User Manual

Tektronix

TekTMS
Front Panel Editor
S3FT120

070-8502-01

This document supports **software** version *2.5*.

Please check for change information at the rear of this manual.

First Printing: October 1992

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

Printed in U.S.A.

SOFTWARE WARRANTY SUMMARY

Tektronix warrants that its software products will conform to the specifications in the documentation provided with the product, when used properly in the specified operating environment, for a period of three (3) months. The warranty period begins on the date of shipment, except that if the program is installed by Tektronix, the warranty period begins on the date of installation or one month after the date of shipment, whichever is earlier. If the software product does not conform as warranted, Tektronix will provide the remedial services as described in the documentation provided with the product.

For products offered without documentation, Tektronix warrants that the media on which the software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If any such medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which the software product is furnished, the software product is provided "as is" without warranty of any kind, either express or implied.

Tektronix does not warrant that the functions contained in any software product will meet Customer's requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and, for warranted products, make suitable arrangements for such service in accordance with the instructions received from Tektronix. If Tektronix is unable, within a reasonable time after receipt of such notice, to provide remedial service for warranted products or, for "as is" products, to provide a replacement that is free from defects in materials and workmanship, Customer may terminate the license for the software product and return the software product and any associated materials for credit or refund.

The above warranties shall not apply to any software product that has been modified or altered by Customer. Tektronix shall not be obligated to furnish service under this warranty with respect to any software product a) that is used in an operating environment other than that specified or in a manner inconsistent with the User Manual and documentation; or b) when the software product has been integrated with other software if the result of such integration increases the time or difficulty of analyzing or servicing the software product or the problems ascribed in the software product.

THE ABOVE WARRANTIES ARE GIVEN BY TEKTRONIX WITH RESPECT TO THE LISTED PRODUCTS IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO PROVIDE REMEDIAL SERVICE WHEN SPECIFIED, REPLACE DEFECTIVE MEDIA, OR REFUND CUSTOMER'S PAYMENT, AS APPLICABLE, IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO CUSTOMER FOR BREACH OF EITHER WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Preface

Purpose of this Manual

The objective of this manual is to explain how to use the Front Panel Editor (FPE) to develop instrument drivers (.ISD files or scripts). Instrument drivers define front panel displays for the Interactive Procedure Generator (IPG). These front panel displays are the instrument controls you use during the creation of IPG test procedures.

An ISD script performs two basic functions:

- Defines the graphical appearance of a logical instrument view for an instrument. This instrument view provides a means for user interaction with the instrument.
- Defines how each control in the logical instrument view interacts with the hardware (i.e., instruments on a bus, and the bus itself).

About this Manual

This manual contains these sections and appendices:

- Section 1, General Information — Provides installation instructions and an overview of the TekTMS/FPE system.
- Section 2, Introduction to the Front Panel Editor — Describes how to use the FPE for creating instrument control panels.
- Section 3, TekTMS/FPE Programming Environment — Provides a complete guide to the extremely versatile and convenient programming made possible by the TekTMS Front Panel Editor.
- Section 4, FPE Tutorial — This is a hands-on introduction to the Front Panel Editor.
- Section 5, TekTMS/FPE Procedures — This section explains the front panel editing processes available in the Front Panel Editor.
- Section 6, Instrument Script Language Descriptions — Defines and explains the ISD language and its usage in creating instrument front panels.
- Appendix A, *Glossary* — A guide to the specialized terms in the FPE.
- Appendix B, *TDM5120.ISD* Script Printout — An example of an *.ISD* file.
- Appendix C, Software Performance Report — This appendix provides a form for submitting problems encountered while using TekTMS/RTG.

Index

Contents

Section 1
General Information

Section 2
Introduction to the Front Panel Editor

Section 3
The TekTMS/FPE Programming Environment

Section 4
FPE Tutorial

Section 5
TekTMS/FPE Procedures

Section 6
Instrument Script Language Descriptions

Appendices

Index

Change Information

List of Illustrations

List of Tables

General Information

This section describes the system requirements and installation of FPE. It also provides a good overview of FPE.

User Requirements

Developing scripts does not require extensive programming skills, but some knowledge of programming is desirable. You should have a working knowledge of PC DOS and Microsoft Windows@.

It will be helpful to refer to the Interactive Procedure Generator User Manual for instructions on how you will use the scripts you write to develop test procedures.

System Requirements

Required Hardware

- Tektronix VXI System Controller, or
- Tektronix system controller, or
- IBM PC-AT@compatible with the following:
  - at least 2 MByte RAM
  - one 1.2 MByte or 1.44 MByte floppy disk drive (5¼″ or 3½″ drives)
  - 40 MByte hard drive
  - an EGA or VGA compatible graphics adapter and display
  - a Microsoft Mouse@or a compatible MS Windows@mouse

Required Software

- MS-DOS Version 3.0 or later
- a Microsoft Windows Version 3.1 or later
- TekTMS/FPE Version 2.5 or later

Installing TekTMS/FPE

TekTMS/FPE is quite easy to install with the supplied INSTALL Windows application. As you select from options during the INSTALL program, Help information will appear that describes the option. When prompted, any C include files should be installed into your existing C library, which is the default for INSTALL. When prompted, you should elect to overwrite exiting files, except for ISD files. Install the new ISD files in a new directory. You may need the old ISD files to run your existing test scripts.

NOTE

For information about installed files and additional product requirements, refer to the READ.*100* files on the product *disk(s).* Before installing *TekTMS/RTG* review, the system requirements on page *1-1.*

To install the software from within the Windows Program Manager, perform the following:

1. Insert product disk number 1 into an appropriate disk drive.
2. In the **File** menu. select **Run.**
3. In the dialog box, enter the drive designator, a colon (:), and the command INSTALL. If the product disk is in drive A you would enter

   ```
   A:INSTALL
   ```

4. The install program will prompt you for the installation location and give you status as the installation proceeds.

To install the software from the DOS prompt, perform the following:

1. Insert product disk number 1 into an appropriate disk drive.
2. At the DOS prompt, enter the drive designator, a colon (:), and the command INSTALL. If the product disk is in drive A you would enter

   ```
   A:INSTALL
   ```

   The INSTALL program will launch Windows and then proceed with the installation.
3. The INSTALL program will prompt you for the installation location and give you status as the installation proceeds.

Overview

Scripts serve two purposes: First, they provide an interactive, graphical, instrument front panel view for user interaction with the instrument while using the Interactive Procedure Generator (IPG). Second, they control the flow of information between a test program and the instrument when running an IPG test procedure.

Scripts allow you to add support for any instrument. You can also have multiple scripts for the same instrument.

With a script you can implement a subset of the instrument functionality, if desired. This approach is useful if you are using only some of the instrument functions, or if you desire a simplified approach.

Also, any number of identical instruments can share a script. With IPG you can call scripts for use as you generate test procedures.

Front panel views provided by scripts have two general uses: Interactive and Program Generation (see Figure 1-1). We can further classify front panel views into two general implementations: fully functional and application Oriented.

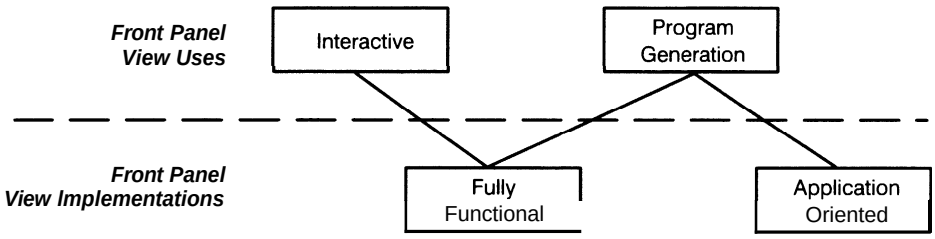


Figure 1-1: Types of Front Panel Views

Interactive Views

Interactive front panel views are for interactive instrument control rather than for program generation. Interactive front panel views are fully functional implementations that contain a complete, or nearly complete, set of instrument controls and/or functions.

Program Generation Views

Program Generation front panel views are generally intended to help you create automated test procedures that can run unattended. These front panel views are fully functional or application oriented implementations, depending on the requirements of the test.

Fully functional implementations support the full range of an instrument's control functions.

Application oriented implementations are tailored, customized, or optimized scripts, that contain only those controls needed for a particular test situation. These scripts also may contain composite controls that combine several

instrument functions into one application-oriented, high-level control function. There are several advantages to using an Application Oriented front panel:

- With fewer controls, it is easier for the program developer or test operator to select a control for a particular application.

  Using only relevant controls reduces the overall size of the view, which uses less memory space, thus allowing more drivers to be stored on the system.

- When implementing only a subset of controls, it is easier to make instrument substitutions because instruments are more likely to share a common subset of controls, than total functionality.

Front Panel View Optimization

Instrument front panel drivers may be optimized whether they are used interactively or for program generation.

As indicated earlier, interactive front panels should contain controls that correspond to actual instrument front panel controls, or in the case of VXI instruments, to specific instrument functions. These criteria closely fit the definition of a fully-functional front panel containing a complete set of the instrument's commonly used controls.

In other cases, the controls implemented for an interactive front panel may not be needed on a front panel optimized for program generation use. For example, it may be possible to adjust the frequency of a signal generator by pressing an INCrement or DECrement pushbutton on the instrument's front panel. While this function might be useful on an interactive front panel, it is not likely to be used in program generation, thus it probably should not be implemented on a program generation front panel.

If both extensive interactive control and program generation are required, create two instrument front panel views with each optimized for its primary use. For instruments with hardware front panels, a program generation front panel view or set of program generation front panel views may be sufficient. For instruments, such as VXI instruments that do not have hardware front panels, it may be desirable to have both program generation and interactive front panels. It also may be desirable to have an interactive front panel when a test operator is expected to use an instrument to troubleshoot a failed unit-under-test (UUT).

Setting, Learn, and Measurement Control Functions

NOTE

In IPG, the keywords Query, Measure, and Measurement are synonymous and fully interchangeable. Any one of them can be used in a Learn or Measurement block to query an instrument control for its current setting or measured value.

An interactive front panel driver should implement the Learn block for each instrument, and the Setting and Measurement blocks for every control. The Learn block allows the user to query the instrument for its current settings, store them in a procedure step, then recall them on demand to restore the instrument to those settings. The inclusion of the Measurement block provides a means for querying the instrument for a control's current setting or measured value, which can be used to update the interactive front panel display (Also see Updating Features, following). On instruments that do not support a query function (Learn or Measurement block) for their controls, a current status query cannot be implemented for updating the front panel display.

NOTE

The instrument help message will tell you whether or not the instrument has a Learn block.

In a program generation front panel view, query functions should be implemented only to acquire a measured value. By eliminating all other query (or measure/measurement) blocks, the driver can be smaller in size, thus minimizing overhead at execution time and allowing more drivers to be loaded into memory.

Actual Versus Composite Controls

On an interactive front panel view, a control is more likely to correspond directly to an instrument control or command than it is on a program generation front panel. For example, a digital multimeter will typically have controls for measuring voltage and current. On an interactive front panel, these controls would be implemented as two separate displays, one for voltage and one for current. Each display would correspond to an actual instrument function. However, if test procedures are to be developed for an application-oriented, program generation front panel where a power measurement is required, the view should contain a control for displaying power rather than two displays for current and voltage. The script for this control would query the instrument for its current and voltage readings, calculate the power, and display it on a one front panel power control. In this case, there would be only one control for displaying power, which would not correspond to an actual instrument control or function.

A program generation front panel will often contain controls that do not correspond to controls on the actual instrument front panel. If a function that involves several instrument controls is used frequently in a test situation, this function could be created as a single multifunction control on the front panel. As indicated in the previous paragraph, calculations could be incorporated into the control functions. For example, several readings might be taken and averaged with only the averaged result displayed in a control on the front panel. In this case, only one control needs to be selected by the test procedure step to obtain the averaged result.

Updating Features

You can optimize an interactive front panel to use the front panel display update features such as automatic refresh, driver control update lists, and the IPG **Update!** command.

Front Panel Appearance

The appearance of front panel displays may be enhanced by using various optimizing techniques. For example, a data display might be enhanced by converting its reading from engineering notation (5E-3) to standard suffix units notation (5 mV). This technique should be used only on interactive front panel views.

On a program generation front panel, interactive features, display enhancements, and the total number of controls may be minimized to make the panel smaller and easier to use, and to provide the minimum overhead at execution time.

Introduction to the Front Panel Editor

What is TekTMS/FPE?

TekTMS/FPE is an acronym for the Tektronix Test Management System/Front Panel Editor (FPE). The FPE is a Microsoft Windows application for creating instrument front panels for programmable instruments. The FPE generates Instrument Software Definition (.ISD) files that are used by TekTMS/IPG, the Interactive Procedure Generator.

TekTMS/FPE Features

Windows Application — TekTMS/FPE uses the Microsoft Windows user interface for pull-down menus, dialog boxes, resizeable and movable windows. Microsoft Windows applications provide system independence — if the system will run Windows, TekTMS/FPE will run, using the support of the Windows system.

Graphical View — As the user creates and edits each control selected from a menu list, the position and size of the controls are visible at the time of creation. Dialog boxes are used to set the control parameters and edit windows are used to edit code for each control.

Instrument Script Language — Users can develop new instrument front panels or modify existing ones with the FPE. This includes both the VXI Front Panel Libraries and the GPIB Front Panel Libraries.

Online Help — TekTMS/FPE provides online help topics covering most of the FPE menu operations and procedures in a hypertext-like format.

