisp :
      if(!ReLoad){
         if(GrpFlag)SetupGrpDisplay();
         else MessagePopup("Run channel configuration first");}
      else MessagePopup("ConfigFile already loaded");
      break;
   case OpenMenu_File_Save :
      if(!ReLoad){
         if(IFlag || GrpFlag)FileSaveSetup();
         else MessagePopup("Run channel configuration first");}
      else MessagePopup("ConfigFile already loaded");
      break;
   case OpenMenu_File_Load :
      if(!ReLoad)ReadCfgFile();
      else {chosen=ConfirmPopup("ConfigFile loaded, Reload ?");
         if(chosen){RestoreFlags();ReadCfgFile();}}
      break;
   case OpenMenu_File_Dos :
      break;
   case OpenMenu_Go :
      iii=TRUE;if(IFlag && GrpFlag) IFlag=0;
      while(iii){
         if(!IFlag && !GrpFlag){
            SaveHandle1=LoadPanel("help.uir", Help2a);DisplayPanel(SaveHandle1);
            MessagePopup("Configuration file empty, run Setup");UnloadPanel(-1);
            OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);iii=FALSE;}
         if(IFlag){
            if(FIFlag){
               if(DIFlag){
                  SetFilePtr(InfoChHndl[ChNo],0,0);Fmt(GrpChString,"%s<0,0,0,");
                  bytes=5;WriteFile(InfoChHndl[ChNo],GrpChString,bytes);
```

```
                    CloseFile(InfoChHndl[ChNo]);UnloadPanel(-1);
                    if(!ReLoad)ConfirmIndParam();
                    else {IndivSensor();iii=FALSE;}}
                 else{chosen=ConfirmPopup("Display Setup notdone, assume default ?");
                    if(chosen){SetDispParam();DIFlag=TRUE;}
                    else iii=FALSE;}}
              else{MessagePopup("Storage Format undefined, use Save under File Menu");iii=FALSE;}}
           if(GrpFlag){
              if(FGFlag){
                 if(DGSCFlag){
                    if(DGDPMFlag){
                       SetFilePtr(InfoGrpChHndl,0,0);
                                    Fmt(GrpChString,"%s<%i,%i,%i,",
                                                               GrpCount,SCCount,DPMCount);
                       bytes=6;if(GrpCount>=10)bytes=7;WriteFile(InfoGrpChHndl,GrpChString,bytes);
                       CloseFile(InfoGrpChHndl);UnloadPanel(-1);GroupSensors();iii=FALSE;}
                    else{chosen=ConfirmPopup("Display Setup - DPM not done, assume No DPM ?");
                       if(chosen)DGDPMFlag=TRUE;else iii=FALSE;}}
                 else{chosen=ConfirmPopup("Display Setup - Strip Chart not done, assume No SC ?");
                       if(chosen)DGSCFlag=TRUE;else iii=FALSE;iii=FALSE;}}
              else{MessagePopup("Storage Format undefined, use Save under File Menu");iii=FALSE;}}}
        break;
     case OpenMenu_Help_Panel :
        HelpHandle=LoadPanel("help.uir",Help1);DisplayPanel(HelpHandle);
        OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu1);
        GetUserEvent(TRUE,&handle,&id);
        UnloadPanel(HelpHandle);OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu);
        OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);
        break;
     default :break;}}
  Dap_Close();UnloadMenuBar();
}


/*
```

Data Acquisition and Storage Routine (Individual Channel)

This routine displays the instrument panel and writes the DAP OS Task routine for single channel, which will be loaded onto the DAP board. It waits for user input and starts the test. It offers functions to change Y scale factor of the Graph dynamically, and Zoom function which can be used to view the waveform in ≈ 3X magnification.
Input : Individual channel Configuration Setup, and all parameters loaded

```
*/
void IndivSensor()
{
  UnloadPanel(-1);UnloadMenuBar();MenuBar=LoadMenuBar("Ald.uir",AldMenu);
  IndivHandle=LoadPanel("Indiv.uir",AldIndiv);UnitTime=0.0;
  if(IndivHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
  DisplayPanel(IndivHandle);SetActivePanel(IndivHandle);
/*-------------- Display Setup start --------------*/
```

```
   DPMColorFn();DPMBColorFn();SCBColorFn();SCTColorFn();
   error=CompareStrings(YAxisName,0,NullString,0,0);
   if(error)SetGraphAttribute(IndivHandle,AldIndiv_StripChart,18,YAxisName);
   GetGraphAttribute(IndivHandle,AldIndiv_StripChart,5,&BColor);
   if(!Grid)SetGraphAttribute(IndivHandle,AldIndiv_StripChart,3,BColor);
   ConfigureAxes(IndivHandle,AldIndiv_StripChart,-1,0,0,0,YScaleMin,YScaleMax);
   SetCtrlVal(IndivHandle,AldIndiv_YScaleMin,YScaleMin);SetCtrlVal(IndivHandle,AldIndiv_YScaleMax,YScaleMax);
/*-------------- File storage routine -------------*/
   Fmt(LabelString,"%s<   S E N S O R   S E L E C T E D   :   %s(C H # %i)",ChName,ChNo);
   SetCtrlVal(IndivHandle,AldIndiv_PanelLabel,LabelString);ii=TRUE;
   while(ii){GetUserEvent(TRUE,&handle,&id);
      if(id==AldMenu_File_Quit){ii=FALSE;iii=FALSE;}
      if(id==AldIndiv_Power){PowerOnStat();ii=FALSE;power=TRUE;iii=TRUE;}}
   if(!Format)SaveFileHndl=OpenFile(FileName,0,0,0);else SaveFileHndl=OpenFile(FileName,0,0,1);
      FmtFile(SaveFileHndl,"%s<Date : \t\t\t\t%s\n\n",daystring);
          Fmt(&SamCopy,"%f<%i",Sampling);ActualSampling=1000/SamCopy;
      if(ActualSampling>=0){
             Fmt(&Frequency,"%i<%f",ActualSampling);Fmt(SampleString,"%s<%i  Hz",Frequency);}
      else Fmt(SampleString,"%s<%f",ActualSampling);Tick=1/ActualSampling;
      FmtFile(SaveFileHndl,"%s<Simple  %s Test at frequency : %s\n",ChName,SampleString);
      FmtFile(SaveFileHndl,"%s<Time Interval(1/Frequency) = %f Sec\n",Tick);
      FmtFile(SaveFileHndl,"%s<Channel Name = : %s\n",ChName);
          FmtFile(SaveFileHndl,"%s<Test ID : %s\n",Specimen);
      FmtFile(SaveFileHndl,"%s<Channel Type : %s\n\n",IndChType);
          FmtFile(SaveFileHndl,"%s<P r o c e s s  D a t a\n\n");
          FmtFile(SaveFileHndl,
             "%s<==========================================================================\n");
      FmtFile(SaveFileHndl,"%s< Time\t   Engineering Unit \n");
      FmtFile(SaveFileHndl,"%s<Seconds\t     %s \n",UnitName);
      FmtFile(SaveFileHndl,
             "%s<==========================================================================\n");
   SetInputMode(MenuBar,AldMenu_Zoom_Full,0);
   while(iii){GetUserEvent(TRUE,&handle,&id);
      switch(id){
         case AldMenu_File_Print :
            chosen=ConfirmPopup("Screen Print selected, Continue Printing");
            if(chosen==TRUE)OutputScreen(0, "");
            break;
         case AldMenu_File_Quit :
            iii=FALSE;
            break;
         case AldIndiv_Power :
            if(power){PowerOffStat();power=FALSE;SetInputMode(MenuBar,AldMenu_File_Quit,1);}
            else{PowerOnStat();power=TRUE;}break;
         case AldIndiv_Run :
            SetInputMode(MenuBar,AldMenu_Help,0);SetInputMode(MenuBar,AldMenu_Zoom_Full,1);
            SetInputMode(MenuBar,AldMenu_File_Print,0);SetInputMode(MenuBar,AldMenu_File_Quit,0);
            if(TDelay!=0){DelayHandle=LoadPanel("indiv.uir",TD);
               chosen=ConfirmPopup("Time Delay set, want to ignore Time Delay ?");i=1;
               if(!chosen){DisplayPanel(DelayHandle);
                  while(i<=TDelay){delay(1);SetCtrlVal(DelayHandle,TD_Second,i);i++;}
                  UnloadPanel(DelayHandle);DisplayPanel(IndivHandle);}}
            Fmt(StrtTimeString,timestr());SetCtrlVal(IndivHandle,AldIndiv_ActualSTime,StrtTimeString);
```

```
  FmtFile(SaveFileHndl,"%s<\nTest Started \t Time(h:m:s) : %s\n",StrtTimeString);
  FmtFile(SaveFileHndl,
          "%s<=====================================================================\n");
  switch(Sampling){
    case 50 : BufferLen=4;break;  /* for Freq = 20 Hz */
    case 20 : BufferLen=5;break;  /* for Freq = 50 Hz */
    case 10 : BufferLen=25;break;  /* for Freq = 100 Hz */
    case 5 : BufferLen=25;break;   /* for Freq = 200 Hz */
    case 2 : BufferLen=25;break;   /* for Freq = 500 Hz */
    default : BufferLen=1;break;}
  SetCtrlVal(IndivHandle,AldIndiv_Sample, SampleString);
/*---------  "Write Dap File" is the routine for writing on to disk the DAP OS task routines ----------*/
  error=WriteIndDapFile(Dap1File,ChSD,ChNo,IndGain,Sampling);
  if(error==-1){MessagePopup("Write Fault in 'WriteDapFile' Function"); return;}
/*--------  "Dap Config" loads the previously disk written *.DAP file on to on-board DAP --------*/
  Dap_Config(DapFName);check=TRUE;FirstRun=TRUE;
  Dap_FinFlush(F);
  while(check){Fmt(CurTimeString,timestr());SetCtrlVal(IndivHandle,AldIndiv_ActualETime,CurTimeString);
    if(!StopFlag)GetUserEvent(FALSE,&handle,&id);else id=-1;
    switch(id){
      case AldMenu_Zoom_Full :
        ZoomHandle=LoadPanel("indiv.uir",AldZoom);
        if(ZoomHandle<0){FmtOut("Unable to load required panel from the resource file.\n");return;}
        zoom=TRUE;HidePanel(IndivHandle);DisplayPanel(ZoomHandle);
                        SetActivePanel(ZoomHandle);DPMColorFn();
        DPMColorFn();DPMBColorFn();SCBColorFn();SCTColorFn();
        SetGraphAttribute(ZoomHandle,AldZoom_StripChart,18,YAxisName);
        SetCtrlVal(ZoomHandle,AldZoom_YScaleMin,YScaleMin);
        if(!Grid)SetGraphAttribute(ZoomHandle,AldZoom_StripChart,3,BColor);
        SetCtrlVal(ZoomHandle,AldZoom_YScaleMax,YScaleMax);
                        MenuBar=LoadMenuBar("Ald.uir",AldMenu1);
        ConfigureAxes(ZoomHandle,AldZoom_StripChart,-1,0,0,0,YScaleMin,YScaleMax);
        break;
      case AldMenu_File_Quit :
        UnloadMenuBar();UnloadPanel(ZoomHandle);MenuBar=LoadMenuBar("Ald.uir",AldMenu);
        SetInputMode(MenuBar,AldMenu_Help,0);SetInputMode(MenuBar,AldMenu_File_Quit,0);
        DisplayPanel(IndivHandle);zoom=FALSE;iii=FALSE;
        break;
      case AldIndiv_Stop :
        if(FirstRun){FmtFile(DapOut,"%s<STOP  %s,%s1\n",Dap1File,Dap1File);
          StopFlag=TRUE;FirstRun=FALSE;}
        if(GetDapAvail(DapIn) < 0){
          check=FALSE;StopFlag=FALSE;FirstRun=TRUE;iii=TRUE;}
        SetInputMode(MenuBar,AldMenu_Zoom_Full,0);
        break;
      case AldIndiv_YScaleMin :
      case AldIndiv_YScaleMax :
      case AldZoom_YScaleMin :
      case AldZoom_YScaleMax :
        if(!zoom){GetCtrlVal(IndivHandle,AldIndiv_YScaleMin,&YScaleMin);
          GetCtrlVal(IndivHandle,AldIndiv_YScaleMax,&YScaleMax);
          ConfigureAxes(IndivHandle,AldIndiv_StripChart,-1,0,0,0,YScaleMin,YScaleMax);}
        if(zoom){GetCtrlVal(ZoomHandle,AldZoom_YScaleMin,&YScaleMin);
```