Mouse Terms

The following mouse-specific terms are used in this manual:

Point	Move the mouse until the pointer rests on what you want to point to.
Press	Hold down the mouse button.
Click	Quickly press and release the mouse button. Click refers to pressing the left button unless otherwise indicated.
Drag	Hold down the left mouse button while moving the mouse across a flat surface, such as the desktop.
Double Click	Click the left mouse button twice in rapid succession.

The Front Panel Screen

Figure 2-1 shows the Front Panel Screen with its key parts labelled.

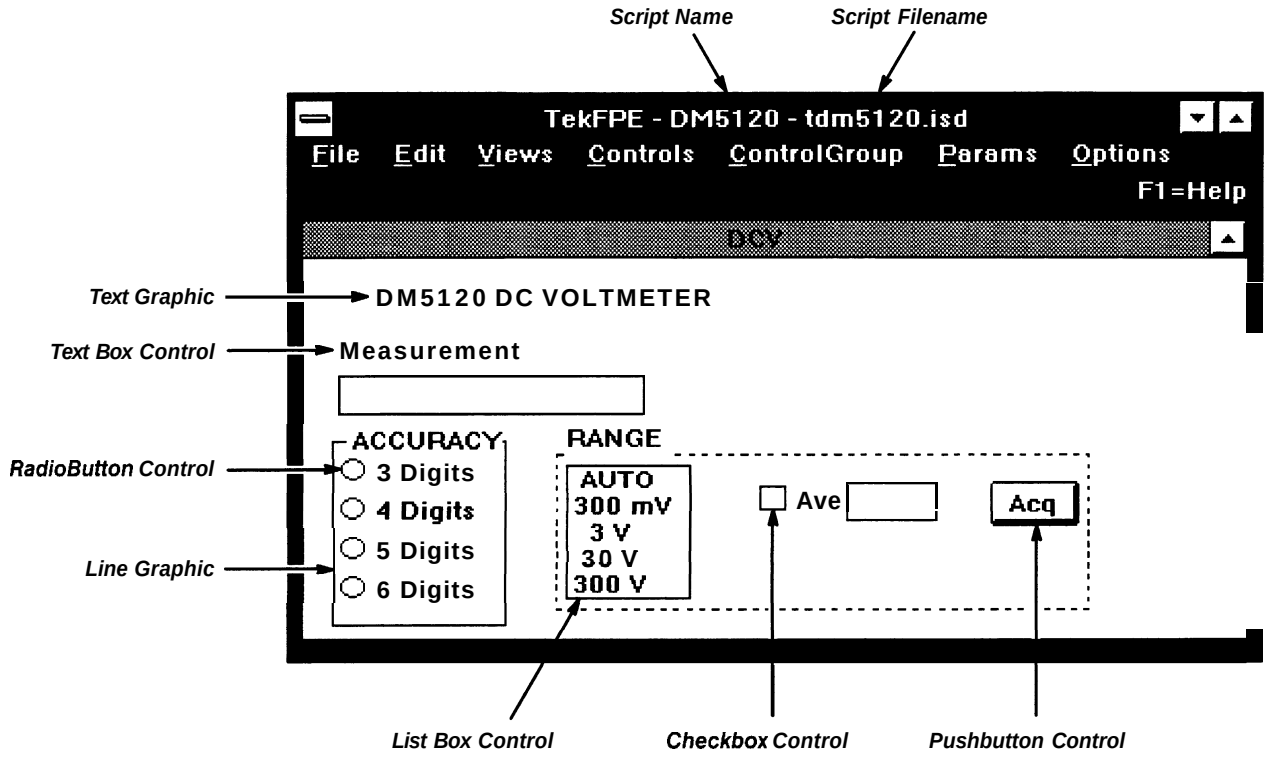


Figure 2-1: Example of an FPE Screen

Using the TekTMS/FPE Menus

FPE commands are organized in menus on the menu bar. Figure 2-2 shows a sample menu.

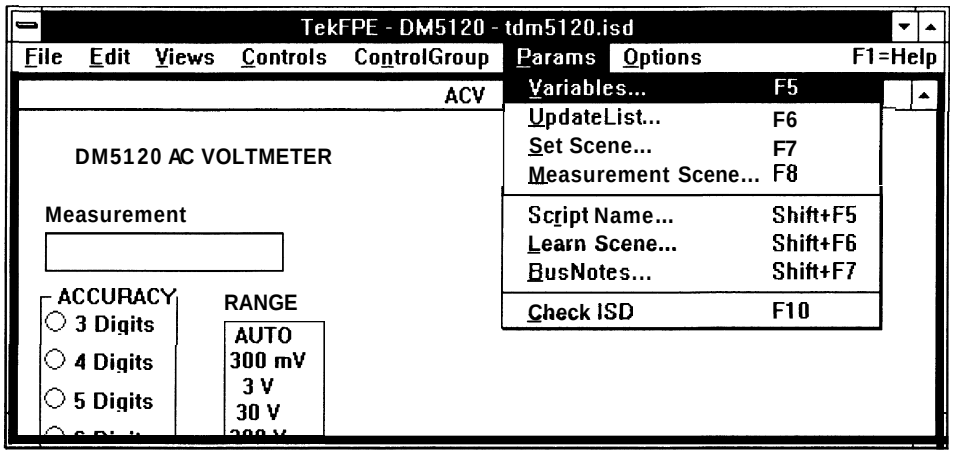


Figure 2-2: FPE Menu Example

Commands followed by dots (...) prompt you for more information when you select them. Commands without dots are executed as soon as you select them.

Contents of the FPE Menus

This list shows the types of commands and options found in the FPE menus:

File	Create, open, and save ISD files.
Edit	Copy, paste, delete, and duplicate controls. The Edit commands also include commands to align controls within a view.
Views	Create, delete, rename, and select views
Controls	List of valid controls for a front panel
ControlGroup	The commands to make, break, and show control groups.
Params	Commands to open dialog boxes for variable settings, updatelist, settings and measure scene editing, learn scene editing, scriptname and header remarks editing, busnotes and check ISD.
Options	Open Notepad, Start IPG, Scroll bars, Save Config, and invoke the toolbox window.

Choosing Commands from Menus

In the Windows environment, there are several ways to display a menu and select a command. These are discussed here.

Keyboard and Underlined Letter

To select a command using the keyboard keys, perform these steps:

4. Press the ALT key plus the keyboard key for the underlined letter in the menu name. This opens the menu.
5. Press the key corresponding to the underlined letter of the command name. If the command you want is already highlighted, press ENTER to select it.

Keyboard with Direction Keys

Another way to select commands using keyboard keys is to use the arrow direction keys in combination with the ALT key:

1. Press ALT.
2. Move the highlight from command to command using the arrow direction keys.
3. Press ENTER to carry out the highlighted command.

Mouse Only

To select commands using only the mouse, click on the menu name with the left mouse button, then chose the command you want by clicking on it with the left mouse button.

To cancel a command with the mouse, click on anything except a menu with the left mouse button.

Shortcut Keys for Commands

The shortcut key names, or key sequences, for commands appear to their right, as shown in Figure 2-3.

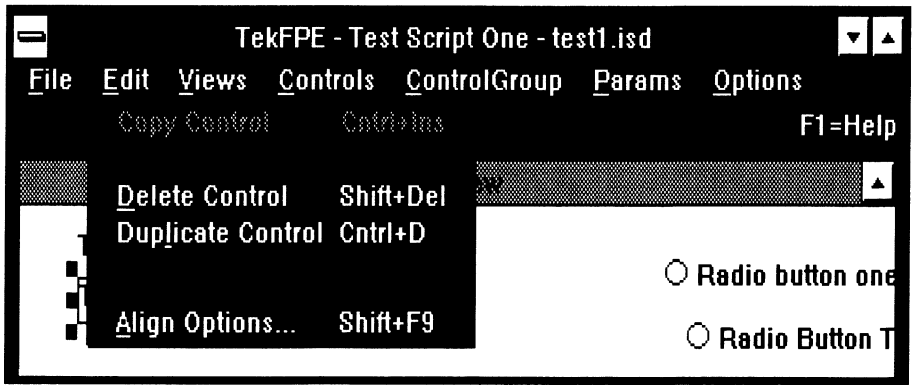


Figure 2-3: Example of a Menu With Shortcut Keys

The shortcut keys (also known as quick access or speed keys) method for selecting and sending commands is quicker because several steps are eliminated. You do not need to first open the menu, and then select the command you want to use. For example, to issue the Align Options command with shortcut keys, simply press the Shift and the F9 keys simultaneously. This action opens the **Align Options** dialog box.

Table 2-1 presents a list of the FPE shortcut key commands available in TekTMS/FPE.

Table 2-1: FPE Shortcut Keys

Menu	Command	Key(s)
File	New Script	Cntrl+N
	Open	Cntrl+F12
	Save	Shift+F12
	SaveAs	F12
	Exit	Alt+F4
Edit	Copy Control	Cntrl+Ins
	Paste Control	Shift+Ins
	Delete Control	Shift+Del
	Duplicate Control	Cntrl+D
	Align	F9
	Align Options	Shift + F9

Table 2-1: FPE Shortcut Keys (Cont.)

Menu	Command	Key(s)
Views	NewView	Shift + F11
	Rename View	F11
	Delete View	Cntrl+F11
	Next View	PgDn/PgUp
ControlGroup	Break ControlGroup	Shift + F2
	Make ControlGroup	F2
	Show ControlGroup	F3
Params	Variables	F5
	Updatelist	F6
	Set Scene	F7
	Measurement Scene	F8
	Script Name	Shift+F5
	Learn Scene	Shift+F6
	BusNotes	Shift+F7
	Check ISD	F10
Options	Scroll Bars	F4
	Save Config	Cntrl+F
	Toolbox	Cntrl+T

Using Dialog Boxes

FPE displays a dialog box when additional information is required or when confirmation of an action is required. The dialog box contains areas where you enter information, select options to commands, or activate controls to execute or cancel a command. While a dialog box is displayed, any input you enter from the keyboard affects the item in the dialog box that has the "input focus". The item with the input focus is highlighted to show its selection.

Many of the items in the dialog box are options that you can turn on or off by clicking the mouse over the option item. Some dialog box items require textual input.

The FPE dialog boxes contain some or all of the following elements.

Edit Box

An edit box is a box that provides an area to enter information. The edit box must have the "focus" in order to receive the keyboard input. The focus is indicated by a flashing cursor inside the edit box. If you type more characters than can fit in the box, the text scrolls, allowing more text — this is true for most edit boxes. Some edit boxes do not allow more than a fixed number

of characters, such as the Variable Name edit box, which has a 15 character maximum due to an ISD limit. If you attempt to enter more characters, they will not be added to the input. Edit boxes are gray if the edit function is not available.

List Box

This is a box that presents a list of selectable items. Select the item you want from the list by clicking on it. Most list boxes allow only one selection. A double click on a list box item does the same thing as a single click selection, followed by an **Ok** command selection. The list box may be gray if the selection is not available.

Check Box

A check box is a box with a label to the right. The purpose of a check box is to turn an option on or off. **On** is shown by an X in the control box, **Off** by an empty box. The check box is gray if the selection is not available.

PushButtons

A command button is a box with a label inside, and referred to as a PushButton. The action a command button produces occurs immediately after you click the button. A command button is gray when its command is not available.

RadioButtons

A radiobutton is a circle, with a label to the right of the circle. RadioButtons provide a mutually exclusive selection, in that only one of a group of items is selectable. Selection is indicated by a filled-in circle. A radiobutton is gray if the selections it offers are not available.

The TekTMS/FPE Programming Environment

Introduction To Menu Commands

This section describes how you can use the TekTMS/FPE environment to create and edit .ISD script files. This section shows how to use TekTMS/FPE menu commands to:

- Create a new .ISD file that can be utilized by TekTMS as the front panel to generate test steps for a TekTMS test script.
- Edit an existing .ISD script file to customize the front panel to the user's specifications.
- Insure TekTMS front panel compatibility.

File Menu Operations

Table 3-1: The File Menu Commands

Command	Function
New Script	Create and edit scripts
Open	Create and edit scripts
Save	Save front panels to ISD files
SaveAs	Save front panels to ISD files
Exit	End the FPE session
About	Identify the installed application and version

New Script Command

The **New Script** command opens a new FPE edit session. If you select the **New Script** command while editing a script, a **Save** dialog box appears, prompting you to save or not save the current script to a file. After the **Save** dialog box, FPE displays the new script dialog box shown in Figure 3-1.

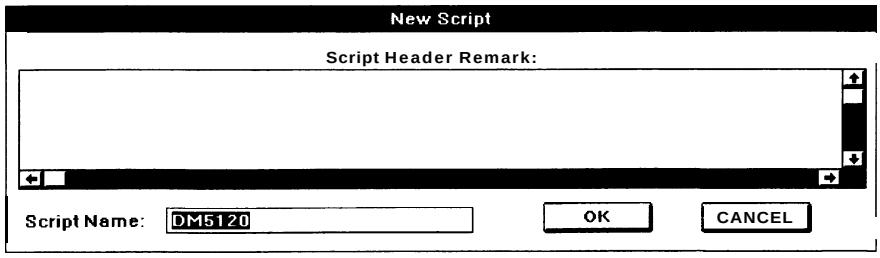


Figure 3-1: The New Script Dialog Box

Script Header Remark — This is a multiple line edit box for entering information about the script file you are creating. Enter any information you want to save with the script file. FPE saves the information at the head of the ISD file as a block of remarks. Examples of information commonly stored with the script file are descriptions of file contents, author names, and version information.

Script Name — Enter the name of the script you are creating in this edit box. The script name length is limited to 32 alphanumeric characters.

Ok — Select the **Ok** command to accept any information you have entered in the **Script Header Remarks** and **Script Name** dialog boxes. The FPE closes the **New Script** dialog box and presents the **New View Name** dialog box shown in Figure 3-2.

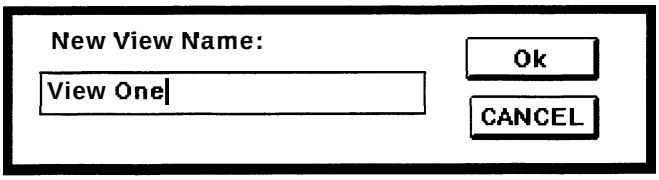


Figure 3-2: New View Name Dialog Box

Cancel — Select the **Cancel** command at any time when the **New Script** dialog box is open to cancel the New Script operation. Any information entered in the **Script Header Remarks** and **Script Name** dialog boxes will not be saved.

NOTE

Dialog boxes, edit boxes, and list boxes are explained in Section 2, Introduction to the Front Panel Editor

New View Name — Enter the view name for the new view you are creating.

Ok — The **Ok** command accepts the panel name and allows an empty panel to be drawn on the screen. The FPE main window title contains the script name. The new panel window title is the panel name entered in the **New View Name** dialog box.

Cancel — The **Cancel** command cancels the panel name and returns you to the FPE menu. Only the script name and remarks are changed at this point.

Open Command

The **Open** command reads in an existing .ISD file from the selected drive disk and opens an editing session, displaying an instrument front panel that is ready for editing. When you select the **Open** command, a dialog box appears like that in Figure 3-3.

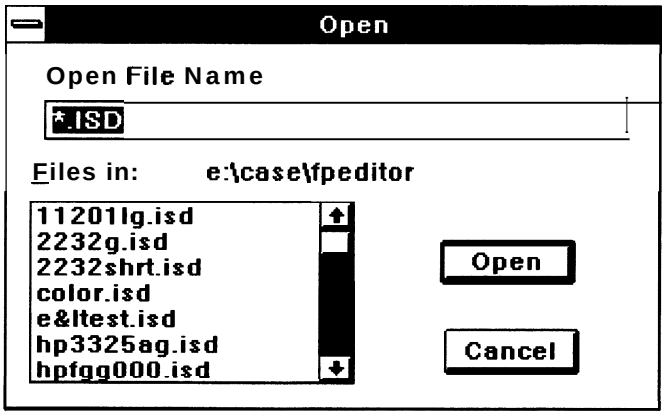


Figure 3-3: The Open Dialog Box

Open File Name — Enter the file name of the stored .ISD file you wish to retrieve in this edit box. You can also specify a subdirectory name in this edit box and thereby change to another subdirectory. Enter the subdirectory name (and its path, if the subdirectory is on another drive) in this edit box. By default, the FPE lists all the file names in the current directory that have the .ISD extension.

To limit the number of file names that appear in the **Files in** list box, use a "wild card" entry to specify a particular group of files. For example, to list only files having the ".ISD" extension, enter **.ISD* as shown here.

Open File Name

*.ISD

To list only the files in the "C:\ISD" subdirectory that start with T and have an extension that ends in D, enter C:\ISD\T*.??D as shown here.

Open File Name

C:\ISD\T*.??D

Next, press RETURN on the keyboard to list the appropriate file names in the list box. The FPE default is set to show all files in the current directory with the ".ISD" extension.

Files in [path] — list box displays the file names in the current directory, or a modified list if you use wildcards to specify a narrower range in the **Open File Name** edit box. Highlight the desired file, drive, or subdirectory in the list and select the **Ok** command to indicate your selection. An alternate way to indicate your selection is to double click on the item you want in the list.

Open — loads the selected .ISD file. If the file name is a valid file name the file will be opened; otherwise, an error message will state that the file could not be found.

Cancel — exits from the **Open** command and return to the main menu.

Save Command

The **Save** command saves the current editor session to an .ISD file. The file name will be the same as when it was opened unless you specify otherwise. When you select the **Save** command, FPE displays the message box shown in Figure 3-4.

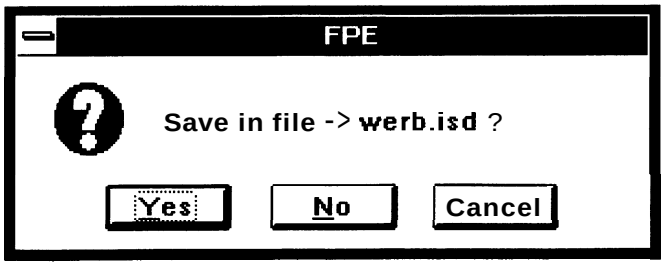


Figure 3-4: The Save Message Box

Save in file -> filename ? — This is a message box that shows the file name in which the current session will be saved. The prompt choices are **Yes, No,** and **Cancel.**

Yes — Select **Yes** to save the current session in the listed file name.

No — Select **No** to save the current session under another file name. If you select **No,** a **SaveAs** dialog box will appear; enter the new file name in the edit box.

Cancel — Select **Cancel** to exit from the **Save** command and return to the main menu.

SaveAs Command

The **SaveAs** command saves the file in a user selected file. When you select the **SaveAs** command, the dialog box in Figure 3-5 appears.

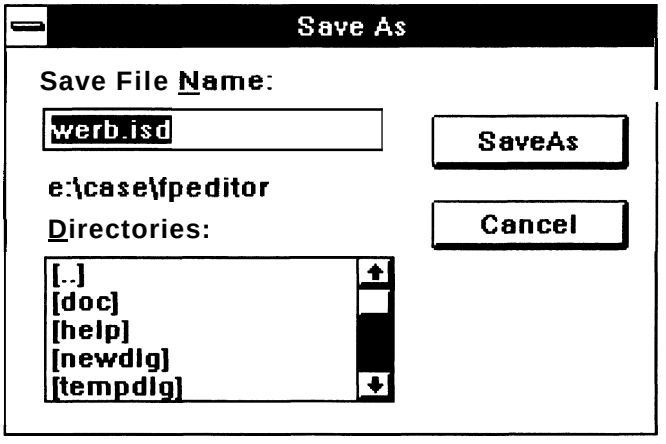


Figure 3-5: The SaveAs Dialog Box

Save File Name — Enter a file name in this edit box for the file in which you want to store your current session. Press RETURN to indicate your entry and the FPE will save the session.

Directory Path — The current directory drive and path appear above the **Directories** list.

Directories — The directories list shows the list of directories and drives on your system. You can select a new drive or directory by double clicking on a drive or directory name within the list.

SaveAs — The **SaveAs** command in the dialog box saves the current session in the named file in the directory shown.

Cancel — Select **Cancel** to exit from the **SaveAs** command and return to the main menu.

Exit Command

The **Exit** command closes the current edit session. If changes to the file have been made during the current session since a **Save** or **SaveAs** command has been issued, a message box appears, prompting you to save your work. When you select the **Exit** command, the message box in Figure 3-6 appears.

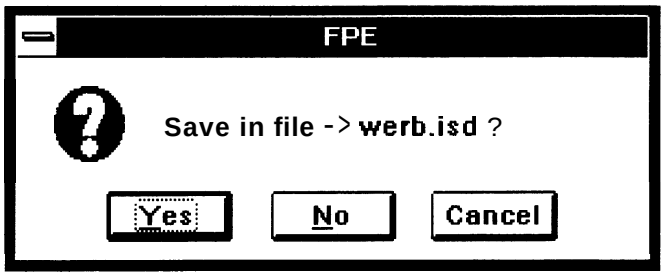


Figure 3-6: The Exit Message Box

Yes — Choose **Yes** to save the file in the listed file name.

No — Choose **No** to exit without saving edit session.

Cancel — Select the **Cancel** command to stop execution of the **Exit** command and return to the main menu.

About Command

The **About** command provides information about TekTMS/FPE. When the **About** command is selected, the dialog box in Figure 3-7 appears.

Figure 3-7: The About Dialog Box

Edit Menu Operations

Table 3-2 lists the Edit Menu commands. The commands are explained below.

Table 3-2: The Edit Menu Commands

Command	Function
Copy Control	Copies controls from view to clipboard
Paste Control	Pastes controls from the clipboard to the current view
Delete Control	Deletes controls from the current view
Duplicate Control	Duplicates control in the view
Align	Aligns controls in the view
Align Options	Provides access to alignment options for arranging controls

Copy Control

The Copy Control command copies one or more controls from the current view to the clipboard. Copy Control must be preceded by a group selection of one or more controls. (See the definition of group selection in Appendix A, Glossary.)

Paste Control

The Paste Control command copies one or more controls from the clipboard to the current view. Paste Control must be preceded by a copy control to clipboard. If there are controls in the clipboard, Paste Control is enabled. Use the mouse to move the cursor to the position where you want to place the controls and click the left mouse button.

Delete Control

The Delete Control command deletes one or more controls from the current view. Delete Control must be preceded by a group selection of one or more controls or a left button selection of a single control.

Duplicate Control

The Duplicate Control command duplicates one control. Duplicate Control must be preceded by a control select to choose the control to duplicate. Duplicate is enabled or disabled by selection of the Duplicate Control menu command, and is indicated by a check mark next to the command. When Duplicate Control is enabled, the editor is in the tool mode and a copy of the selected control is positioned when you press the left mouse button. Select Duplicate for each additional new control. The short-cut key sequence Cntrl+D also selects Duplicate.

Turn the **Duplicate** command off by selecting **Duplicate Control** when a check mark is present. An alternate method to turn **Duplicate Control** off is to press the right mouse button.

Align

The **Align** command aligns one or more controls within a view using the alignment attributes set in the **Align Options** dialog box. **Align** must be preceded by a group selection of one or more controls in the current view.

Align Options

The **Align Options** command provides access to the alignment attributes for aligning controls. When the **Align Options** command is selected, the dialog box in Figure 3-8 is displayed.

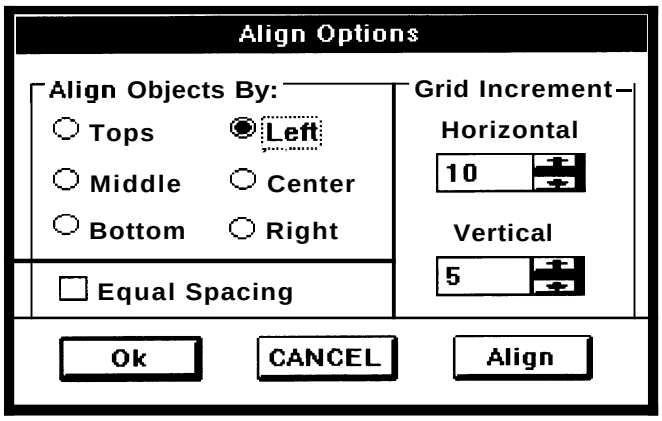


Figure 3-8: The Align Options Dialog Box

Alignment Options

The Front Panel Editor provides six ways to align controls. These are described here:

Top — The selected controls are aligned by the top coordinate value. The tops of the selected controls are aligned with the top of the control closest to the view's top. If Equal Spacing is not on, the horizontal position is not changed.

Middle — The selected controls are aligned by the middle coordinate value. The middles of the selected controls are aligned with the midpoint of the control closest to the top of the view's midpoint. If Equal Spacing is not on, the horizontal position is not changed.

Bottom — The selected controls are aligned by the bottom coordinate value. The bottoms of the selected controls are aligned with the control closest to the bottom of the view. If Equal Spacing is not on, the horizontal position is not changed.

Left — The selected controls are aligned by the left coordinate value. The left sides of the selected controls are aligned with the control closest to the left side of the view. If Equal Spacing is not on, the vertical position is not changed.

Center — The selected controls are aligned by the center coordinate value. The centers of the selected controls are aligned with the control center of the control closest to the left side of the view. If Equal Spacing is not on, the vertical position is not changed.

Right — The selected controls are aligned by the right coordinate value. The right sides of the selected controls are aligned with the control closest to the right side of the view. If Equal Spacing is not on, the vertical position is not changed.

Setting Grid Increments

There are two ways to set grid increments for the FPE, horizontally and vertically. These are explained here.

Horizontal Grid Spacing — Use the horizontal control to change the horizontal spacing of the grid. If the **Equal Spacing** option is enabled, the FPE uses the value assigned for horizontal grid spacing when you align controls. If the **Equal Spacing** option is not enabled, the value is ignored. To change the horizontal spacing, either enter the value you want for horizontal spacing in the box, or click on the up and down arrows with the mouse to increment or decrement the horizontal value.

Vertical Grid Spacing — Use the vertical control to change the vertical spacing of the grid. As with Horizontal Grid Spacing, the FPE uses the value assigned for vertical grid spacing to align controls if the Equal Spacing option is enabled. If not enabled, the FPE ignores the value. Change the vertical spacing in the same way as for Horizontal grid spacing.

Equal Spacing

The **Equal Spacing** control enables or disables the FPE equal spacing option for the selected controls within the current view. The axis of equal spacing is set by the controls in the **Align Objects** options, e.g., Tops, Middle, Bottom. If **Equal Spacing** is enabled, the FPE aligns the selected controls using either the horizontal or the vertical grid increment values.

Align — The **Align** command is available only when controls have been selected. The **Align** command closes the **Align Options** dialog box and places the selected controls, using the spacing and grid increment attributes set in the **Align Options** dialog box.

Ok — Select the **Ok** command when you want to leave the **Align Options** dialog box and make the selected values current. The **Ok** command does not produce alignment of controls. To align controls, use either the **Align** command in the **Align Options** dialog box, or the **Align** command in the **Edit** menu.

Cancel — Select the **Cancel** command to exit from the **Align Options** dialog box without changing any of the alignment values.