```
                GetCtrlVal(ZoomHandle,AldZoom_YScaleMax,&YScaleMax);
                ConfigureAxes(ZoomHandle,AldZoom_StripChart,-1,0,0,0,YScaleMin,YScaleMax);}
              break;
            default :break;}
            if(check)Dap_Read(BufferLen,data);
            if(Unit1){if(IndDispType){UnitVal=(1.0*10.0);Fmt(UnitName,"%s<  V");}
            else{UnitVal=(Scale*10.0);}}
            if(!Unit1){if(IndDispType){UnitVal=(1.0*10.0);Fmt(UnitName,"%s<  V");}
            else{UnitVal=(Scale*10.0);}}
            Volt[0]=(data[0]*10.0)/32768.0;EnggUnit[0]=((Volt[0]+Offset)*UnitVal)/10.00;
            if(DispUnit)Fmt(EnggString,"%s<%f[p4]  %s",EnggUnit[0],UnitName);
            else Fmt(EnggString,"%s<%f[p4]",EnggUnit[0]);
            Fmt(VoltString,"%s<%f[p4]  %s",Volt[0]," v");
            if(zoom){PlotStripChart(ZoomHandle,AldZoom_StripChart,EnggUnit,1,0,0,4);
              SetCtrlVal(ZoomHandle,AldZoom_Digital,EnggString);}
            if(!zoom){PlotStripChart(IndivHandle,AldIndiv_StripChart,EnggUnit,1,0,0,4);
              SetCtrlVal(IndivHandle,AldIndiv_Digital,EnggString);
              SetCtrlVal(IndivHandle,AldIndiv_Volt,VoltString);}
            for(i=0;i<BufferLen;i++){
              Fmt(UnitTimeString,"%s<%f",UnitTime);
              Volt[i]=(data[i]*10.0)/32768.0;
              EnggUnit[i]=((Volt[i]+Offset)*UnitVal)/10.00;
              Fmt(EnggString,"%s<%f[p4]",EnggUnit[i]);
              FmtFile(SaveFileHndl,"%s<%s\t\t%s\n",UnitTimeString,EnggString);
              UnitTime+=Tick;}}
        break;
      default :
        break;}}
  FmtFile(SaveFileHndl,"%s<________________________________________\n");
  FmtFile(SaveFileHndl,"%s<Test Started \t Time(h:m:s) : %s\n",StrtTimeString);
  FmtFile(SaveFileHndl,"%s<Test Ended \t Time(h:m:s) : %s\n",CurTimeString);
  UnloadPanel(-1);NoteHandle=LoadPanel("setup.uir", Note);DisplayPanel(NoteHandle);
  GetUserEvent(TRUE,&NoteHandle,&id);
  switch(id){
    case AldMenu_File_Print :
      chosen=ConfirmPopup("Screen Print selected, Continue Printing");
      if(chosen){ConfigurePrinter("LPT1", 1, 8.0, 10.0,1);OutputScreen(0, "");}break;
    case Note_Done :
      GetCtrlVal(NoteHandle, Note_NoteText, NoteText);error=CompareStrings(NoteText,0,NullString,0,0);
      if(!error)Fmt(NoteText,"%s<NO NOTE");
          FmtFile(SaveFileHndl,"%s<________________________________________\n");
      FmtFile(SaveFileHndl,"%s<Operator Note : %s\n",NoteText);break;
    default :break;}
  CloseFile(SaveFileHndl);
  RestoreFlags();UnloadMenuBar();OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu);
  FmtFile(DapOut,"ERASE  %s,%s1\n",Dap1File,Dap1File);PowerOffStat();
  UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1); DisplayPanel(OpenHandle);
}
```

```
/*
```

Data Acquisition and Storage Routine (Group Channels)

This routine displays the instrument panel for 8 channels and writes the DAP OS Task routine for all the channels, which will be loaded onto the DAP board. It waits for user input and starts the test. Analog input Channels can be up to 512 in number with different parameter settings. Stores data onto a file specified in file format and stamps date, test time start, and end time.

Input : Group channel configuration setup, and all parameters loaded
Output : Acquires data and stores according to the format specified.

```
*/

void    GroupSensors()
{
  UnloadPanel(-1);GroupHandle=LoadPanel("group.uir",AldGroup);
  if(GroupHandle<0){FmtOut("Unable to load required panel from the resource file.\n");return;}
  UnloadMenuBar();MenuBar=LoadMenuBar("Ald.uir",AldMenu);DisplayPanel(GroupHandle);ii=TRUE;iii=TRUE;
  Fmt(LabelString,"%s< G R O U P  C H A N N E L S   (T O T A L  :  %i  C H A N N E L S)",GrpCount);
  SetCtrlVal(GroupHandle,AldGroup_Label,LabelString);Fmt(daystring,datestr());
  if(Format)SaveFileHndl=OpenFile(GrpFName,0,0,1);else SaveFileHndl=OpenFile(GrpFName,0,0,0);
  FmtFile(SaveFileHndl,"%s<DATE : \t\t\t\t%s\n\n", daystring);
  for(i=0;i<GrpCount;i++){
    SetPrefix=GrpPageNo[i];Fmt(GTmp,"%s<%i",SetPrefix);
    if(Format)TmpHndl[SetPrefix]=OpenFile(GTmp,0,0,1);
    if(!Format)TmpHndl[SetPrefix]=OpenFile(GTmp,0,0,0);
    GTick[SetPrefix]=1/GrpSample[SetPrefix];
    switch(GrpChType[i]){
      case 0 : Fmt(GrpType,"%s<Analog Input");break;
      case 1 : Fmt(GrpType,"%s<Analog Output");break;
      case 2 : Fmt(GrpType,"%s<Digital Input");break;
      case 3 : Fmt(GrpType,"%s<Digital Output");break;
      default : break;}
    FmtFile(SaveFileHndl,"%s<\t\tCh #%i :  \tSampling at %f\n\t\t\t\t\t%s\n",i,GrpSample[i],GrpType);
    FmtFile(TmpHndl[SetPrefix],
        "%s<\n\n══════════════╡ CHANNEL #%i ╞══════════════\n",GrpPageNo[i]);
    switch(GrpUName[i]){
      case 0 : Fmt(GrpDUnit,"%s<  'C");break;
      case 1 : Fmt(GrpDUnit,"%s<  'F");break;
      case 2 : Fmt(GrpDUnit,"%s<  In");break;
      case 3 : Fmt(GrpDUnit,"%s<  MM");break;
      case 4 : Fmt(GrpDUnit,"%s<  Ms");break;
      default : break;}
    FmtFile(TmpHndl[SetPrefix],"%s<Seconds\t\t  %s\n",GrpDUnit);
    FmtFile(TmpHndl[SetPrefix],"%s<  Time\t\t\n");
    FmtFile(TmpHndl[SetPrefix],
        "%s<════════════════════════════════════════════════════════\n");}
  FmtFile(SaveFileHndl,"%s<G R O U P  S E N S O R  R E A D I N G S : for %i Channels\n",GrpCount);
  FmtFile(SaveFileHndl,"%s<CHANNEL  NAMES :\n");
FmtFile(SaveFileHndl,"%s<\nP R O C E S S  D A T A\n");
while(ii){GetUserEvent(TRUE,&handle,&id);
   if(id==OpenMenu_File_Print){chosen=ConfirmPopup("Screen Print selected, Continue Printing");
      if(chosen==TRUE)OutputScreen(0, "");}
   if(id==AldMenu_File_Quit){ii=FALSE;iii=FALSE;}
   if(id==AldGroup_Power){GrpPowerOn(SCCount,DPMCount);ii=FALSE;power=TRUE;
      for(i=0;i<SCCount;i++){
         SetSCPrefix=SC_ChNo[i];
         SetGraphAttribute(GroupHandle,OSCCONST+i,5,GSCB_Color[SetSCPrefix]);
         if(!GGrid[SetSCPrefix])SetGraphAttribute(GroupHandle,OSCCONST+i,3,GSCB_Color[SetSCPrefix]);
         SetTraceAttribute(GroupHandle,OSCCONST+i,1,0,GSCT_Color[SetSCPrefix]);
         Fmt(AldSCTitle,"%s<Channel #%i",SetSCPrefix);
         SetGraphAttribute(GroupHandle,OSCCONST+i,0,AldSCTitle);
         ConfigureAxes(GroupHandle,OSCCONST+i,-1,0,0,0,
                  GYScaleMin[SetSCPrefix],GYScaleMax[SetSCPrefix]);
         Fmt(AldSCTitle,"%s<%f Hz",GrpSample[SetSCPrefix]);
         SetCtrlVal(GroupHandle,SCFCONST+i,AldSCTitle);}
      for(i=0;i<DPMCount;i++){
         SetDPMPrefix=DPM_ChNo[i];
         SetCtrlAttribute(GroupHandle,DPMCONST+i,4,GDPM_Color[SetDPMPrefix]);
         SetCtrlAttribute(GroupHandle,DPMCONST+i,6,GDPMB_Color[SetDPMPrefix]);
         Fmt(AldSCTitle,"%s<Channel #%i",SetDPMPrefix);
         SetCtrlAttribute(GroupHandle,DPMCONST+i,0,AldSCTitle);
         Fmt(AldSCTitle,"%s<%f Hz",GrpSample[SetDPMPrefix]);
         SetCtrlVal(GroupHandle,DPMFCONST+i,AldSCTitle);}}}
while(iii){GetUserEvent(TRUE,&handle,&id);
   switch(id){
      case AldGroup_Power :
         if(power){GrpPowerOff();power=FALSE;SetInputMode(MenuBar,AldMenu_File_Quit,1);}
         else{GrpPowerOn(SCCount,DPMCount);power=TRUE;}
         break;
      case AldMenu_File_Print :
         chosen=ConfirmPopup("Screen Print selected, Continue Printing");
         if(chosen==TRUE)OutputScreen(0, "");
         break;
      case AldMenu_File_Quit :
         iii=FALSE;break;
      case AldGroup_Run :
         SetInputMode(MenuBar,AldMenu_Help,0);SetInputMode(MenuBar,AldMenu_File_Quit,0);
         SetInputMode(MenuBar,AldMenu_Zoom,0);
         check=TRUE;FirstRun=TRUE;StopFlag=0;Fmt(StrtTimeString,timestr());
         SetCtrlVal(GroupHandle,AldGroup_ActualSTime,StrtTimeString);
         FmtFile(SaveFileHndl,"%s<Test Started \t Time(h:m:s) : %s\n",StrtTimeString);
         error=WriteGrpDapFile();
            if(error==-1){MessagePopup("Write Fault in 'WriteGrpDapFile' Function"); return;}
         Dap_Config("Group.dap");Dap_FinFlush(F);
         while(check){Fmt(CurTimeString,timestr());
                  SetCtrlVal(GroupHandle,AldGroup_ActualETime,CurTimeString);
            if(!StopFlag)GetUserEvent(FALSE,&handle,&id);
            switch(id){
               case AldGroup_Stop :
                  if(FirstRun){FmtFile(DapOut,"%s<STOP GRP,GRP_A\n");
                           NoSCDisp=TRUE;NoDPMDisp=TRUE;
```