View Menu Operations

Table 3-3 lists the **View Menu** commands. The commands are explained below.

Table 3-3: The View Menu Commands

Command	Function
New View	Create a new view
Rename View	Rename the current view
Delete View	Delete view from script
Next View	Show the next view, descending order
Reorder View	Change the listing order
View List	Select a view from a list

New View Command

The **New View** command creates a new view. Choosing the **New View** command causes this dialog box to appear:

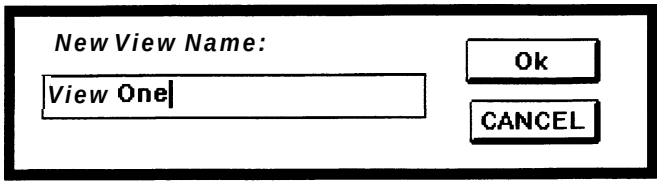


Figure 3-9: The New View Dialog Box

New View Name — Enter the name of the view you wish to create.

Ok — When you select the **Ok** command, the FPE closes the **New View** dialog box, and creates a new view window.

Cancel — Selecting the **Cancel** command causes the **New View** dialog box to close, and returns you to the main menu without creating a new view.

Delete View Command

The **Delete View** command deletes the current view from the edit session. Choosing the **Delete View** command causes the message box shown in Figure 3-10 to appear.

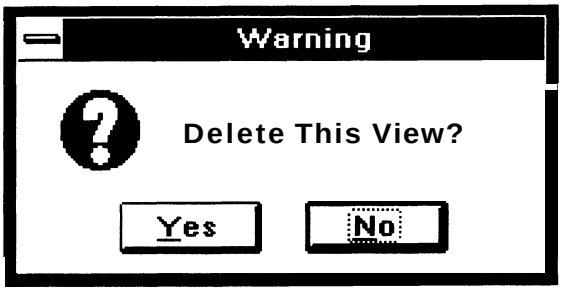


Figure 3-10: The Delete View Message Box

Yes — Choose the **Yes** command if you wish to delete the current view from the edit session.

No — Choose the **No** command if you wish to return to the main menu deleting the current view.

Rename View Command

The **Rename View** command renames a current view. Choosing the **Rename View** command causes the dialog box in Figure 3-11 to appear.

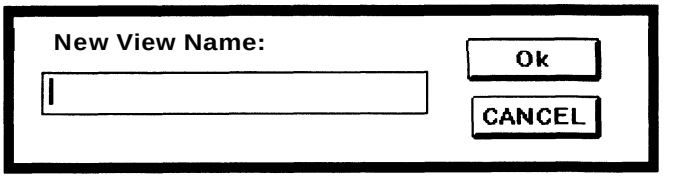


Figure 3-11: The New View Dialog Box

New View Name — Enter the new view name in this edit box

Ok — When you select the **Ok** command, the FPE closes the **New View** dialog box, and changes the name of the current view.

Cancel — Selecting the **Cancel** command closes the **New View** dialog box, and returns you to the main menu without changing the name of the current view.

Next View Command

The **Next View** command makes the next view appear for editing. If there is only one view, the **Next View** command is not available. If **Next View** command is selected while the last view on the list is displayed, the first view on the list will then be displayed.

Reorder View Command

The **Reorder View** command provides the capability to change the order in which the views are written to the ISD file. Changing the order will also change the order the views are listed in the FPE View Menu. Usually the first view in an ISD script is the initial view displayed when an ISD is opened with either the FPE or IPG applications.

Choosing the **Reorder View** command causes the dialog box in Figure 3-12 to appear.

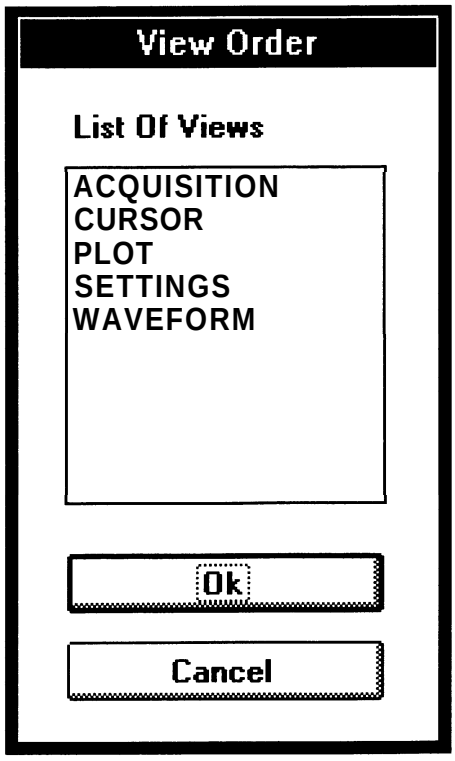


Figure 3-12: The Reorder View Dialog Box

To change the order, use the mouse cursor to select the view that you wish to move and then place the cursor at the new position in the list of views.

NOTE

All control variables within an ISD must be defined before they are used. Generally this is not a problem when changing view order because a control is not used outside the defining view block. If the control is used as an ISD global be sure to define the control variable before using it.

View List Command

The **View List** command is a unique menu item because it gives you a dynamic menu list of the views available in the current script. The view name is added to the list on creation of a new view and deleted from the list on deletion of a view. You can select a view by selecting the view name from the view menu list of view names. The selected view becomes the current view for editing.

Controls Menu Operations

Table 3-4 lists the **Controls Menu** commands. The commands are explained below.

Table 3-4: The Controls Menu Commands

Command	Function
Textbox	Create text box controls
Editbox	Create edit box controls
Listbox	Create list box controls
Pushbutton	Create pushbutton controls
Checkbox	Create check box controls
Radiobutton	Create radiobutton controls
Text	Create text graphics
Line	Create line graphics
Wave Display	Create wave display controls

Creation Cursor

Choose the type of control you wish to create from the **Controls** menu. The current control is shown by a check mark in front of the control name. The editor is switched to the create mode upon selection of a control. Create mode is indicated by the cursor changing to a cross-hair type cursor as shown in Figure 3-13.

Figure 3-13: The Control Creation Cursor

Textbox Control Command

Textbox is a display control that displays textual information as shown in Figure 3-14. The Textbox can have an optional label associated with it.

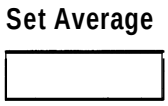


Figure 3-14: Sample Textbox Control

Textbox Control Options — The options dialog box for text box options provides for the setting of the **Textbox** control's optional parameters. The options dialog box for the **Textbox** control is shown in Figure 3-15.

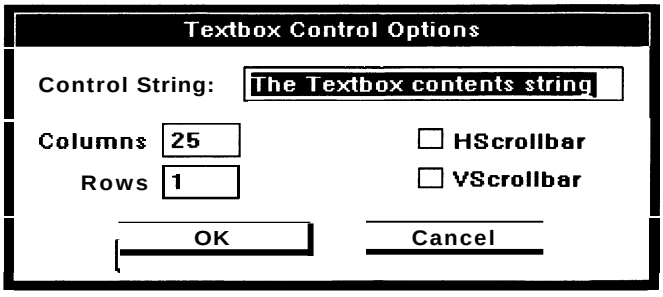


Figure 3-15: The Textbox Control Options Dialog Box

Control String — The **Control String** edit box allows setting of the initial text to show in the **Textbox** control.

Columns — The **Columns** edit box allows the setting of the width of the **Textbox** control. The number represents the width in characters of the control. The control's width can also be changed by using the mouse to drag the control to the desired width.

Rows — The **Rows** edit box allows the setting of the height of the **Textbox** control. The number represents the height in characters of the control. The control's height can also be changed by using the mouse to drag the control to the desired height.

HScrollbar — The **HScrollbar** check box turns horizontal scroll bars on and off for the **Textbox** control. An **X** in the check box indicates horizontal scroll bars are visible.

VScrollbar — The **VScrollbar** check box turns vertical scroll bars on/off for the **Textbox** control. An **X** in the check box indicates vertical scroll bars are visible.

Ok — The **Ok** command will save your changes and exit the dialog box.

Cancel — The **Cancel** command will exit the **Options** dialog box without saving your changes.

Editbox Control Command

Editbox is an input and a display control.

The **Editbox** control can display information and allow the user to enter text or values. The edit box may have an optional label associated with it as shown in Figure 3-16.

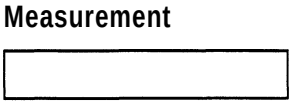


Figure 3-16: Sample Editbox Control

Editbox Control Options — The **Editbox Control Options** dialog box provides for the setting of the **Editbox** control's optional parameters.

The **Options** dialog box for the **Editbox** control is shown in Figure 3-17.

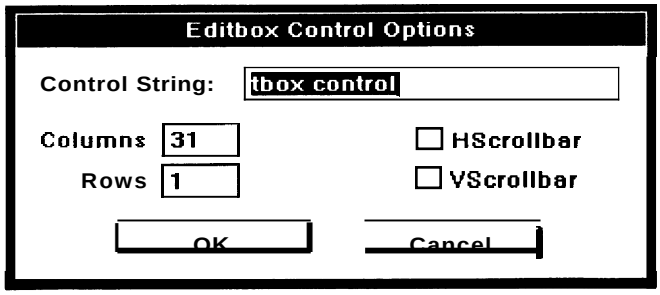


Figure 3-17: The Editbox Control Options Dialog Box

Control String — The **Control String** edit box allows setting of the initial text to show in the **Editbox** control.

Columns — The **Columns** edit box allows the setting of the width of the **Editbox** control. The number represents the width in characters of the control. The control width can also be changed by using the mouse to drag the control to the desired width.

Rows — The **Rows** edit box allows the setting of the height of the **Editbox** control. The number represents the height in characters of the control. The control height can also be changed by using the mouse to drag the control to the desired height.

HScrollBar — The **HscrollBar** check box turns horizontal scroll bars onloff for the **Editbox** control. An **X** in the check box indicates horizontal scroll bars are visible.

VScrollBar — The **VscrollBar** check box turns vertical scroll bars on/off for the **Editbox** control. An **X** in the check box indicates vertical scroll bars are visible.

Ok — The **Ok** command will save the optional parameters and exit the dialog box.

Cancel — The **Cancel** command will exit the **Options** dialog box without saving the optional control parameters.

ListBox Control Command

ListBox is primarily a display control. The list box displays a list of items that the control user may select from during panel operation. A Listbox control has an optional label associated with it. Figure 3-18 shows an example of a list box control.

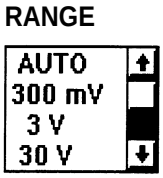


Figure 3-18: Sample ListBox Control

ListBox Control Options — The **ListBox Control Options** dialog box provides for the setting of the **ListBox** control's optional parameters.

The **Options** dialog box for the **ListBox** control is shown in Figure 3-19.

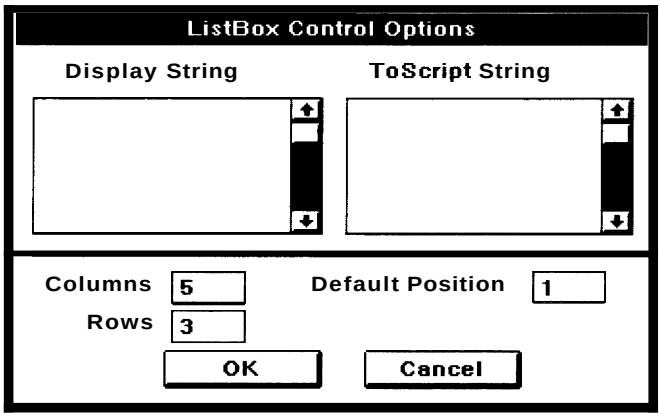


Figure 3-19: The ListBox Control Options Dialog Box

Display String — The **Display String** edit box allows the setting of the text shown within the **ListBox** control. The text is typed into the edit box and appears as it will in the list box control.

ToScript String — The **ToScript String** edit box allows the definition of the ToScript parameters for the **ListBox** control. The order of the entries correlates to the **Display String** edit box strings.

Columns — The **Columns** edit box allows the setting of the width of the **ListBox** control. The number represents the width in characters of the control. The control width can also be changed by using the mouse to drag the control to the desired width.

Rows — The **Rows** edit box allows the setting of the height of the **ListBox** control. The number represents the height in characters of the control. The control height can also be changed by using the mouse to drag the control to the desired height.

Default Position — The default position edit box allows for setting the initial selection in the ListBox control. The number designates the row that is selected when the ListBox control is initially shown.

Ok — The Ok command will save the control's optional parameters and exit the dialog box and return you to the main menu.

NOTE

All other dialog boxes will close if you press the RETURN key. The ListBox options dialog box does not close when a RETURN key is entered. The RETURN key is used to allow multiple lines in the Display String and the *ToScript* String edit boxes. The dialog box is closed by either using the mouse to select the ***Ok*** command, or by using the tab key to move the focus to the ***Ok*** command and then entering a RETURN key.

Cancel — The Cancel command will exit the Options dialog box without saving the optional control parameters.

PushButton Control Command

PushButton is an input control as shown in Figure 3-20. The control is ON while active. Use this control where no response is required.

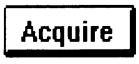


Figure **3-20:** Sample PushButton Control

PushButton Control Options — The PushButton Control Options dialog box provides for the setting of the PushButton optional parameters.

The options dialog box for the PushButton control is shown in Figure 3-21.

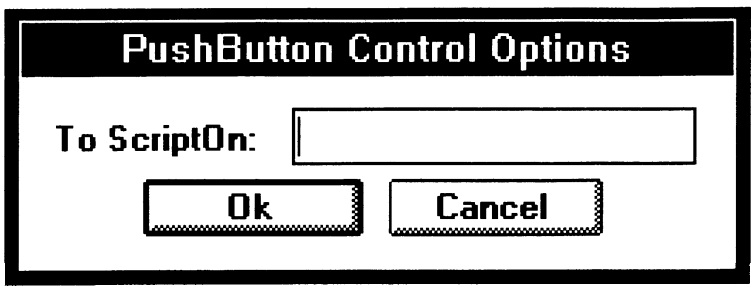


Figure **3-21**: PushButton Control Options Dialog Box

To ScriptOn — The **To ScriptOn** edit box allows setting the text for the **To ScriptOn** parameter. Enter the text as you wish it to appear between double quotes and do not enter the quotes.

Ok — The **Ok** command will save the optional parameters and exit the dialog box.

Cancel — The **Cancel** command will exit the options dialog box without saving the optional control parameters.

CheckBox Control Command

CheckBox is an input control as shown in Figure 3-22. A check box shows the state of a control: an X indicates ON, and no X indicates OFF.

Figure 3-22: Sample CheckBox Control

CheckBox Control Options — The **CheckBox Control Options** dialog box provides for the setting of the **CheckBox** control optional parameters.

The options dialog box for the **CheckBox** control is shown in Figure 3-23.

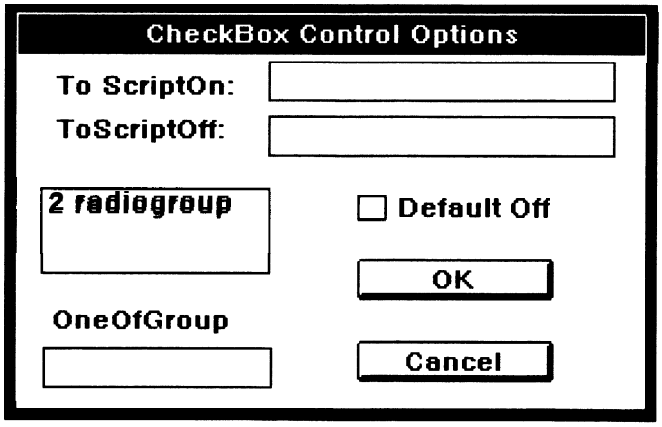


Figure 3-23: CheckBox Control Options Dialog Box

To ScriptOn — The **To ScriptOn** edit box allows setting the text for the ToScriptOn parameter. Enter the text as you wish it to appear between double quotes. Do not enter the double quotes.

To ScriptOff — The **To ScriptOff** edit box allows setting the text for the ToScriptOff parameter. Enter the text as you wish it to appear between double quotes. Do not enter the double quotes.

OneOfGroup — The **OneOfGroup** parameter is set using the edit box below the OneOfGroup title. The list above the OneOfGroup title lists the names of OneOfGroup variable names in use for the current script. If the desired group is one of the group names already used, you can set the OneOfGroup variable by double clicking on the name of the group in the list. A double click selection places the OneOfGroup name in the edit box and becomes the control's OneOfGroup parameter. The number before the group name indicates the number of times the OneOfGroup variable is used in the script.

Default On/Off — The **Default On** or **Default Off** check box sets the initial state of the check box control. If the state is Default On, the check box control will be checked initially. If it is Default Off, the check box control will not be checked initially.

Ok — The **Ok** command will save the optional parameters and exit the dialog box.

Cancel — The **Cancel** command will exit the options dialog box without saving the optional control parameters.

RadioButton Control Command

RadioButton is an input control. The RadioButton shows mutually exclusive selections as in Figure 3-24. Within a RadioButton group only one button may be in the ON state, all others are OFF. The ON state is shown by filling the circle, OFF state is shown by an empty circle.

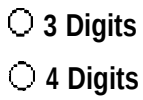


Figure 3-24: Sample RadioButton Control

RadioButton Control Options — The **RadioButton Control Options** dialog box provides for the setting of the **RadioButton** control optional parameters.

Figure 3-25 shows the **Options** dialog box for the **RadioButton** control.

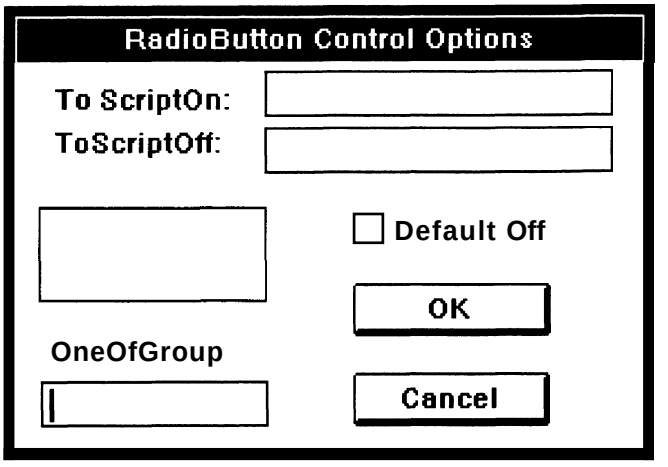


Figure 3-25: RadioButton Control Options Dialog Box

To ScriptOn — The **To ScriptOn** edit box allows setting the text for the **To ScriptOn** parameter. Enter the text as you wish it to appear between double quotes, but do not enter the double quotes.

To ScriptOff — The **To ScriptOff** edit box allows setting the text for the **To ScriptOff** parameter. Enter the text as you wish it to appear between double quotes, but do not enter the double quotes.

OneOfGroup — The **OneOfGroup** parameter is set using the edit box below the OneOfGroup title. The list above the OneOfGroup title lists the names of OneOfGroup variable names in use for the current script. If the group is already set, you can select the current RadioButton into the group by double clicking on the name of the group in the list. A double click selection places the OneOfGroup name in the edit box and becomes the RadioButton controls OneOfGroup parameter. The number before the group name indicates the number of times the variable is used in the script.

Default On/Off — The **Default On** or **Default Off** check box sets the initial state of the RadioButton control. If the state is Default On, the RadioButton control will be checked initially. If the state is Default Off, the RadioButton control will not be checked initially.

Ok — The **Ok** command will save the optional parameters and exit the dialog box.

Cancel — The **Cancel** command will exit the options dialog box without saving the optional control parameters.

Text Control Command

The **Text** control is a graphics aid that serves only as a label on a panel, as seen in Figure 3-26. There is no instrument interaction with this control. This control is sometimes referred to as a ConnectText control.

DM5120 AC VOLTMETER

Figure 3-26: Sample Text Control

ConnectText Control string is set with the **Label** edit box in the parameters control dialog box.

Line Control Command

Line control is a graphics aid. The Line is static and does not interact with the instrumentation. ConnectLine is an alternate term used to describe the line control.

Figure 3-27 shows the use of lines to visually group RadioButton controls.

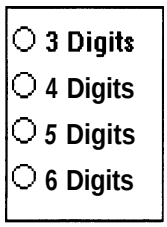


Figure 3-27: Example Showing the Use of Line Control

WaveDisplay Control Command

The **WaveDisplay** display control, shown in Figure 3-28, displays instrument data, usually waveforms.

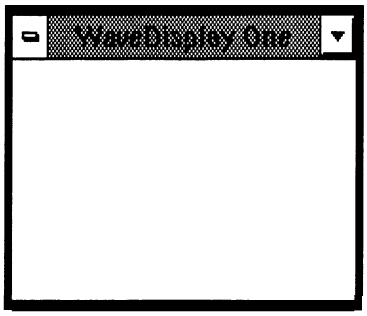


Figure 3-28: Sample WaveDisplay Control

ControlGroup Menu Operations

Table 3-5 lists the **ControlGroup Menu** commands. The commands are explained below.

Table 3-5: The ControlGroup Menu Commands

Command	Function
Break ControlGroup	Remove controls from an ISD ControlGroup
Make ControlGroup	Group controls into an ISD ControlGroup
Show ControlGroup	Show the controls in an ISD ControlGroup

Break ControlGroup Command

The **Break ControlGroup** command removes the selected controls from a common ControlGroup. The **Break ControlGroup** command must be preceded by a **Show ControlGroup** command or a select group action using the mouse.

Make ControlGroup Command

The **Make ControlGroup** command builds an ISD ControlGroup with the selected controls as members of that group. All controls within a Control-Group share the same setting and measurement code. A selection of the controls must precede the selection of the **Make ControlGroup** command.

Show ControlGroup Command

The **Show ControlGroup** command shows the controls that are in the same ISD ControlGroup. The **Show ControlGroup** command must be preceded by a single control selection, the selected control being one of the members of the ControlGroup you wish to see. The control group is shown by sur-rounding the controls that are in the ISD ControlGroup with a selection box. Figure 3-29 shows group selection.

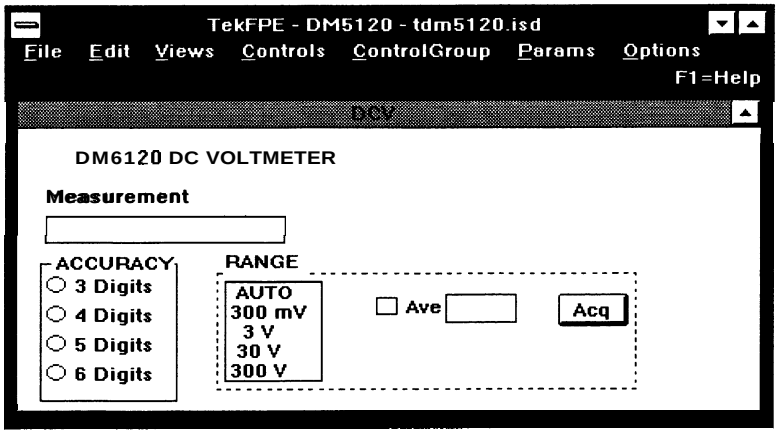


Figure 3-29: Group Selection of Controls

See Section 6, Instrument Script Language Descriptions, for a more detailed description of the ISD ControlGroup block.

Params Menu Operations

Table 3-6 lists the **Params Menu** commands. The commands are explained below.

Table 3-6: The Params Menu Commands

Command	Function
Variables	Show the control parameters dialog boxes
Updatelist	Show the controls updatelist
Set Scene	Open a Settings Scene edit window
Measurement Scene	Open a Measurement Scene edit window
Script Name	Open a Script Name edit window
Learn Scene	Open a Learn Scene edit window
BusNotes	Provide selection of bus parameters
Check ISD	Check the ISD for TekTMS compatibility

Variables Command

The **Variables** command opens a dialog box for setting control parameters. The required control parameters are presented to the user in this dialog box. Each control must have a valid variable name and variable type. In addition to the required parameters, the parameter dialog box has a label edit box for entering a control's label string and a dropdown listbox to provide selection

of the optional parameters for the control. A control must be selected before the **Variables** command is available for selection. The control parameter dialog box may also be opened by double clicking on a control.

An example of the parameter dialog box is shown here. The current control type is shown in the window title bar and if there are current parameters set, they are displayed in the appropriate places.

Name — Enter the name you wish to give the current control. The variable name has a 16 character maximum length and cannot contain spaces. In addition the variable name's first character must be an alphabetic character.

Type — Each control requires a variable type. The ranges of variable types available depends on the type of control you are defining. Only the valid variable types are presented for selection in a dropdown Listbox. If a variable type is choice is not made, a default type, string, is set for the control.

Label — Enter the label text you wish to associate with the current control in the Label edit box. The Label text is shown inside the pushbutton, Radio-Button, and checkbox controls. The text box, edit box, and list box labels are displayed above the control. The **Wavedisplay** control displays the label in the title bar of the wavedisplay window.

Parameters — The **Parameters** dropdown listbox allows the selection of additional control parameters, and selection of the scene code edit windows. Only the valid control options for the current control are presented as in Figure 3-30.

The control parameter dialog box also contains a position and size status box. The position of the current control is reported to the user in screen coordinates as well as the size of the control.

The control parameter dialog box is closed by selecting the system menu button, either by double clicking on it, or a single click followed by a **Close** command selection.

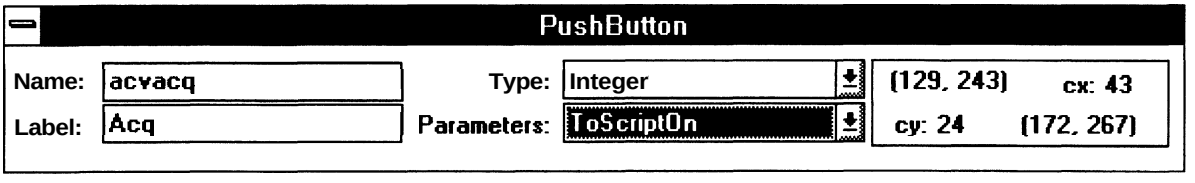


Figure 3-30: Control Parameters Dialog Box for the PushButton

UpdateList Command

The **UpdateList** command opens a dialog box that shows a list of controls within the current view. The list highlights the control names that are in the currently selected controls UpdateList. To show the UpdateList for other controls, select another control within the current view.

When you select the **UpdateList** command, the dialog box in Figure 3-31 is displayed.

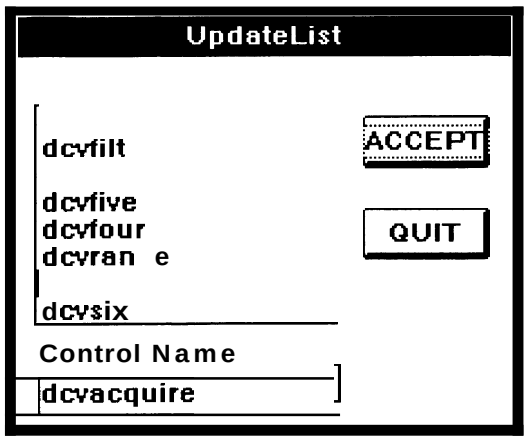


Figure 3-31: UpdateList Dialog Box

List of Views Controls — The controls listed are all the controls within the current view. If a listed control is on the current controls UpdateList, the control's variable name is highlighted. To select or deselect a control variable name, use the mouse to click on the name you wish to add or delete from the current control UpdateList.