```
                StopFlag=TRUE;FirstRun=FALSE;}
            if(GetDapAvail(DapIn) < 0){UnloadPanel(MessHandle);
              check=FALSE;StopFlag=0;NoDPMDisp=FALSE;NoSCDisp=FALSE;}
            break;
          case AldMenu_File_Quit :
            GrpPowerOff();check=FALSE;
            break;
          default :
            break;}
        /* TSampling variations here */
        Dap_Read(TSampling,data);
        for(i=0;i<TSampling;i+=2){
          Scan(&data[i],"%f>%i",&Prefix);Scan(&data[i+1],"%f>%f",&T_data);
          if(GrpUnitType[Prefix])UnitVal=(10.00);else UnitVal=(GrpScale[Prefix]*10.00);
          Volt[0]=(T_data*10.0)/32768.0;EggUnit[0]=((Volt[0]+GrpOffset[0])*UnitVal)/10.00;
          Fmt(EnggString,"%s<%f[p4]",EnggUnit[0]);Fmt(UnitTimeString,"%s<%f",UTime[Prefix]);
          FmtFile(TmpHndl[Prefix],"%s<%s\t\t%s\n",UnitTimeString,EnggString);
          UTime[Prefix]+=GTick[Prefix];
          if(!NoDPMDisp){k=0;Dii=TRUE;FndDPM=FALSE;
            while(k<DPMCount && Dii){
              if(DPM_ChNo[k]==Prefix){
                FndDPM=TRUE;Dii=FALSE;}
              else k++;}
            if(FndDPM)SetCtrlVal(GroupHandle,DPMCONST+k,EnggString);}
          if(!NoSCDisp){k=0;Dii=TRUE;FndSC=FALSE;
            while(k<SCCount && Dii){
              if(SC_ChNo[k]==Prefix){
                FndSC=TRUE;Dii=FALSE;}
              else k++;}
            if(FndSC)PlotStripChart(GroupHandle,OSCCONST+k,EnggUnit,1,0,0,4);}
          else{MessHandle=LoadPanel("Setup.uir",Message1);DisplayPanel(MessHandle);}}}
      break;
    default :
      break;}}
for(i=0;i<GrpCount;i++){bytes=TRUE;SetPrefix=GrpPageNo[i];SetFilePtr(TmpHndl[SetPrefix],0,0);
  while(bytes>0){bytes=ReadFile(TmpHndl[SetPrefix],Transferbyte,1024);
    WriteFile(SaveFileHndl,Transferbyte,bytes);}
  CloseFile(TmpHndl[SetPrefix]);}
      FmtFile(SaveFileHndl,
        %s<=========================================================================\n");
FmtFile(SaveFileHndl,"%s<Test Started \t Time(h:m:s) : %s\n",StrtTimeString);
FmtFile(SaveFileHndl,"%s<Test Ended \t Time(h:m:s) : %s\n",CurTimeString);
UnloadPanel(-1);NoteHandle=LoadPanel("setup.uir", Note);DisplayPanel(NoteHandle);
GetUserEvent(TRUE, &NoteHandle, &id);
if(id==Note_Done){GetCtrlVal(NoteHandle, Note_NoteText, NoteText);
  error=CompareStrings(NoteText,0,NullString,0,0);if(!error) Fmt(NoteText,"%s<NO NOTE");
  FmtFile(SaveFileHndl,
        "%s<=========================================================================\n");
  FmtFile(SaveFileHndl,"%s<Operator Note : %s\n",NoteText);}
CloseFile(SaveFileHndl);
RestoreFlags();UnloadMenuBar();
```

```
    OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu);FmtFile(DapOut,"%s<%s","reset\n");GrpPowerOff();
    UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);
}

/*
```

Individual Channel Configuration Routine

This routine displays the setup panel and writes the variables on to a *.CFG file, which can be loaded through File Menu Load on opening menu. Some of the important parameters which can be set are, Sampling rate, Scale factor, Offset, TimeDelay, Input Gain, and etc.

Input : None
Output : Loads the respective variables and saves under a CH#*.CFG file.

```
*/
void   SetupIndCh()
{
    Setii=TRUE;UnloadPanel(-1);ISetupHandle=LoadPanel("setup.uir", ISetup);
    if(ISetupHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
    DisplayPanel(ISetupHandle);OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu1);
    while(Setii){GetUserEvent(TRUE, &handle, &id);SetInputMode(OpenMenuBar,OpenMenu1_Help_Sys,0);
     switch(id){
       case ISetup_Done :
          chosen=ConfirmPopup("Write on to Configuration File ?");
          if(chosen){GetCtrlVal(ISetupHandle,ISetup_Sampling,&Sampling);
                 GetCtrlVal(ISetupHandle, ISetup_TimeDelay, &TDelay);
            GetCtrlVal(ISetupHandle,ISetup_Offset,&Offset);GetCtrlVal(ISetupHandle, ISetup_Scale, &Scale );
            GetCtrlVal(ISetupHandle,ISetup_ChNo,&ChNo);GetCtrlVal(ISetupHandle,ISetup_ChName,ChName);
           GetCtrlVal(ISetupHandle,ISetup_IpCh,ChSD);GetCtrlVal(ISetupHandle,ISetup_DispType,&IndDispType);
            GetCtrlVal(ISetupHandle,ISetup_Unit,&Unit1);GetCtrlVal(ISetupHandle,ISetup_UName,UnitName);
            error=CompareStrings(ChName,0,NullString,0,0);
                 if(!error)Fmt(ChName,"%s<NAME NOT SPECIFIED");
            GetCtrlVal(ISetupHandle,ISetup_ChType,IndChType);GetCtrlVal(ISetupHandle,ISetup_Gain,&IndGain);
            Fmt(IConfig,"%s<Ch#%i.cfg",ChNo);InfoChHndl[ChNo]=OpenFile(IConfig,0,0,1);
            FmtFile(InfoChHndl[ChNo],"%s<******* This is a GENERATED FILE, do not Edit *****\n");
            FmtFile(InfoChHndl[ChNo],"%s<1,%i,%s,%s,%i,%i,%f,%f,%i,%s,%i,",
               ChNo,ChName,ChSD,Sampling,TDelay,Offset,Scale,IndGain,UnitName,Unit1);
            IFlag=TRUE;Setii=FALSE;}break;
       case ISetup_DispType :
          GetCtrlVal(ISetupHandle,ISetup_DispType,&chosen);
          if(chosen)SetInputMode(ISetupHandle,ISetup_UName,0);
          else SetInputMode(ISetupHandle,ISetup_UName,1);
          break;
       case ISetup_Default :
          DefaultPanel(ISetupHandle);break;
       case ISetup_Cancel :
          chosen=ConfirmPopup("Are you sure ?");if(chosen){IFlag=FALSE;Setii=FALSE;RestoreFlags();}
          break;
       case OpenMenu_Help_Control :
            MessHandle=LoadPanel("help.uir", Help6);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(ISetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(ISetupHandle);
            RecallPanelState(ISetupHandle,"setup.rec");
            break;
         case OpenMenu_Help_Panel :
            MessHandle=LoadPanel("help.uir", Help5);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(ISetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(ISetupHandle);
            RecallPanelState(ISetupHandle,"setup.rec");
            break;
         default :break;}}
      UnloadPanel(-1); OpenHandle=LoadPanel("Ald.uir",Ald1); DisplayPanel(OpenHandle);
      OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu);
   }

   /*
```

Group Channel Configuration Setup

This routine displays the setup panel and reads variables on to a file *.CFG, which can be loaded later on through File Save Menu under Open Menu. The Sampling rate, Scale factor, Offset, Input gain, and Timedelay are some of the parameters which can be defined for a group of Analog input channels (Maximum = 512). These 512 channel's parameters are setup under 512 pages, one for each channel.

Input : None
Output : Initializes the variables and writes variables onto a *.CFG file

```
*/
void SetupGrpCh()
{
   UnloadPanel(-1);GSetupHandle=LoadPanel("setup.uir", GSetup);
   if(GSetupHandle<0){FmtOut(" Unable to load required panel from the resource file.,");return;}
   DisplayPanel(GSetupHandle);GrpCount=0;Grpii=TRUE;OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu1);
   GetProgramDir(dirname);
   if(FileSelectPopup(dirname,"*.cfg","Enter Configuration File Name",1,1,1,ConfigFName)==0){
      UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1);
      DisplayPanel(OpenHandle);OpenMenuBar=LoadMenuBar("ald.uir",OpenMenu);return;}
   InfoGrpChHndl=OpenFile(ConfigFName,0,0,1);NextFlag=0;
   FmtFile(InfoGrpChHndl,
      "%s<******* This is a GENERATED FILE, do not Edit *****\n");FmtFile(InfoGrpChHndl,"%s<0,");
   while(GrpCount<MAXCH && Grpii){Fmt(GrpChString,"%s<PAGE #%i.",GrpCount);
      if(GrpCount>0&&NextFlag)DefaultPanel(GSetupHandle);SetCtrlVal(GSetupHandle,GSetup_Text,GrpChString);
      GetUserEvent(TRUE,&handle,&id);
      switch(id){
         case GSetup_Next :
            chosen=ConfirmPopup("Write on to Configuration File ?");
```