Control Name — The text box shows the name of the current control.

Accept — The **Accept** command changes the current controls UpdateList to the control variable names selected in the list box.

Quit — The **Quit** command closes the UpdateList dialog box and returns you to the main menu. The **Quit** command does not change the controls UpdateList, the **Accept** command must precede the **Quit** command to change the controls UpdateList.

Set Scene Command

The **Set Scene** command opens a scene edit window. The edit window provides the means to define the setting code for the current control. If the current control is in a ControlGroup that has multiple controls, the setting scene code is for all controls. Upon selection of the **Set Scene** command, the **Set Scene** edit window shown here is displayed. The type of edit window and the name of the current control is shown in the title bar of the window as in Figure 3-32.

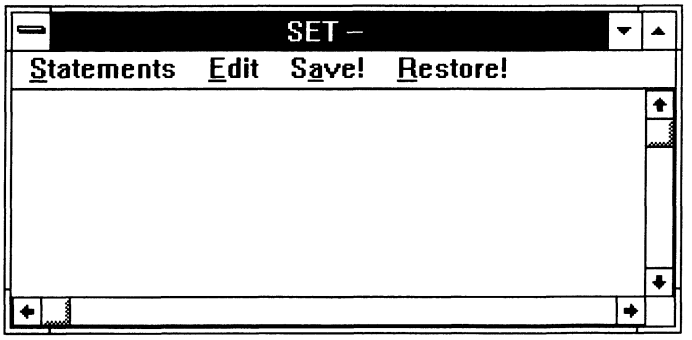


Figure 3-32: Set Scene Edit Window

Statements Command — The **Statement** menu command contains canned scene statements that are inserted into the scene code upon selection. The statements are inserted at the cursor position in the edit window. Some of the statements are complete statements and require no further input, while other statements require you to finish the statement with the information you wish to enter. In general, position the cursor at the point where additional information is required for each statement. The details of the ISD statements may be found in Section 6, Instrument Script Language Descriptions, of this manual. Select the application **Help** button for specific command information.

SendToInst	Inserts `cont->"";`
ReceiveFromInst	Inserts `inst->"";`

If Statement	Inserts `if() then`
	`else`
	`endif`

While Statement	Inserts `while() do`
Temporary Variable	Inserts `tempvar:;`

GPIB Commands — The **GPIB Commands** menu command is a pop-up menu that inserts canned statements for bus control into the script scene. They include:

Attention	Inserts `ATN;`
Device Clear	Inserts `DCL;`
Go To Local	Inserts `GTL;`
Group Execute Trigger	Inserts `GET;`
Interface Clear	Inserts `IFC;`
Local Lockout	Inserts `LLO;`
Remote Enable	Inserts `REN;`

Selective Device Clear	Inserts `SDC;`
TimeOut	Inserts `TIM;`
UnTalk	Inserts `UNT;`
Unlisten	Inserts `UNL;`

Functions Command — The **Functions** menu command is a pop-up menu that insert canned statements into the script scene for functions available in the ISD script language. The functions include:

Chr ()	Inserts `Chr();`
Display ()	Inserts `Display();`
FastDCRead ()	Inserts `FastDCRead();`
FastDCWrite ()	Inserts `FastDCWrite();`
ReadLength ()	Inserts `ReadLength();`
SerialPoll ()	Inserts `TimeDelay();`
TimeDelay ()	Inserts `TimeDelay();`
WaveFormToAdif ()	Inserts `WaveFormToAdif();`
WaveFormToVar ()	Inserts `WaveFormToVar();`

Edit Command — The **Edit** menu command provides commands for editing scene code. The following edit commands are available in the **Edit** menu:

- **Undo** — Some edit steps can be undone by selecting the **Undo** command. If the **Undo** command is gray, the command is not available because there is nothing to undo.
- **Cut** — The **Cut** command cuts selected text from the scene edit window to the clipboard.
- **Copy —** The **Copy** command copies the selected text from the scene edit window to the clipboard.
- **Paste** — The **Paste** command copies text from the clipboard into the scene window. If there is text in the clipboard the paste command is enabled.
- **Clear All —** The **Clear All** command clears the entire contents of the scene edit window.

Save! Command — The **Save!** command causes the scene edit windows contents to be saved in the ISD script.

Restore! Command — The **Restore!** command restores the previous scene text. The previous scene text is the text that existed for the control before the edit window was opened, or restores the text saved the last time.

Measurement Scene Command

The **Measurement Scene** command opens a scene edit window. The edit window provides the means to define the measure block code for the current control. If the current control is in a ControlGroup that has multiple controls, the measure scene code is for all controls. Upon selection of the **Measure Scene** command, the **Measure Scene** edit window shown here is displayed. The type of edit window and the name of the current control is shown in the title bar of the window as in Figure 3-33.

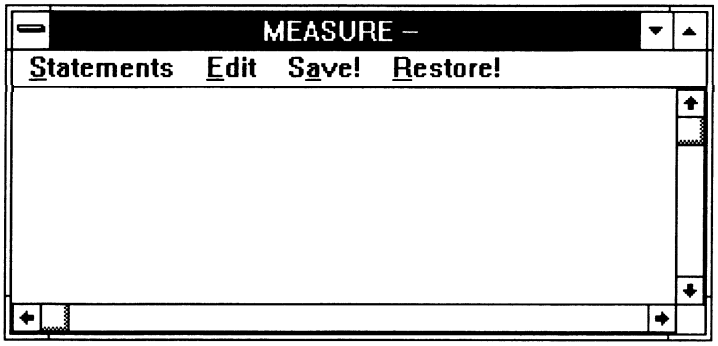


Figure 3-33: Measurement Scene Edit Window

Statements Command — The **Statement** menu command contains canned scene statements that are inserted into the scene code upon selection. The statements are inserted at the cursor position in the edit window. Some of the statements are complete statements and require no further input, while other statements require you to finish the statement with the information you wish to enter. In general, position the cursor at the point for additional information for the statement. The details of the ISD statements may be found in Section 6, Instrument Script Language Descriptions, of this manual. The statements available with this command follow.

SendToInst	Inserts `cont->"";`
ReceiveFromInst	Inserts `inst->"";`

If Statement	Inserts `if() then`
	`else`
	`endif`

While Statement	Inserts `while() do`
Temporary Variable	Inserts `tempvar:;`

GPIB Commands — The **GPIB Commands** menu command is a pop-up menu that inserts canned statements for bus control into the script scene. They include:

Attention	Inserts `ATN;`
Device Clear	Inserts `DCL;`
Go To Local	Inserts `GTL;`
Group Execute Trigger	Inserts `GET;`
Interface Clear	Inserts `IFC;`
Local Lockout	Inserts `LLO;`
Remote Enable	Inserts `REN;`
Selective Device Clear	Inserts `SDC;`
TimeOut	Inserts `TIM;`
UnTalk	Inserts `UNT;`
Unlisten	Inserts `UNL;`

Functions Command — The **Functions** menu command is a pop-up menu that insert canned statements into the script scene for functions available in the ISD script language. The functions include:

Chr ()	Inserts `Chr();`
Display ()	Inserts `Display();`
FastDCRead ()	Inserts `FastDCRead();`
FastDCWrite ()	Inserts `FastDCWrite();`
ReadLength ()	Inserts `ReadLength();`
SerialPoll ()	Inserts `TimeDelay();`
TimeDelay ()	Inserts `TimeDelay();`
WaveFormToAdif ()	Inserts `WaveFormToAdif();`
WaveFormToVar ()	Inserts `WaveFormToVar();`

Edit Command — The **Edit** menu command provides commands for editing scene code. The following edit commands are available in the **Edit** menu command:

- **Undo** — Some edit steps can be undone by selecting the **Undo** command. If the **Undo** command is gray, the command is not available because there is nothing to undo.
- **Cut** — The **Cut** command cuts selected text from the scene edit window to the clipboard.
- **Copy —** The **Copy** command copies the selected text from the scene edit window to the clipboard.
- **Paste** — The **Paste** command copies text from the clipboard into the scene window. If there is text in the clipboard the **Paste** command is enabled.

- **Clear All** — The **Clear All** command clears the entire contents of the scene edit window.

Save! Command — The **Save!** command causes the scene edit windows contents to be saved in the **ISD** script.

Restore! Command — The **Restore!** command restores the previous scene text. The previous scene text is the text that existed for the control before the edit window was opened, or restores the text saved the last time.

Script Name Command

The **Script Name** command opens the **New Script** dialog box. The **New Script** dialog box provides a way to change the script name and the contents of the header remarks section of the **ISD** script. The **New Script** dialog box is detailed under the **New Script** command section.

Learn Scene Command

The **Learn Scene** command opens a learn block dialog box, as shown in Figure 3-34, and opens set and measure scene edit windows. **Setting** and **Measure** edit windows, shown in Figures 3-35 and 3-36, are the same as the ones for coding the controls setting and measure scene code, except that the titles in the edit windows indicate they are **Learn** block edit windows.

The **Learn Scene** command opens the **Learn Scene** dialog box to provide for the input of the Learn blocks variable name and variable type. The variable name given here is the only reference in the settings and measurement code for the Learn block.

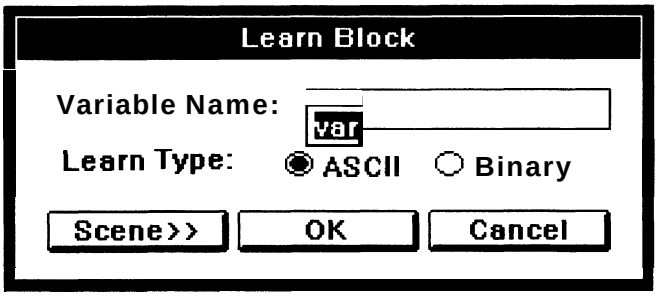


Figure 3-34: Learn Scene Dialog Box

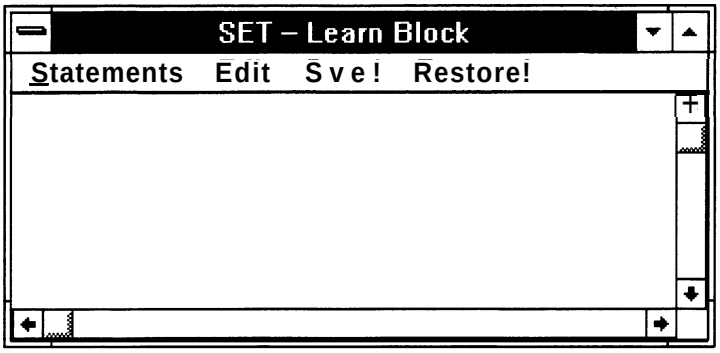


Figure 3-35: Learn Scene Set Edit Window

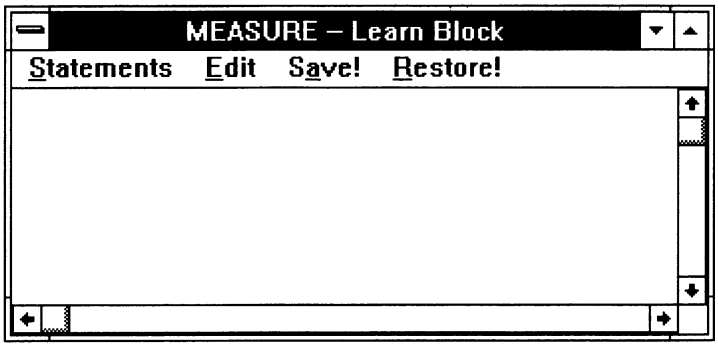


Figure 3-36: Learn Scene Measure Edit Window

Statements Command — The **Statement** menu command contains canned scene statements that are inserted into the scene code upon selection. The statements are inserted at the cursor position in the edit window. Some of the statements are complete statements and require no further input, the other statements require you to finish the statement with the information you wish to enter. In general, the cursor is positioned at the point for additional information for the statement. The details of the ISD statements may be found in Section 6, Instrument Script Language Descriptions, of this manual. The statements available with this command follow.

SendToInst	Inserts `cont->"";`
ReceiveFromInst	Inserts `inst->"";`

If Statement	Inserts `if() then`
	
	`else`
	`endif`

While Statement	Inserts `while() do`
Temporary Variable	Inserts `tempvar:;`

- **GPIB Commands —** The **GPIB Commands** menu command is a pop-up menu that inserts canned statements for bus control into the script scene. They include:

Attention	Inserts `ATN;`
Device Clear	Inserts `DCL;`
Go To Local	Inserts `GTL;`
Group Execute Trigger	Inserts `GET;`
Interface Clear	Inserts `IFC;`
Local Lockout	Inserts `LLO;`
Remote Enable	Inserts `REN;`
Selective Device Clear	Inserts `SDC;`
TimeOut	Inserts `TIM;`
UnTalk	Inserts `UNT;`
Unlisten	Inserts `UNL;`

- **Functions Command** — The **Functions** menu command is a pop-up menu that insert canned statements into the script scene for functions available in the ISD script language. The functions include:

Chr ()	Inserts `Chr();`
Display ()	Inserts `Display();`
FastDCRead ()	Inserts `FastDCRead();`
FastDCWrite ()	Inserts `FastDCWrite();`
ReadLength ()	Inserts `ReadLength();`
SerialPoll ()	Inserts `TimeDelay();`
TimeDelay ()	Inserts `TimeDelay();`
WaveFormToAdif ()	Inserts `WaveFormToAdif();`
WaveFormToVar ()	Inserts `WaveFormToVar();`

Edit Command — The **Edit** menu command provides commands for editing scene code. The following edit commands are available in the Edit menu command:

- **Undo** — Some edit steps can be undone by selecting the **Undo** command. If the **Undo** command is gray, the command is not available because there is nothing to undo.
- **Cut** — The **Cut** command cuts selected text from the scene edit window to the clipboard.
- **Copy** — The **Copy** command copies the selected text from the scene edit window to the clipboard.