```
  if(chosen){
     GetCtrlVal(GSetupHandle,GSetup_PageNo,&GrpPageNo[GrpCount]);
     SetPrefix=GrpPageNo[GrpCount];
     GetCtrlVal(GSetupHandle,GSetup_Sampling,&GrpSample[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_TDelay,&GrpTDelay[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_Offset,&GrpOffset[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_Scale,&GrpScale[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_Unit,&GrpUnit[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_UnitType,&GrpUnitType[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_ChSD,&GrpChSD[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_UName,&GrpChUName[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_ChType,&GrpChType[SetPrefix]);
     GetCtrlVal(GSetupHandle,GSetup_GrpGain,&GrpGain[SetPrefix]);
     FmtFile(InfoGrpChHndl,"%s<%i,%i,%f,%i,%f,%f,%i,%i,%i,%i,",
        SetPrefix,GrpChSD[SetPrefix],GrpSample[SetPrefix],GrpTDelay[SetPrefix],
        GrpOffset[SetPrefix],GrpScale[SetPrefix],GrpUnit[SetPrefix],
        GrpChType[SetPrefix],GrpUName[SetPrefix],GrpGain[SetPrefix]);
     GrpCount++;SetBackgroundColor(GrpCount);GrpFlag=TRUE;NextFlag=TRUE;}
   break;
case GSetup_Done :
   chosen=ConfirmPopup("Go with the Present Channel ?");
   if(chosen){
      GetCtrlVal(GSetupHandle,GSetup_PageNo,&GrpPageNo[GrpCount]);
      SetPrefix=GrpPageNo[GrpCount];
      GetCtrlVal(GSetupHandle,GSetup_Sampling,&GrpSample[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_TDelay,&GrpTDelay[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_Offset,&GrpOffset[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_Scale,&GrpScale[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_Unit,&GrpUnit[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_UnitType,&GrpUnitType[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_ChSD,&GrpChSD[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_UName,&GrpChUName[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_ChType,&GrpChType[SetPrefix]);
      GetCtrlVal(GSetupHandle,GSetup_GrpGain,&GrpGain[SetPrefix]);
      FmtFile(InfoGrpChHndl,"%s<%i,%i,%f,%i,%f,%f,%i,%i,%i,%i,",
         SetPrefix,GrpChSD[SetPrefix],GrpSample[SetPrefix],GrpTDelay[SetPrefix],
         GrpOffset[SetPrefix],GrpScale[SetPrefix],GrpUnit[SetPrefix],
         GrpChType[SetPrefix],GrpUName[SetPrefix],GrpGain[SetPrefix]);
      GrpFlag=TRUE;GrpCount++;Grpii=FALSE;}
    if(!chosen)Grpii=FALSE;
    break;
  case GSetup_UnitType :
     GetCtrlVal(GSetupHandle,GSetup_UnitType,&chosen);
     if(chosen)SetInputMode(GSetupHandle,GSetup_UName,0);
     else SetInputMode(GSetupHandle,GSetup_UName,1);NextFlag=FALSE;
     break;
  case GSetup_Default :
     NextFlag=TRUE;
     break;
  case GSetup_Cancel :
  case OpenMenu_File_Quit :
     chosen=ConfirmPopup("Are you sure ?");
     if(chosen){RestoreFlags();
```

```
            GrpFlag=FALSE;Grpii=FALSE;GrpCount=0;}
         NextFlag=FALSE;
         break;
      case OpenMenu_Help_Control :
         MessHandle=LoadPanel("help.uir", Help6);
         if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
         SavePanelState(GSetupHandle,"setup.rec");DisplayPanel(MessHandle);
         GetUserEvent(TRUE,&handle,&id);UnloadPanel(MessHandle);DisplayPanel(GSetupHandle);
         RecallPanelState(GSetupHandle,"setup.rec");
         break;
      case OpenMenu_Help_Panel :
         MessHandle=LoadPanel("help.uir", Help5);
         if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
         SavePanelState(GSetupHandle,"setup.rec");DisplayPanel(MessHandle);
         GetUserEvent(TRUE,&handle,&id);UnloadPanel(MessHandle);DisplayPanel(GSetupHandle);
         RecallPanelState(GSetupHandle,"setup.rec");
         break;
      default :
         break;}}
  SetBackgroundColor(0);UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1); DisplayPanel(OpenHandle);
  OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu);
}

void    SetupSimulCh(){}

/*
```

File Storage Format (Individual and Group Channels)

This routine displays the file format setup panel reads into file format variables and writes them on to the same *.CFG file created by channel configuration setup.

Input : Opened *.CFG file
Output : Initializes the variables and writes them on to .CFg file.

```
*/
void    FileSaveSetup()
{
  ii=TRUE;UnloadPanel(-1);SaveHandle=LoadPanel("setup.uir",SaveExt);UnloadPanel(-1);
  if(SaveHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
  DisplayPanel(SaveHandle);Save1Handle=LoadPanel("setup.uir",FilePanel);
  if(Save1Handle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
  DisplayPanel(Save1Handle);SetActivePanel(Save1Handle);FISetupHandle = LoadPanel("setup.uir", FileInd);
  if(FISetupHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
  FGSetupHandle=LoadPanel("setup.uir",FileGroup);SetInputMode(OpenMenuBar,OpenMenu_Help_Control,1);
  if(FGSetupHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
  OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu1);
  GetUserEvent(TRUE,&handle,&id);
     switch(id){
     case FilePanel_Ind : DisplayPanel(FISetupHandle);SetInputMode(Save1Handle,FilePanel_Grp,0);
        SetInputMode(Save1Handle,FilePanel_Ind,0);
        while(ii){GetUserEvent(TRUE, &handle, &ndx);
```

```
    switch(ndx){
    case FileInd_Done :
       if(IFlag){
          chosen=ConfirmPopup("Writing on to Configuration File ?");
          if(chosen){GetCtrlVal(FISetupHandle,FileInd_FileName,FileName);
             GetCtrlVal(FISetupHandle,FileInd_Precn,&StorePrecn);
             GetCtrlVal(FISetupHandle,FileInd_Buffer,&FileBuffer);
             GetCtrlVal(FISetupHandle,FileInd_Format,&Format);
             GetCtrlVal(FISetupHandle,FileInd_SpecimenNo,Specimen);
             error=CompareStrings(Specimen,0,NullString,0,0);
               if(!error) Fmt(Specimen,"%s<ID NOT SPECIFIED");
             FmtFile(InfoChHndl[FileChNo],"%s<1,%i,%i,%s,%s,%i,",
                  Format,FileBuffer,Specimen,FileName,StorePrecn);
             ii=FALSE; FIFlag=TRUE;}}
       else {MessagePopup("Individual channel setup not done, run Setup");ii=FALSE;}
       break;
    case FileInd_Default :
       FISetupHandle=LoadPanel("setup.uir",FileInd);
       DisplayPanel(FISetupHandle);
       break;
    case OpenMenu_Help_Control :
       MessHandle=LoadPanel("help.uir", Help6);
       if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
       SavePanelState(GSetupHandle,"setup.rec");DisplayPanel(MessHandle);
       GetUserEvent(TRUE,&handle,&id);UnloadPanel(MessHandle);DisplayPanel(GSetupHandle);
       RecallPanelState(GSetupHandle,"setup.rec");
       break;
    case OpenMenu_Help_Panel :
       MessHandle=LoadPanel("help.uir", Help5);
       if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
       SavePanelState(GSetupHandle,"setup.rec");DisplayPanel(MessHandle);
       GetUserEvent(TRUE,&handle,&id);UnloadPanel(MessHandle);DisplayPanel(GSetupHandle);
       RecallPanelState(GSetupHandle,"setup.rec");
       break;
    case OpenMenu_File_Quit :
       ii=FALSE;
       break;
    case OpenMenu_File_Print :
       chosen=ConfirmPopup("Screen Print selected, Continue Printing");
       if(chosen==TRUE)OutputScreen(0, "");
       break;
    case FileInd_Cancel :
       chosen=ConfirmPopup("Are you sure ?");
       if(chosen) ii=FALSE;
       break;
    default :
       break;}}
  break;
case FilePanel_Grp : FGSetupHandle=LoadPanel("setup.uir", FileGroup);
  SetInputMode(Save1Handle,FilePanel_Ind,0);SetInputMode(Save1Handle,FilePanel_Grp,0);
  DisplayPanel(FGSetupHandle);SetActivePanel(FGSetupHandle);ii=TRUE;
  while(ii){GetUserEvent(TRUE, &handle, &id);
    if(id==FileGroup_Done && GrpFlag){chosen=ConfirmPopup("Writing on to Configuration File ?");
```

```
            if(chosen){GetCtrlVal(FGSetupHandle,FileGroup_GrpFName,GrpFName);
              GetCtrlVal(FGSetupHandle,FileGroup_Buffer,&FileBuffer);
              GetCtrlVal(FGSetupHandle,FileGroup_BufferType,&BufferType);
              GetCtrlVal(FGSetupHandle,FileGroup_Format,&Format);
              GetCtrlVal(FGSetupHandle,FileGroup_GrpPrecn,&GrpPrecn);
              FmtFile(InfoGrpChHndl,"%s<77,%i,%i,%s,%i,%i,%s,",
              Format,FileBuffer,Specimen,BufferType,GrpPrecn,GrpFName);
              FGFlag=TRUE;ii=FALSE;}}
            if(id==FileGroup_Done && !GrpFlag){
                MessagePopup("Group Channel Setup not done, Run 'Setup'");ii=FALSE;}
            if(id==FileGroup_Default){FGSetupHandle=LoadPanel("setup.uir",FileGroup);
                DisplayPanel(FGSetupHandle);}
            if(handle==OpenMenuBar && id==OpenMenu_Help){}
            if(id==FileGroup_Cancel)ii=FALSE;
                if(handle==OpenMenuBar && id==OpenMenu_File_Quit)ii=FALSE;
            if(id==OpenMenu_File_Quit)ii=FALSE;
            if(handle==OpenMenuBar && id==OpenMenu_File_Print){
              chosen=ConfirmPopup("Screen Print selected, Continue Printing");
              if(chosen)OutputScreen(0, "");}}
        break;
      case FilePanel_Simul :
        break;
      case OpenMenu_File_Quit :
        ii=FALSE;
        break;
      default :
        break;}
  UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);id=0;
  OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu);
}
/*
```

Display Format Setup Routine (Individual Channel)

This routine displays the setup panel and initializes the display variables and writes parameters on to a *.CFG file created earlier by the configuration setup procedure. Digital Panel Meter and Strip Chart (Real-Time Graph) can be color coded with Y scale factor defined.

Input : Opened .CFG file
Output : Initializes the display variables and writes them onto .CFG file.