- **Paste** — The **Paste** command copies text from the clipboard into the scene window. If there is text in the clipboard the **Paste** command is enabled.
- **Clear All —** The **Clear All** command clears the entire contents of the scene edit window.

Save! Command — The **Save!** command causes the scene edit windows contents to be saved in the ISD script.

Restore! Command — The **Restore!** command restores the previous scene text. The previous scene text is the text that existed for the control before the edit window was opened, or restores the text saved the last time.

BusNotes Command

The **BusNotes** command is a command used to set the ISD script bus parameters. The **BusNotes** command opens the **Bus Selection** dialog box shown in Figure 3-37.

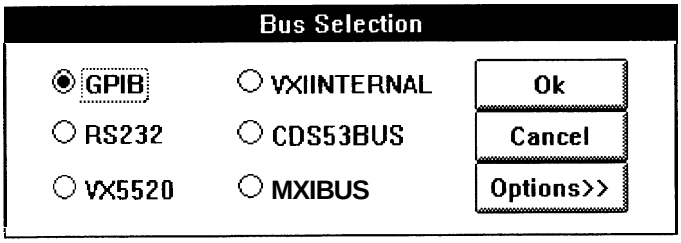


Figure 3-37: The Bus Selection Dialog Box

A detailed description of these bus types may be found in the IPG manual in the Instrument Script Language Description section. The purpose of each command in the **Bus Selection** dialog box follows:

CDS53BUS — Sets the ISD script bus type to CDS.

GPIB — Sets the ISD script bus type to GPIB.

MXIBUS — Sets the ISD script bus type to MXI.

RS232 — Sets the ISD script bus type to RS232.

VX5520 — Sets the ISD script bus type to VX5520.

VXIinternal — Sets the ISD script bus type to VXIinternal.

Options Command — The **Options** command opens the selected bus type parameter dialog boxes. The options dialog boxes allow the setting of the selected bus type configuration parameters.

CDSBUS Parameters — The **Options** command selection with the **CSA53BUS** bus type selected will cause the **CDSBUS Parameters** dialog box to appear as shown in Figure 3-38.

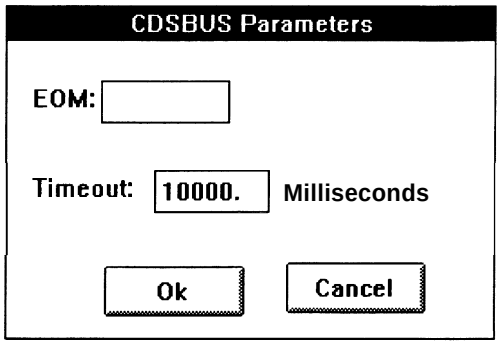


Figure 3-38: The CDSBUS Parameters Dialog Box

EOM Edit Box — The **EOM** edit box provides the selection of the EOM character(s) automatically appended to the end of a device-dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.

Timeout Value — The **Timeout** value sets the wait period before an error report. The timeout parameter is specified in milliseconds.

- **Ok —** The **Ok** command saves the set parameters, closes the options dialog box, and returns you to the **Bus Selection** dialog box.
- **Cancel** — The **Cancel** command discards changes made to the bus parameters and returns you to the **Bus Selection** dialog box.

GPIB Parameters Dialog Box — The **Options** command selection with the GPIB bus type selected will cause the **GPIB Parameters** dialog box, shown in Figure 3-39, to appear.

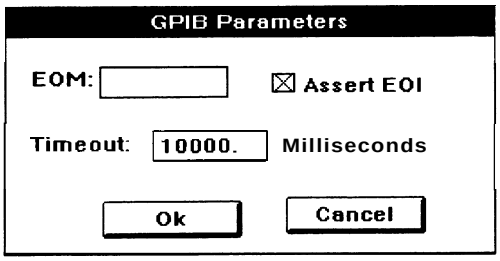


Figure 3-39: The GPIB Parameters Dialog Box

- **EOM Edit Box —** The **EOM** edit box provides the selection of the EOM character(s) automatically appended to the end of a device-dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.
- **Assert EOI Value —** The **Assert EOI** value enables/disables the assertion of EOI on the last byte of messages.
- **Timeout Value** — The **Timeout** value sets the wait period before an error report. The timeout parameter is specified in milliseconds.
- **Ok —** The **Ok** command saves the set parameters, closes the options dialog box, and returns you to the **Bus Selection** dialog box.
- **Cancel —** The **Cancel** command discards changes made to the bus parameters and returns you to the **Bus Selection** dialog box.

MXI Parameters Dialog Box — The **Options** command selection with the **MXI** bus type selected will cause the **MXI Parameters** dialog box, shown in Figure 3-40, to appear.

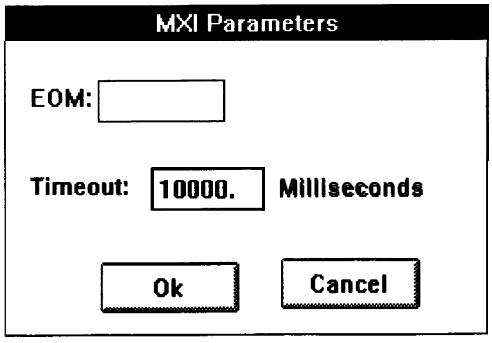


Figure 3-40: The MXI Parameters Dialog Box

EOM Edit Box — The **EOM** edit box provides the selection of the EOM character(s) automatically appended to the end of a device-dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.

Timeout Value — The **Timeout** value sets the wait period before an error report. The timeout parameter is specified in milliseconds.

- **Ok —** The **Ok** command saves the set parameters, closes the options dialog box, and returns you to the **Bus Selection** dialog box.
- **Cancel —** The **Cancel** command discards changes made to the bus parameters and returns you to the **Bus Selection** dialog box.

RS232 Parameters Dialog Box — The **Options** command selection with the RS232 bus type selected will cause the **RS232 Parameters** dialog box, shown in Figure 3-41, to appear.

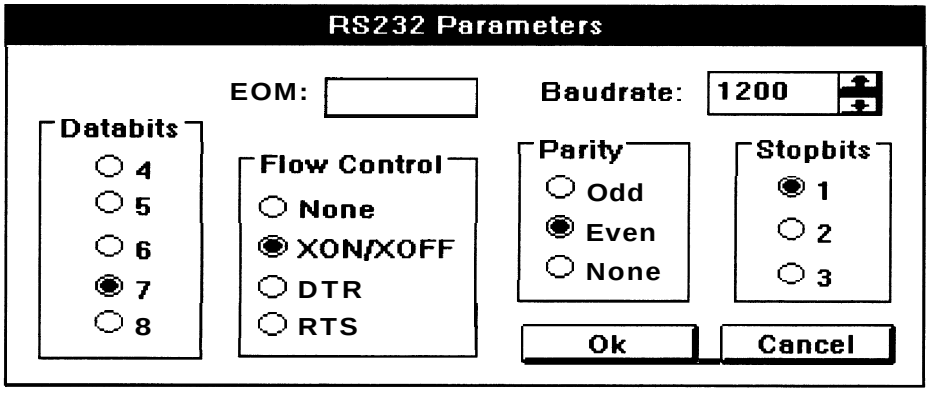


Figure 3-41: The RS232 Parameters Dialog Box

- **EOM Edit Box —** The **EOM** edit box provides the selection of the EOM character(s) automatically appended to the end of a device-dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.
- **Baudrate Edit Box** — The **Baudrate** edit box sets the baud rate for the RS232 bus. The baud rate may be changed by typing in the value, or selected by using the up/down arrows next to the edit box.

  Parity Value — The **Parity** value selects the parity type for the RS232 bus. The parity is set to Odd, Even, or None. The parity selections are mutually exclusive.
- **Flow Control Value** — The **FlowControl** value sets the type of flow control. The available types are, none, xon/xoff, DTR/DSR, or RTS/CTS. The selections are mutually exclusive.

- **Stopbits Value** — The **Stopbits** value sets the number of stop bits used. The available stop bits are, 1, 2, or 3 bits. The stop bit selections are mutually exclusive.
- **Databit Value** — The **Databit** value selects the number of data bits for the RS232 bus. The number of bits may be set at 4, 5, 6, 7, or 8 bits. The Databit selection is mutually exclusive.
- **Ok —** The **Ok** command saves the set parameters and closes the options dialog box returning you to the **Bus Selection** dialog box.
- **Cancel —** The **Cancel** command discards changes made to the bus parameters and returns you to the **Bus Selection** dialog box.

VX5520 Parameters Dialog Box — The **Options** command selection with the VX5520 bus type selected will cause the **VX5520 Parameters** dialog box, shown in Figure 3-42, to appear.

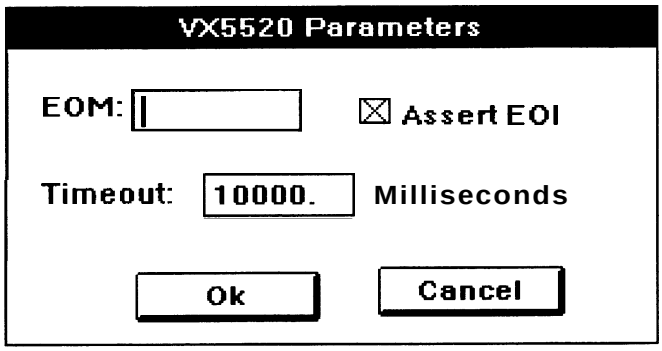


Figure 3-42: VX5520 Parameters Dialog Box

- **EOM Edit Box —** The **EOM** edit box provides the selection of the EOM character(s) automatically appended to the end of a device-dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.
- **Assert EOI Command** — The **Assert EOI** command enables/disables the assertion of EOI on the last byte of messages.
- **Timeout Value** — The **Timeout** value sets the wait period before an error report. The timeout parameter is specified in milliseconds.
- **Ok —** The **Ok** command saves the set parameters and closes the options dialog box returning you to the **Bus Selection** dialog box.
- **Cancel —** The **Cancel** command discards changes made to the bus parameters and returns you to the **Bus Selection** dialog box.

WIinternal Parameters Dialog Box — The options command selection with the VXIinternal bus type selected will cause the **VXIinternal** Parameters dialog box shown, in Figure 3-43, to appear.

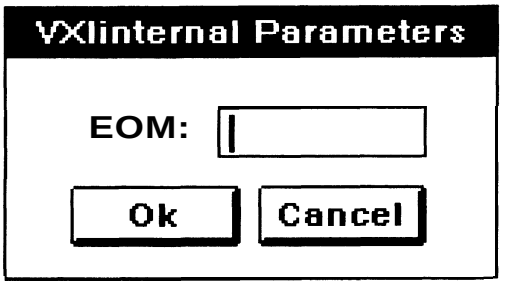


Figure **3-43:** The WIinternal Parameters Dialog Box

- EOM Edit Box — The EOM edit box provides the selection of the EOM character(s) automatically appended to the end of a device–dependent message. The ASCII value of the desired character(s) must be used. If there are two characters, separate the ASCII values with a space.
- Ok — The Ok command saves the set parameters and closes the options dialog box returning you to the Bus Selection dialog box.
- Cancel — The Cancel command discards changes made to the bus parameters and returns you to the Bus Selection dialog box.

Check ISD Command

The Check ISD command checks the ISD script for compatibility with TekTMS/IPG. Use this command to check a script for errors in the scene code blocks and for undefined parameters required by controls.

Option Menu Operations

Table 3-7 lists the Option Menu commands. The commands are explained below.

Table 3-7: The Option Menu Commands

Command	Function
Open NotePad	Open notepad with the current script
Start TekTMS	Start TekTMS/IPG
Scroll Bars	Place scroll bars on the current view
Save **Config**	Save the FPE configuration
ToolBox	Show a tool window

Open NotePad Command

The Open NotePad command generates a temporary ISD script file and opens notepad with the script text visible.

Start TekTMS Command

The Start **TekTMS/IPG** command starts the TekTMS/IPG application. The FPE looks for TekTMS in the current directory. If TekTMS/IPG is not found in the current directory, FPE will search the environment string for the path to TekTMS/IPG.

Scroll Bars Command

The Scroll Bars command causes scroll bars to appear on the views. The scroll bars are used to scroll controls into the view as required. The scrollable area is limited to a fixed area. Initially the scroll bars are positioned in the middle and center of the current view. The view can be scrolled a fixed amount horizontally and vertically.

Save Config Command

The Save **Config** command enables/disables the saving of the editor configuration. The name of the last file edited and the state of the Save **Config** command are saved between FPE application sessions. If you select the Save **Config** command, the next time FPE is opened the last ISD file edited will be automatically opened.

ToolBox Command

The **ToolBox** command opens a window that shows the available controls for the edit session. The current control is highlighted. Use the mouse cursor to select a different tool the same as using the Controls Menu command. Figure 4-1 on page 4-3 shows the Controls Toolbox with each tool labeled.