```
*/
void SetupIndDisplay()
{
  SetDii=TRUE;UnloadPanel(-1);DSetupHandle=LoadPanel("setup.uir",DSetupInd);
  if(DSetupHandle<0){FmtOut(" Unable to load required panel from the resource
                                        file.\n");return;}DisplayPanel(DSetupHandle);
  while(SetDii){
    GetUserEvent(TRUE,&DSetupHandle,&id);
    switch(id){
      case DSetupInd_Done :
/*------------------------ DPM Variables ----------------------*/
```

```
    chosen=ConfirmPopup("Write on to Configuration File ?");
    if(chosen){GetCtrlVal(DSetupHandle,DSetupInd_DPM_Color,&DPM_Color);
       GetCtrlVal(DSetupHandle,DSetupInd_Precn,&Precn);
       GetCtrlVal(DSetupHandle,DSetupInd_DPMB_Color,&DPMB_Color);
       GetCtrlVal(DSetupHandle,DSetupInd_DispUnit,&DispUnit);
/*-------------- Strip Chart Variables ------------------*/
       GetCtrlVal(DSetupHandle,DSetupInd_SCB_Color,&SCB_Color);
       GetCtrlVal(DSetupHandle,DSetupInd_SCT_Color,&SCT_Color);
       GetCtrlVal(DSetupHandle,DSetupInd_Grid,&Grid);
       GetCtrlVal(DSetupHandle,DSetupInd_YScaleMin,&YScaleMin);
       GetCtrlVal(DSetupHandle,DSetupInd_YScaleMax,&YScaleMax);
       GetCtrlVal(DSetupHandle,DSetupInd_YAxisName,YAxisName);
       FmtFile(InfoChHndl[ChNo],"%s<0,%i,%i,%i,%i,%i,%i,%i,%f,%f,%s,",
          DPM_Color,Precn,DPMB_Color,DispUnit,SCB_Color,SCT_Color,Grid,
          YScaleMin,YScaleMax,YAxisName);
       SetDii=FALSE;DIFlag=TRUE;}
    break;
  case DSetupInd_DPMB_Color :
    GetCtrlVal(DSetupHandle,DSetupInd_DPMB_Color,&DPMB_Color);
    SetCtrlAttribute(DSetupHandle,DSetupInd_DPM,6,DPMB_Color);
    break;
  case DSetupInd_DPM_Color :
    GetCtrlVal(DSetupHandle,DSetupInd_DPM_Color,&DPM_Color);
    SetCtrlAttribute(DSetupHandle,DSetupInd_DPM,4,DPM_Color);
    break;
  case DSetupInd_SCB_Color :
    GetCtrlVal(DSetupHandle,DSetupInd_SCB_Color,&SCB_Color);
    SetGraphAttribute(DSetupHandle,DSetupInd_DSC,5,SCB_Color);
    break;
  case DSetupInd_Grid :
    GetCtrlVal(DSetupHandle,DSetupInd_SCB_Color,&BColor);
    GetCtrlVal(DSetupHandle,DSetupInd_Grid,&Grid);
    if(!Grid)SetGraphAttribute(DSetupHandle,DSetupInd_DSC,3,BColor);
    else SetGraphAttribute(DSetupHandle,DSetupInd_DSC,3,8);
    break;
  case DSetupInd_YScaleMin :
    GetCtrlVal(DSetupHandle,DSetupInd_YScaleMin,&YScaleMin);
    ConfigureAxes(DSetupHandle,DSetupInd_DSC,-1,0,0,0,YScaleMin,YScaleMax);
    break;
  case DSetupInd_YScaleMax :
    GetCtrlVal(DSetupHandle,DSetupInd_YScaleMax,&YScaleMax);
    ConfigureAxes(DSetupHandle,DSetupInd_DSC,-1,0,0,0,YScaleMin,YScaleMax);
    break;
  case DSetupInd_YAxisName :
    GetCtrlVal(DSetupHandle,DSetupInd_YAxisName,YAxisName);
    SetGraphAttribute(DSetupHandle,DSetupInd_DSC,18,YAxisName);
    break;
  case DSetupInd_Default :
    UnloadPanel(-1);DSetupHandle=LoadPanel("setup.uir",DSetupInd);
    DisplayPanel(DSetupHandle);
    break;
  case DSetupInd_Cancel :
    RestoreFlags();SetDii=FALSE;
```

```
            break;
         case OpenMenu_Help_Control :
            MessHandle=LoadPanel("help.uir", Help6);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
            RecallPanelState(DSetupHandle,"setup.rec");
            break;
         case OpenMenu_Help_Panel :
            MessHandle=LoadPanel("help.uir", Help5);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
            RecallPanelState(DSetupHandle,"setup.rec");
            break;
         default :
            break;}}
   SetBackgroundColor(0);OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu);UnloadPanel(-1);
   OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);
}

/*
```

Display Format Setup Routine (Group Channels)

This routine displays in pages the setup panel and initializes the display array variables and writes parameters on to a *.CFG file created earlier by the configuration setup procedure. Channels can be configured on to any of the four DPMs and Strip charts. Digital Panel Meter and Strip Chart (Real-Time Graph) can be color coded with different Y scale factor defined.

Input : Opened .CFG file
Output : Initializes the display variables and writes them onto .CFG file.

```
*/
void SetupGrpDisplay()
{
   UnloadPanel(-1);DHandle=LoadPanel("setup.uir",DispExt);
   SetInputMode(OpenMenuBar,OpenMenu_Help_Control,0);
   if(DHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
   DisplayPanel(DHandle);D1Handle=LoadPanel("setup.uir",DispPanel);
   if(D1Handle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
   DisplayPanel(D1Handle);OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu1);
   GetUserEvent(TRUE,&handle,&id);
   if(id==DispPanel_SCSw){
      Dii=TRUE;UnloadPanel(-1);PageCount=0;DSetupHandle=LoadPanel("setup.uir",DSetGrpSC);
      if(DSetupHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
      DisplayPanel(DSetupHandle);SetActivePanel(DSetupHandle);
      if(GrpCount<=4)MaxDisp=GrpCount;else MaxDisp=4;
      FmtFile(InfoGrpChHndl,"%s<88,");NextFlag=FALSE;SCCount=0;
```

```
while(PageCount<MaxDisp && Dii){
   if(PageCount>0 && NextFlag){DefaultPanel(DSetupHandle);SetBackgroundColor(GrpCount);}
   Fmt(GrpChString,"%s<PAGE#%i.",PageCount);SetCtrlVal(DSetupHandle,DSetGrpSC_Page,GrpChString);
   GetUserEvent(TRUE,&DSetupHandle,&id);
   switch(id){
      case DSetGrpSC_Next :
         chosen=ConfirmPopup("Write on to Configuration File ?");
         if(chosen){
/*--------------- Strip Chart Parameters ---------------------*/
            GetCtrlVal(DSetupHandle,DSetGrpSC_SC_ChNo,&SC_ChNo[PageCount]);
            SetSCPrefix=SC_ChNo[PageCount];
            GetCtrlVal(DSetupHandle,DSetGrpSC_SCB_Color,&GSCB_Color[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_SCT_Color,&GSCT_Color[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_Grid,&GGrid[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMin,&GYScaleMin[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMax,&GYScaleMax[SetSCPrefix]);
            FmtFile(InfoGrpChHndl,"%s<%i,%i,%i,%i,%f,%f,",
               SetSCPrefix,GSCB_Color[SetSCPrefix],GSCT_Color[SetSCPrefix],GGrid[SetSCPrefix],
               GYScaleMin[SetSCPrefix],GYScaleMax[SetSCPrefix]);
            PageCount++;DGSCFlag=TRUE;SCCount++;NoSCDisp=FALSE;NextFlag=TRUE;
            SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,5,0);
                  SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,3,8);}
         break;
      case DSetGrpSC_Done :
         chosen=ConfirmPopup("Go with the present page setup ?");
         if(chosen){
/*--------------- Strip Chart Parameters -------------------*/
            GetCtrlVal(DSetupHandle,DSetGrpSC_SC_ChNo,&SC_ChNo[PageCount]);
            SetSCPrefix=SC_ChNo[PageCount];
            GetCtrlVal(DSetupHandle,DSetGrpSC_SCB_Color,&GSCB_Color[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_SCT_Color,&GSCT_Color[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_Grid,&GGrid[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMin,&GYScaleMin[SetSCPrefix]);
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMax,&GYScaleMax[SetSCPrefix]);
            FmtFile(InfoGrpChHndl,"%s<%i,%i,%i,%i,%f,%f,",
               SetSCPrefix,GSCB_Color[SetSCPrefix],GSCT_Color[SetSCPrefix],GGrid[SetSCPrefix],
               GYScaleMin[SetSCPrefix],GYScaleMax[SetSCPrefix]);
            PageCount++;DGSCFlag=TRUE;SCCount++;NoSCDisp=FALSE;Dii=FALSE;
            SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,5,0);
              SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,3,8);}
         if(!chosen && !DGSCFlag){DGSCFlag=TRUE;NoSCDisp=TRUE;Dii=FALSE;}
         else Dii=FALSE;
         break;
      case DSetGrpSC_SCB_Color :
         GetCtrlVal(DSetupHandle,DSetGrpSC_SCB_Color,&SCB_Color);
         SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,5,SCB_Color);NextFlag=0;
         break;
      case DSetGrpSC_Grid :
         GetCtrlVal(DSetupHandle,DSetGrpSC_SCB_Color,&BColor);
         GetCtrlVal(DSetupHandle,DSetGrpSC_Grid,&Grid);NextFlag=0;
         if(!Grid)SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,3,BColor);
         else SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,3,8);
         break;
         case DSetGrpSC_YScaleMin :
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMin,&YScaleMin);
            ConfigureAxes(DSetupHandle,DSetGrpSC_DSC,-1,0,0,0,YScaleMin,YScaleMax);NextFlag=0;
            break;
         case DSetGrpSC_YScaleMax :
            GetCtrlVal(DSetupHandle,DSetGrpSC_YScaleMax,&YScaleMax);
            ConfigureAxes(DSetupHandle,DSetGrpSC_DSC,-1,0,0,0,YScaleMin,YScaleMax);NextFlag=0;
            break;
         case DSetGrpSC_YAxisName :
            GetCtrlVal(DSetupHandle,DSetGrpSC_YAxisName,YAxisName);
            SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,18,YAxisName);NextFlag=0;
            break;
         case DSetGrpSC_Default : NextFlag=TRUE;
            SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,5,0);
                  SetGraphAttribute(DSetupHandle,DSetGrpSC_DSC,3,8);
            Fmt(GrpChString,"%s<PAGE #%i.",PageCount);
                  SetCtrlVal(DSetupHandle,DSetGrpSC_Page,GrpChString);
            break;
         case DSetGrpSC_Cancel :
            chosen=ConfirmPopup("Ignore complete display setup ?");
            if(chosen){
               for(i=0;i<MAXCH;i++){GSCB_Color[i]=0;GSCT_Color[i]=0;GYScaleMin[i]=0;
                  GYScaleMax[i]=0;GGrid[i]=0;SC_ChNo[i]=0;}
               Dii=FALSE;DGSCFlag=FALSE;SCCount=0;}
            break;
         case OpenMenu_Help_Control :
            MessHandle=LoadPanel("help.uir", Help6);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
            RecallPanelState(DSetupHandle,"setup.rec");
            break;
         case OpenMenu_Help_Panel :
            MessHandle=LoadPanel("help.uir", Help5);
            if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
            SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
            GetUserEvent(TRUE,&handle,&id);
            UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
            RecallPanelState(DSetupHandle,"setup.rec");
            break;
         default :
            break;}}}
if(id==DispPanel_DPMSw){
   Dii=TRUE;UnloadPanel(-1);PageCount=0;DSetupHandle=LoadPanel("setup.uir",DSetGrpDPM);
   if(DSetupHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
   DisplayPanel(DSetupHandle);SetActivePanel(DSetupHandle);if(GrpCount<=4)MaxDisp=GrpCount;else
   MaxDisp=4;
   FmtFile(InfoGrpChHndl,"%s<99,");NextFlag=FALSE;DPMCount=0;
   while(PageCount<MaxDisp && Dii){
      if(PageCount>0 && NextFlag){DefaultPanel(DSetupHandle);SetBackgroundColor(GrpCount);}
     Fmt(GrpChString,"%s<PAGE%i.",PageCount);SetCtrlVal(DSetupHandle,DSetGrpDPM_Page,GrpChString);
      GetUserEvent(TRUE,&DSetupHandle,&id);
```

```
switch(id){
   case DSetGrpDPM_Next :
      chosen=ConfirmPopup("Write on to Configuration File ?");
      if(chosen){
/*---------- Display Parameters --------------*/
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPM_ChNo,&DPM_ChNo[PageCount]);
         SetDPMPrefix=DPM_ChNo[PageCount];
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPM_Color,&GDPM_Color[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_Precn,&GPrecn[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPMB_Color,&GDPMB_Color[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DispUnit,&GDispUnit[SetDPMPrefix]);
         FmtFile(InfoGrpChHndl,"%s<%i,%i,%i,%i,%i,",
            SetDPMPrefix,GDPM_Color[SetDPMPrefix],GPrecn[SetDPMPrefix],
            GDPMB_Color[SetDPMPrefix],GDispUnit[SetDPMPrefix]);
         PageCount++;DGDPMFlag=TRUE;DPMCount++;NoDPMDisp=FALSE;NextFlag=TRUE;
         SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,6,0);
               SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,4,14);}
      break;
   case DSetGrpDPM_Done :
      chosen=ConfirmPopup("Go with the present page setup ?");
      if(chosen){
/*---------------- Display Parameters ----------------*/
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPM_ChNo,&DPM_ChNo[PageCount]);
         SetDPMPrefix=DPM_ChNo[PageCount];
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPM_Color,&GDPM_Color[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_Precn,&GPrecn[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DPMB_Color,&GDPMB_Color[SetDPMPrefix]);
         GetCtrlVal(DSetupHandle,DSetGrpDPM_DispUnit,&GDispUnit[SetDPMPrefix]);
         FmtFile(InfoGrpChHndl,"%s<%i,%i,%i,%i,%i,",
            SetDPMPrefix,GDPM_Color[SetDPMPrefix],GPrecn[SetDPMPrefix],
            GDPMB_Color[SetDPMPrefix],GDispUnit[SetDPMPrefix]);
         PageCount++;DGDPMFlag=TRUE;DPMCount++;NoDPMDisp=FALSE;Dii=FALSE;
         SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,6,0);
               SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,4,14);}
               SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,4,14);}
      if(!chosen && !DGDPMFlag){DGDPMFlag=TRUE;NoDPMDisp=TRUE;Dii=FALSE;}
      else Dii=FALSE;
      break;
   case DSetGrpDPM_DPMB_Color :
      GetCtrlVal(DSetupHandle,DSetGrpDPM_DPMB_Color,&DPMB_Color);
      SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,6,DPMB_Color);NextFlag=0;
      break;
   case DSetGrpDPM_DPM_Color :
      GetCtrlVal(DSetupHandle,DSetGrpDPM_DPM_Color,&DPM_Color);
      SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,4,DPM_Color);NextFlag=0;
      break;
   case DSetGrpDPM_Default : NextFlag=TRUE;
      SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,6,0);
         SetCtrlAttribute(DSetupHandle,DSetGrpDPM_DPM,4,14);
      Fmt(GrpChString,"%s<PAGE  #%i.",PageCount);
         SetCtrlVal(DSetupHandle,DSetGrpDPM_Page,GrpChString);
      break;
   case DSetGrpDPM_Cancel :
      chosen=ConfirmPopup("Ignore complete display setup ?");
```

```
                if(chosen){
                    for(i=0;i<MAXCH;i++){GDPM_Color[i]=0;GDPMB_Color[i]=0;DPM_ChNo[i]=0;}
                    Dii=FALSE;DGDPMFlag=FALSE;DPMCount=0;}
                break;
              case OpenMenu_Help_Control :
                MessHandle=LoadPanel("help.uir",Help6);
                if(MessHandle<0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
                GetUserEvent(TRUE,&handle,&id);
                UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
                RecallPanelState(DSetupHandle,"setup.rec");
                break;
              case OpenMenu_Help_Panel :
                MessHandle=LoadPanel("help.uir", Help5);
                if(MessHandle < 0){FmtOut(" Unable to load required panel from the resource file.\n");return;}
                SavePanelState(DSetupHandle,"setup.rec");DisplayPanel(MessHandle);
                GetUserEvent(TRUE,&handle,&id);
                UnloadPanel(MessHandle);DisplayPanel(DSetupHandle);
                RecallPanelState(DSetupHandle,"setup.rec");
                break;
              default :
                break;}}}
   if(id==OpenMenu_Help_Panel || id==OpenMenu_Help_Control){
      MessagePopup("Context help function not available at this time");}
   SetBackgroundColor(0);OpenMenuBar=LoadMenuBar("Ald.uir", OpenMenu);UnloadPanel(-1);
   OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);id=0;ErrFlag=FALSE;
}

/*-----  "Write Dap File" is the routine for writing on to disk the DAP OS task routines ------*/

int     WriteIndDapFile(DapFileName,DapSD,DapChNo,DapGain,DapSample)
char     *DapFileName,DapSD[4];
int     DapGain;
{
char    str1[20],str2[20],str3[20],str4[20],str5[25],str6[20],str7[20];
char    str8[20],str9[20],str10[20],str11[20],str12[20];
double  ChCount;
   Fmt(DapFName,"%s<%s.DAP",DapFileName);DapFileHndl=OpenFile(DapFName,0,0,1);
   if(DapFileHndl != -1){Fmt(str1,"%s<RESET");Fmt(str2,"%s<PIPE  P");
      Fmt(str3,"%s<IDEFINE");Fmt(str4,"%s<SET  CHANNEL 0");
      Fmt(str5,"%s<TIME1000.0");Fmt(str6,"%s<DIRECT(0,P");Fmt(str7,"%s<END");
      Fmt(str8,"%s<PDEFINE");Fmt(str9,"%s<FORMAT(P");Fmt(str10,"%s<START");
      FmtFile(DapFileHndl,
         "%s<%s\n%s%i\n%s  %s 1\n\t%s  %c%i  %i\n\t%s\n\t%s%i,%i)\n\t%s\n%s  %s1\n\t%s%i)\n\t%s\n%s
         %s,%s1", str1,str2,DapChNo,str3,DapFileName,str4,DapSD[0],DapChNo,DapGain,str5,str6,
          ChNo,DapSample,str7,str8,DapFileName,str9,ChNo,str7,str10,DapFileName,DapFileName);
      CloseFile(DapFileHndl);}
   return(DapFileHndl);
}
```

```
/*  "Write Dap File" is the routine for writing on to disk the DAP OS task routines */
int     WriteGrpDapFile()
{
  Fmt(DapFName,"%s<GROUP.DAP");DapFileHndl=OpenFile(DapFName,0,0,1);
  if(DapFileHndl != -1){FmtFile(DapFileHndl, "%s<RESET\n");FmtFile(DapFileHndl, "%s<PIPE ");
     for(i=0;i<GrpCount-1;i++){SetPrefix=GrpPageNo[i];
        FmtFile(DapFileHndl, "%s<P%i,",SetPrefix);}
     FmtFile(DapFileHndl, "%s<P%i\n",GrpPageNo[i]);
     FmtFile(DapFileHndl, "%s<IDEFINE GRP %i\n",GrpCount);
     for(i=0;i<GrpCount;i++){SetPrefix=GrpPageNo[i];
        if(GrpChSD[SetPrefix])Fmt(GrpChSD1,"%s<D");else  Fmt(GrpChSD1,"%s<S");
        FmtFile(DapFileHndl,
             "%s<\tSET CHANNEL %i %c%i %i\n",i,GrpChSD1[0],SetPrefix,GrpGain[SetPrefix]);}
     FmtFile(DapFileHndl, "%s<\tTIME ");Total=0.0;Greatest=GrpSample[0];
     for(i=1;i<MAXCH;i++){
        if(Greatest<GrpSample[i])Greatest=GrpSample[i];}
     TSampling=Greatest*GrpCount;ThroWhile=FALSE;
     Fmt(&TotalSampling,"%f<%i",TSampling);
     TimeConst=1000*(1000/TotalSampling);
     while(TimeConst>26000.0){
        TotalSampling*=2;
        TimeConst=1000*(1000/TotalSampling);
        ThroWhile=TRUE;}
     FmtFile(DapFileHndl,"%s<%f[p1]\n",TimeConst);TotalSamPerCh=(TotalSampling/GrpCount);
     for(i=0;i<GrpCount;i++){
        SetPrefix=GrpPageNo[i];
        if(ThroWhile)SkipCount[SetPrefix]=TotalSamPerCh/GrpSample[SetPrefix];
        else SkipCount[SetPrefix]=Greatest/GrpSample[SetPrefix];}
     for(i=0;i<GrpCount;i++){
        SetPrefix=GrpPageNo[i];
        FmtFile(DapFileHndl,"%s<\tDIRECT (%i, P%i, %i)\n",i,SetPrefix,SkipCount[SetPrefix]);}
     FmtFile(DapFileHndl,"%s<\tEND\n");FmtFile(DapFileHndl,"%s<PDEFINE  GRP_A\n");
     for(i=0;i<GrpCount;i++){
        SetPrefix=GrpPageNo[i];
        FmtFile(DapFileHndl,"%s<\tFORMAT  (\"%i\",P%i)\n",SetPrefix,SetPrefix);}
     FmtFile(DapFileHndl,"%s<\tEND\n");
        FmtFile(DapFileHndl,"%s<START  GRP,GRP_A\n");
     CloseFile(DapFileHndl);return(DapFileHndl);}
  else{MessagePopup("Write error for DAP file");return;}
}
```

```
/*
```

Reading Configuration File (Individual and Group channels)

This routine reads the selected *.cfg file and loads the data on to the respective variables. The routine automatically selects group or individual setup from the data read.

Input : None
Output : Initailizes the experiment variables

```
*/

void ReadCfgFile()
{
   GetProgramDir(dirname);
   if(FileSelectPopup(dirname,"*.cfg", "Select a Config. File to load",1,1,1,FName) != 1)return;
   handle=OpenFile(FName,1,2,1);if(handle == -1){MessagePopup("Unable to open configuration file !");return;}
/*----if(~error1){MessagePopup("Invalidconfigurationfile!");CloseFile(handle);UnloadPanel(MessHandle);return;}---*/
   MessHandle=LoadPanel("setup.uir",Message);DisplayPanel(MessHandle);Delay(2);
   ScanFile(handle,"%s> %i[x]",&GrpCount);ScanFile(handle,"%s> %i[x]",&SCCount);
   ScanFile(handle,"%s> %i[x]",&DPMCount);SetFilePtr(handle,STRINGLEN+1,0);
   ScanFile(handle,"%s> %i[x]",&FileIndiv);
/*--------- FileIndiv = 1 ; Individual File Type -------*/
   if(FileIndiv){
      ScanFile(handle,"%s> %i[x]%s[xt44]%s[xt44]%i[x]%i[x]%f[x]%f[x]%i[x]%s[xt44]%i[x]%i[x]",
         &ChNo,ChName,ChSD,&Sampling,&TDelay,&Offset,&Scale,&IndGain,UnitName,&Unit1,&chosen);
      if(chosen){ScanFile(handle,"%s> %i[x]%i[x]%s[xt44]%s[xt44]%i[x]",
            &Format,&FileBuffer,Specimen,FileName,&StorePrecn);
         ScanFile(handle,"%s> %i[x]%i[x]%i[x]%i[x]%i[x]%i[x]%i[x]",
            &error,&DPM_Color,&Precn,&DPMB_Color,&DispUnit,&SCB_Color,&SCT_Color);
         ScanFile(handle,"%s> %i[x]%f[x]%f[x]%s[xt44]",
            &Grid,&YScaleMin,&YScaleMax,YAxisName);}
      if(!chosen){ScanFile(handle,"%s> %i[x]%i[x]%i[x]%i[x]%i[x]%i[x]",
            &DPM_Color,&Precn,&DPMB_Color,&DispUnit,&SCB_Color,&SCT_Color);
         ScanFile(handle,"%s> %i[x]%f[x]%f[x]%s[xt44]",
            &Grid,&YScaleMin,&YScaleMax,YAxisName);
         ScanFile(handle,"%s> %i[x]%i[x]%i[x]%s[xt44]%s[xt44]%i[x]",
            &error,&Format,&FileBuffer,Specimen,FileName,&StorePrecn);}
      IFlag=TRUE;FIFlag=TRUE;DIFlag=TRUE;ReLoad=TRUE;}
/*--------- FileIndiv = 0 ; Group File Type ----------------*/
   if(!FileIndiv){
      for(i=0;i<GrpCount;i++){
         ScanFile(handle,"%s> %i[x]",&GrpPageNo[i]);
         SetPrefix=GrpPageNo[i];
         ScanFile(handle,"%s> %i[x]%f[x]%i[x]%f[x]%f[x]%i[x]%i[x]%i[x]%i[x]",
            &GrpChSD[SetPrefix],&GrpSample[SetPrefix],&GrpTDelay[SetPrefix],
            &GrpOffset[SetPrefix],&GrpScale[SetPrefix],&GrpUnit[SetPrefix],
            &GrpChType[SetPrefix],&GrpUName[SetPrefix],&GrpGain[SetPrefix]);}
      ScanFile(handle,"%s> %i[x]",&FileRDisp);
      if(FileRDisp==FILEONLY){
         ScanFile(handle,"%s> %i[x]%i[x]%s[xt44]%i[x]%i[x]%s[xt44]",
     &Format,&FileBuffer,Specimen,&BufferType,&GrpPrecn,GrpFName);
  ScanFile(handle,"%s>%i[x]",&ScRDpm);
  if(ScRDpm==SCONLY){
     for(i=0;i<SCCount;i++){ScanFile(handle,"%s>%i[x]",&SC_ChNo[i]);SetSCPrefix=SC_ChNo[i];
        ScanFile(handle,"%s>%i[x]%i[x]%i[x]%f[x]%f[x]",
           &GSCB_Color[SetSCPrefix],&GSCT_Color[SetSCPrefix],
           &GGrid[SetSCPrefix],&GYScaleMin[SetSCPrefix],
           &GYScaleMax[SetSCPrefix]);}
     ScanFile(handle,"%s>%i[x]",&Dummy);
     for(i=0;i<DPMCount;i++){
              ScanFile(handle,"%s>%i[x]",&DPM_ChNo[i]);SetDPMPrefix=DPM_ChNo[i];
        ScanFile(handle,"%s>%i[x]%i[x]%i[x]%i[x]",
           &GDPM_Color[SetDPMPrefix],&GPrecn[SetDPMPrefix],
           &GDPMB_Color[SetDPMPrefix],&GDispUnit[SetDPMPrefix]);}}
          if(ScRDpm==DPMONLY){
     for(i=0;i<DPMCount;i++){
              ScanFile(handle,"%s>%i[x]",&DPM_ChNo[i]);SetDPMPrefix=DPM_ChNo[i];
        ScanFile(handle,"%s>%i[x]%i[x]%i[x]%i[x]",
           &GDPM_Color[SetDPMPrefix],&GPrecn[SetDPMPrefix],
           &GDPMB_Color[SetDPMPrefix],&GDispUnit[SetDPMPrefix]);}
     ScanFile(handle,"%s>%i[x]",&Dummy);
     for(i=0;i<SCCount;i++){ScanFile(handle,"%s>%i[x]",&SC_ChNo[i]);SetSCPrefix=SC_ChNo[i];
        ScanFile(handle,"%s>%i[x]%i[x]%i[x]%f[x]%f[x]",
           &GSCB_Color[SetSCPrefix],&GSCT_Color[SetSCPrefix],
           &GGrid[SetSCPrefix],&GYScaleMin[SetSCPrefix],
           &GYScaleMax[SetSCPrefix]);}}}
if(FileRDisp==DPMONLY || FileRDisp==SCONLY){ScRDpm=FileRDisp;
   if(ScRDpm==SCONLY){
      for(i=0;i<SCCount;i++){ScanFile(handle,"%s>%i[x]",&SC_ChNo[i]);SetSCPrefix=SC_ChNo[i];
         ScanFile(handle,"%s>%i[x]%i[x]%i[x]%f[x]%f[x]",
            &GSCB_Color[SetSCPrefix],&GSCT_Color[SetSCPrefix],
            &GGrid[SetSCPrefix],&GYScaleMin[SetSCPrefix],
            &GYScaleMax[SetSCPrefix]);}
      ScanFile(handle,"%s>%i[x]",&Dummy);
      for(i=0;i<DPMCount;i++){
         ScanFile(handle,"%s>%i[x]",&DPM_ChNo[i]);
         SetDPMPrefix=DPM_ChNo[i];
         ScanFile(handle,"%s>%i[x]%i[x]%i[x]%i[x]",
            &GDPM_Color[SetDPMPrefix],&GPrecn[SetDPMPrefix],
            &GDPMB_Color[SetDPMPrefix],&GDispUnit[SetDPMPrefix]);}}
   if(ScRDpm==DPMONLY){
      for(i=0;i<DPMCount;i++){
               ScanFile(handle,"%s>%i[x]",&DPM_ChNo[i]);SetDPMPrefix=DPM_ChNo[i];
         ScanFile(handle,"%s>%i[x]%i[x]%i[x]%i[x]",
            &GDPM_Color[SetDPMPrefix],&GPrecn[SetDPMPrefix],
            &GDPMB_Color[SetDPMPrefix],&GDispUnit[SetDPMPrefix]);}
      ScanFile(handle,"%s>%i[x]",&Dummy);
      for(i=0;i<SCCount;i++){ScanFile(handle,"%s>%i[x]",&SC_ChNo[i]);SetSCPrefix=SC_ChNo[i];
         ScanFile(handle,"%s>%i[x]%i[x]%i[x]%f[x]%f[x]",
            &GSCB_Color[SetSCPrefix],&GSCT_Color[SetSCPrefix],
            &GGrid[SetSCPrefix],&GYScaleMin[SetSCPrefix],
            &GYScaleMax[SetSCPrefix]);}}
   ScanFile(handle,"%s>%i[x]%i[x]%s[xt44]%i[x]%i[x]%s[xt44]",
            &Format,&FileBuffer,Specimen,&BufferType,&GrpPrecn,GrpFName);}
        GrpFlag=TRUE;FGFlag=TRUE;DGSCFlag=TRUE;DGDPMFlag=TRUE;ReLoad=TRUE;}
    CloseFile(handle);UnloadPanel(MessHandle);OpenHandle=LoadPanel("Ald.uir",Ald1);DisplayPanel(OpenHandle);
}

/*----------- Variables initialization center -----------*/

void    RestoreFlags()
{
    IFlag=0;GrpFlag=0;DIFlag=0;DGSCFlag=0;DGDPMFlag=0;FGFlag=0;FIFlag=0;FileChNo=0;check=0;
    power=0;ErrFlag=0;i=0;Setii=0;SetDii=0;SetFii=0;ii=0;iii=0;zoom=0;Sampling=0;DPMSwitch=1;
    SCSwitch=1;GrpFlag=0;SCCount=0;DPMCount=0;GrpCount=0;ChGain=1;FileEntry=0;UnitTime=0.0;
    StopFlag=0;ReLoad=0;NoDPMDisp=FALSE;NoSCDisp=FALSE;FndDPM=FALSE;FndSC=FALSE;Dii=0;
    id=0;
    for(i=0;i<MAXCH;i++){UTime[i]=0;GrpPageNo[i]=0;GDPM_Color[i]=0;GDPMB_Color[i]=0;
       GSCB_Color[i]=0;GSCT_Color[i]=0;GrpSample[i]=0;GrpScale[i]=0;GrpOffset[i]=0;
       GrpTDelay[i]=0;GrpChSD[i]=0;GrpGain[i]=0;GrpUnit[i]=0;GYScaleMin[i]=0;
       GYScaleMax[i]=0;GGrid[i]=0;SC_ChNo[i]=0;DPM_ChNo[i]=0;}
    FillBytes(GrpChString,0,50,' ');FillBytes(ConfigFName,0,15,' ');FillBytes(AldSCTitle,0,50,' ');
}

/*-------------- Sets the Digital Panel Meter display color ----------------*/
void    DPMColorFn()
{
    if(!zoom)SetCtrlAttribute(IndivHandle,AldIndiv_Digital,4,DPM_Color);
    else SetCtrlAttribute(ZoomHandle,AldZoom_Digital,4,DPM_Color);
}

/*-------------- Sets the Digital Panel Meter background color -------------*/
void    DPMBColorFn()
{
    if(!zoom)SetCtrlAttribute(IndivHandle,AldIndiv_Digital,6,DPMB_Color);
    else SetCtrlAttribute(ZoomHandle,AldZoom_Digital,6,DPMB_Color);
}

*/--------------- Sets the Strip Chart Trace color -----------------------*/
void    SCTColorFn()
{
    if(!zoom)SetTraceAttribute(IndivHandle,AldIndiv_StripChart,1,0,SCT_Color);
    else SetTraceAttribute(ZoomHandle,AldZoom_StripChart,1,0,SCT_Color);
}

/* -------------- Sets the Strip chart background color -------------------*/
void    SCBColorFn()
{
    if(!zoom)SetGraphAttribute(IndivHandle,AldIndiv_StripChart,5,SCB_Color);
    else SetGraphAttribute(ZoomHandle,AldZoom_StripChart,5,SCB_Color);
}
```

```
/*---------------- Display parameters default setting ---------------------*/
void   SetDispParam()
{
   DPM_Color=14;Precn=2;DPMB_Color=0;DispUnit=1;SCB_Color=0;SCT_Color=9;
   Grid=1;YScaleMin=-10.0;YScaleMax=10.0;Fmt(YAxisName,"%s<Engineering Units");
   FmtFile(InfoChHndl[ChNo],"%s<0,%i,%i,%i,%i,%i,%i,%i,%f,%f,%s,",
      DPM_Color,Precn,DPMB_Color,DispUnit,SCB_Color,SCT_Color,Grid,YScaleMin,YScaleMax,YAxisName);
}

/*---------------- Individual channel parameter check list ---------------------*/
void   ConfirmIndParam()
{
   IndParamHandle=LoadPanel("help.uir",IndParam);DisplayPanel(IndParamHandle);
   SetInputMode(OpenMenuBar,OpenMenu_Go,0);SetInputMode(OpenMenuBar,OpenMenu_Setup,0);
   SetInputMode(OpenMenuBar,OpenMenu_File,0);SetInputMode(OpenMenuBar,OpenMenu_Help,0);
   ActualSampling=1000/Sampling;
   Fmt(&Frequency,"%i<%f",ActualSampling);Fmt(SampleString,"%s<%i Hz",Frequency);
   if(Unit1)Fmt(UString,"%s<SI");else Fmt(UString,"%s<English");
   if(IndDispType)Fmt(U1String,"%s<Voltage");else Fmt(U1String,"%s<Engineering");
   if(Format)Fmt(FString,"%s<Binary");else Fmt(FString,"%s<ASCII");
   if(FileBuffType)Fmt(BString,"%s<Double");else Fmt(BString,"%s<Single");
   SetCtrlVal(IndParamHandle,IndParam_l1,ChName);SetCtrlVal(IndParamHandle,IndParam_l2,ChNo);
   SetCtrlVal(IndParamHandle,IndParam_l3,SampleString);SetCtrlVal(IndParamHandle,IndParam_l4,IndChType);
   SetCtrlVal(IndParamHandle,IndParam_l5,IndGain);SetCtrlVal(IndParamHandle,IndParam_l6,TDelay);
   SetCtrlVal(IndParamHandle,IndParam_l7,Scale);SetCtrlVal(IndParamHandle,IndParam_l8,UString);
   SetCtrlVal(IndParamHandle,IndParam_l9,Offset);SetCtrlVal(IndParamHandle,IndParam_l10,U1String);
   if(!IndDispType)SetCtrlVal(IndParamHandle,IndParam_l11,UnitName);
   SetCtrlVal(IndParamHandle,IndParam_l12,ChSD);
   SetCtrlVal(IndParamHandle,IndParam_l13,FString);SetCtrlVal(IndParamHandle,IndParam_l14,FileBuffer);
   SetCtrlVal(IndParamHandle,IndParam_l15,BString);SetCtrlVal(IndParamHandle,IndParam_l16,FileName);
   SetCtrlVal(IndParamHandle,IndParam_l17,Specimen);SetCtrlVal(IndParamHandle,IndParam_l18,YAxisName);
   SetCtrlVal(IndParamHandle,IndParam_l19,Precn);
   GetUserEvent(TRUE,&handle,&id);
   switch(id){
      case IndParam_Yes :
         IndivSensor();iii=FALSE;break;
      case IndParam_No :
         UnloadPanel(-1);OpenHandle=LoadPanel("Ald.uir",Ald1);iii=FALSE;
         RestoreFlags();
         DisplayPanel(OpenHandle);OpenMenuBar=LoadMenuBar("Ald.uir",OpenMenu);break;
      default :break;}
}

/*---------------- Group channel panel meter illumination routine ----------*/
void   GrpPowerOn(SCCnt,DPMCnt)
int DPMCnt,SCCnt;
{
   SetInputMode(GroupHandle,AldGroup_Label,1);SetCtrlVal(GroupHandle,AldGroup_PLed,1);
   SetInputMode(GroupHandle,AldGroup_Run,1);SetInputMode(GroupHandle,AldGroup_GrpStamp,1);
   SetInputMode(GroupHandle,AldGroup_Stop,1);SetInputMode(GroupHandle,AldGroup_Date,1);
   SetInputMode(GroupHandle,AldGroup_Realdate,1);SetInputMode(GroupHandle,AldGroup_STime,1);
   SetInputMode(GroupHandle,AldGroup_ActualSTime,1);SetInputMode(GroupHandle,AldGroup_ETime,1);
   SetInputMode(GroupHandle,AldGroup_ActualETime,1);
```

```
    for(i=0;i<SCCnt;i++) SetInputMode(GroupHandle,SCFCONST+i,1);
    for(i=0;i<DPMCnt;i++){
       SetInputMode(GroupHandle,DPMCONST+i,1);SetInputMode(GroupHandle,DPMFCONST+i,1);}
    Fmt(daystring,datestr());SetCtrlVal(GroupHandle,AldGroup_Realdate,daystring);
}

/*------------------ Group channel panel meter power off routine ------------*/
void    GrpPowerOff()
{
    SetInputMode(GroupHandle,AldGroup_Label,0);SetCtrlVal(GroupHandle,AldGroup_PLed,0);
    SetInputMode(GroupHandle,AldGroup_Run,0);SetInputMode(GroupHandle,AldGroup_GrpStamp,0);
    SetInputMode(GroupHandle,AldGroup_Stop,0);
    SetInputMode(GroupHandle,AldGroup_Date,0);SetInputMode(GroupHandle,AldGroup_Realdate,0);
    SetInputMode(GroupHandle,AldGroup_STime,0);SetInputMode(GroupHandle,AldGroup_ActualSTime,0);
    SetInputMode(GroupHandle,AldGroup_ETime,0);SetInputMode(GroupHandle,AldGroup_ActualETime,0);
    for(i=0;i<MAXDISPLAY;i++){SetInputMode(GroupHandle,DPMCONST+i,0);
       SetInputMode(GroupHandle,SCFCONST+i,0);SetInputMode(GroupHandle,DPMFCONST+i,0);}
}

/*---------------- Individual channel panel meter illumination routine ----------*/
void    PowerOnStat()
{
    SetCtrlVal(IndivHandle,AldIndiv_LPLed,1);SetInputMode(IndivHandle,AldIndiv_Run,1);
    SetInputMode(IndivHandle,AldIndiv_Stamp,1);SetInputMode(IndivHandle,AldIndiv_Stop,1);
    SetInputMode(IndivHandle,AldIndiv_Digital,1);SetInputMode(IndivHandle,AldIndiv_SamText,1);
    SetInputMode(IndivHandle,AldIndiv_EnggText,1);SetInputMode(IndivHandle,AldIndiv_VoltText,1);
    SetInputMode(IndivHandle,AldIndiv_Volt,1);SetInputMode(IndivHandle,AldIndiv_YScaleMin,1);
    SetInputMode(IndivHandle,AldIndiv_YScaleMax,1);SetInputMode(IndivHandle,AldIndiv_Sample,1);
    SetInputMode(IndivHandle,AldIndiv_STime,1);SetInputMode(IndivHandle,AldIndiv_Date,1);
    SetInputMode(IndivHandle,AldIndiv_ETime,1);SetInputMode(IndivHandle,AldIndiv_Realdate,1);
    SetInputMode(IndivHandle,AldIndiv_ActualSTime,1);SetInputMode(IndivHandle,AldIndiv_ActualETime,1);
    Fmt(daystring, datestr());SetCtrlVal(IndivHandle,AldIndiv_Realdate, daystring);
    SetInputMode(IndivHandle,AldIndiv_PanelLabel,1);SetInputMode(IndivHandle,AldIndiv_Sample,1);
}

/*------------------ Individual channel panel meter power off routine ------------*/
void    PowerOffStat()
{
    SetCtrlVal(IndivHandle,AldIndiv_LPLed,0);SetInputMode(IndivHandle,AldIndiv_SamText,0);
    SetInputMode(IndivHandle,AldIndiv_Stamp,0);SetInputMode(IndivHandle,AldIndiv_Run,0);
    SetInputMode(IndivHandle,AldIndiv_Stop,0);SetInputMode(IndivHandle,AldIndiv_Digital,0);
    SetInputMode(IndivHandle,AldIndiv_EnggText,0);SetInputMode(IndivHandle,AldIndiv_VoltText,0);
    SetInputMode(IndivHandle,AldIndiv_Volt,0);SetInputMode(IndivHandle,AldIndiv_YScaleMin,0);
    SetInputMode(IndivHandle,AldIndiv_YScaleMax,0);SetInputMode(IndivHandle,AldIndiv_Sample,0);
    SetInputMode(IndivHandle,AldIndiv_ETime,0);SetInputMode(IndivHandle,AldIndiv_Stamp,0);
    SetInputMode(IndivHandle,AldIndiv_Date,0);SetInputMode(IndivHandle,AldIndiv_STime,0);
    SetInputMode(IndivHandle,AldIndiv_Realdate,0);SetInputMode(IndivHandle,AldIndiv_ActualSTime,0);
    SetInputMode(IndivHandle,AldIndiv_ActualETime,0);SetInputMode(IndivHandle,AldIndiv_Sample,0);
    SetInputMode(IndivHandle,AldIndiv_PanelLabel,0);
}
```

GLOSSARY OF ITEMS

Microsoft, MS, and MS-DOS are registered trademarks of Microsoft Corporation. IBM is the registered trademarks of International Business Machines Corporation. Intel is the registered trademarks of Intel Corporation. LabWindows is the registered trademarks of National Instruments Corporation. DAP, Microstar Labs is the registered trademarks of Microstar Laboratories, Inc.