Help Menu Operations

What the Help Menu Does

The **Help** menu provides online help for the commands and procedures for TekTMS/FPE application. The FPE help system uses the Microsoft help application, WINHELP.EXE. The FPE help system is not the same as the TekTMS help system. Winhelp is called and the FPE help file loaded. Figure 3-44 shows how the first screen appears.

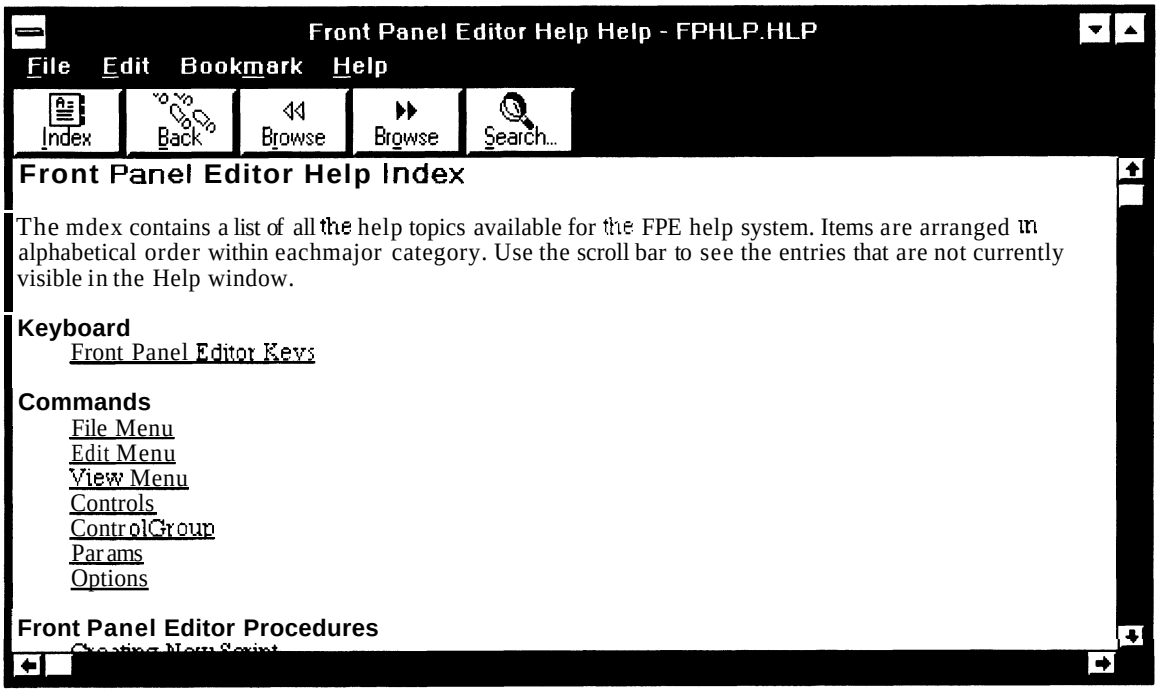


Figure 3-44: Initial FPE Help Screen

The first **Help** screen presents an index to the help topics available. Items in the list that have additional information are indicated by underlined text. The cursor changes to a hand shaped cursor if more information is available. A click of the left mouse button will move you to the additional information.

FPE Tutorial

Introduction to the Tutorial

This section demonstrates the essentials of using TekTMS/FPE in a tutorial that builds an instrument front panel script. You can use this front panel script to generate a set of test steps in TekTMS/IPG. Use the Tektronix DM5010 instrument for this tutorial. The tasks presented in this tutorial are:

- Open the FPE application.
- Create a new instrument front panel.
- Create new controls within the new view.
- Check the ISD script for TekTMS/IPG compatibility.
- Save the front panel to a ISD script file.
- Load the front panel script into the IPG.

This tutorial assumes you are already familiar with the Microsoft Windows interface. If not, please refer to the appropriate Microsoft Windows manuals and possibly their tutorial on learning Windows.

This tutorial also assumes you are using a mouse. Although most FPE functionality is accessible from the keyboard, a mouse is required to get the full benefit from the FPE application.

Getting Started

If you have not installed TekTMS/FPE yet, please do so before you continue. Refer to Section 2, Getting Started for installation instructions.

For this example we assume you have installed TekTMS/FPE in directory C:\FPE. If not, replace C:\FPE, in the following command, with your drive and directory path where you have installed TekTMS/FPE.

Enter **cd C:\FPE** to make the FPE directory the default directory.

Enter **WIN fpedit** to load the FPE application.

Alternatively, if Windows is already loaded, use the mouse to click on the FPE icon to start the FPE application just as you would any other Windows application.

When FPE is loaded, the menu will have two items in it; the **File** and **Help.**

Using the Help Command

Help information on the FPE is available whenever you use FPE. The FPE Help system uses the Microsoft Windows Help Application to provide help. The Microsoft Help Application is described in the Microsoft Windows Users Guide.

Open the Help window either by using the mouse to select the **Help** menu item, or by using the F1 help key. Select a control or dialog and press the **F1** key to get help information on the selected item.

The initial window lists Help topics that you can read. Use the mouse to select a topic. Upon selection of the topic, the help text is shown.

There is help information available on the menu commands, accelerator keys, and most of the procedures.

Creating a New View

To create a new view:

1. Select the **File New Script** command from the main menu.
2. Enter "DM5010" in the **Script Name** edit box. DM5010 will become the name of the new script. You may enter any text you wish to associate with this script as remarks in the **Remarks** edit box. This text will be placed at the beginning of the script file and is not part of the ISD script. Do not enter any text here now.
3. Choose the Ok command to close the **New Script** dialog box. After the **New Script** dialog box is closed, the **New View Name** dialog box appears.
4. Enter "DCV" in the **New View Name** edit box. DCV is the name of the new view you are creating.
5. Choose the **Ok** command to close the dialog box and return to the main menu.

Notice that the applications main window title now shows the application name and the name of the script. After the script has been saved once, the file name will also appear in the application window's title bar.

The view name appears in the title bar of the new panel window.

You now have a view with nothing in it. The empty view can make a valid ISD script; however, it does not perform any useful function.

Creating Controls in the New Panel

The first control you will create is a subtitle for the front panel. Here are the steps to follow:

1. Choose the **Controls Text** command from the main menu. Alternately, you can select the Text button from the Controls Toolbox shown in Figure 4-1. This selection makes the **Text** control the current control type. You can use the Controls Toolbox in selecting other controls in this Tutorial or you can elect to close the Toolbox.

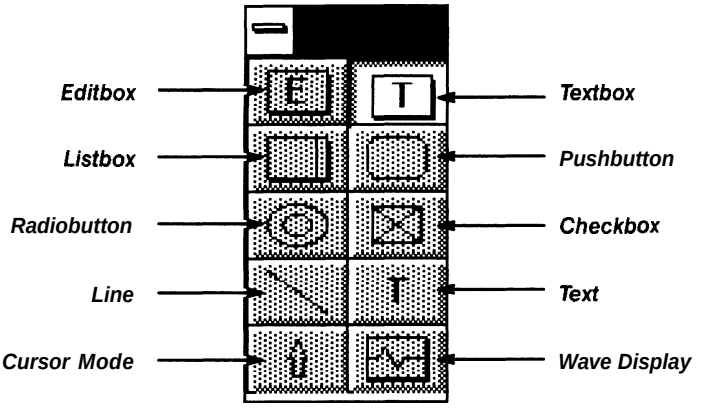


Figure 4-1: Controls Toolbox

2. Using the mouse, position the cursor in the view where you wish to place the text. Click and drag the cursor to size the **Text** control. The selection rectangle drawn by the drag operation is replaced with the new text control.
3. The **Text** control is initially drawn with the default string **Text** visible. To replace this text with your text, double click on the main part of the control to open the parameters dialog box for the text control.
4. Enter "DM5010 DC VOLTMETER" in the **Label** edit box and press the Return key.
5. The new text is inserted into the text control and shown within the view. FPE automatically resizes the text control to fit the length of the text. Although FPE allows you to size the text control, when TekTMS/IPG uses the text control, it will again be resized to fit the length of the text.
6. If the **Text** control needs to be repositioned, use the mouse to select the control and drag it to the new position.
7. Next, create a control to serve as the digital readout for the DC voltmeter. A **Textbox** control will serve this readout function. Follow these steps:
   a. Choose the **Controls Textbox** command from the main menu.

b. Position the cursor in view at the position you wish to draw the **Textbox** control. Hold down the left mouse button while you drag the selection rectangle to the desired size.

c. The title of the parameter dialog box for the control changes to **TextBox** to indicate the type of control you are creating.

d. Enter the controls variable name by entering "dcvreading" into the **Name** edit box.

e. Select the **FLOAT** variable type using the **Type** selection box. Click on the arrow to select from a list of the available types.

f. Enter "Measurement" in the **Label** edit box. The minimum required parameters for the new control are now complete. The optional parameters of the control are accessible through the **Parameters** selection box. However, for this control, there are no optional parameters.

8. Next, you need the capability to select the accuracy of the voltmeter display. The accuracy is determined by the number of digits shown in the measurement display control. For the DM5010, you must select one of four options. The **RadioButton** control with its one-of-many selection functions fits the accuracy control required. Four radiobuttons serve the accuracy selection function for the voltmeter. In addition to the radiobutton controls, use a text control to label the group and use lines to visually group the four radiobuttons.

9. Select and create a radiobutton control using the same steps you used before to create the preceding controls.

NOTE

You could create all four controls using the above step, however, for variety, use the duplicate function to create the remaining three *radiobutton* controls.

10. While the newly created **RadioButton** control has the focus, choose the **Edit Duplicate** command. The selection of the **Duplicate** command puts the editor in the duplicate mode. While in the duplicate mode, the create cursor appears. Use the left mouse button to create a duplicate of the control. Reselect the **Edit Duplicate** command to create each duplicate or use the shortcut key CTRL-D to select the duplicate command.

11. Each radiobutton requires a valid **OneOfGroup** variable name. Radiobuttons with the same **OneOfGroup** variable name are mutually exclusive. Only one button within this **OneOfGroup** is on at any one time. To set the **OneOfGroup** variable name, follow these steps:

    a. Open the controls optional parameter dialog box for each control by selecting **OneofGroupName** from the **Parameters** selection box.

b. Enter "accuracy" into the **OneOfGroup** edit box for each of the radiobutton controls. This gives the four radiobuttons the OneOf-Group common name **accuracy.** This common **OneOfGroup** name associates the four radiobuttons in a mutually exclusive relationship.

c. If the OneOfGroup name is already in the list that appears above the **OneOfGroup** edit box, use the mouse to select the **OneOfGroup** variable name by double clicking on the desired **OneOfGroup** name. The selected text will appear in the **OneOfGroup** edit box and become the radiobutton control's **OneOfGroup** variable name when you close the options dialog box with the **Ok** command.

d. Enter the **Radiobutton** control parameters for each radiobutton using the following text.

Radiobutton 1:	**Name**	`"dcvthree"`
	Type	`INT`
	Label	`"3 Digits"`
	OneOfGroup	`"accuracy"`
Radiobutton 2:	**Name**	`"dcvfour"`
	Type	`INT`
	Label	`"4 Digits"`
	OneOfGroup	`"accuracy"`
Radiobutton 3:	**Name**	`"dcvfive"`
	Type	`INT`
	Label	`"5 Digits''`
	OneOfGroup	`"accuracy"`
Radiobutton **4:**	**Name**	`"dcvsix"`
	Type	`INT`
	Label	`"6 Digits"`
	OneOfGroup	`"accuracy"`

Setting the UpdateList

When the number of digits selected changes, it is helpful if the DC voltmeter readout changes to reflect the new number of digits. To do this, you need to set one more parameter for the radiobutton controls. The UpdateList parameter lists the controls to update whenever this control is activated. To add the **Readout Control** variable name (dcvreading) to each radiobutton UpdateList parameter, follow these steps:

1. Set the focus to the 3 **Digits** radiobutton control and choose the **UpdateList** command in the **Parameters** selection box. The **UpdateList** dialog box will appear listing the controls in the current view and showing the variable name of the control in a separate edit box.

2. Select the variable names within the list by clicking the mouse on the desired name in the list. Repeated clicking on the same name will toggle the selection. Multiple names can be selected from the list.

3. Select the **dcvreading** variable name from the list of variable names, then select the **Accept** command. The **Accept** command adds the selected variable names to the current controls UpdateList. Don't select the **Quit** command yet, we're not done with all the UpdateLists.

4. To set the UpdateLists for the other three radiobutton controls, use the mouse to select the next radiobutton control. The next radiobutton variable name appears in the **Control Name** text box of the **UpdateList** dialog box, indicating which control is selected. Set the UpdateList for all four of the radiobutton controls to include **dcvreading** variable name. Don't forget to use the **Accept** command before selecting each new radiobutton control.

5. After all radiobutton UpdateLists are set, choose the **Quit** command to leave the **UpdateList** dialog box.

Aligning Controls

There are now four radiobutton controls in the view and unless you were very careful, they are probably not evenly spaced or lined up with each other. To align them, follow the steps described here.

1. Assuming the controls are physically close to one another, use the mouse to drag a selection rectangle around the four radiobuttons. Your selection may look something like Figure 4-2.

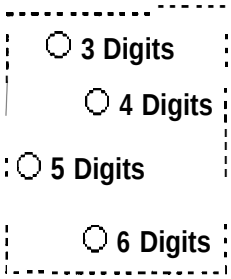


Figure 4-2: Radiobutton Selection

2. Choose the **Edit Align Options** command to open the **Align** dialog box. For this example, select the **Left** command in the **Align Objects By** section of the dialog box to align the radiobutton left sides.

3. If you also want the controls to be evenly spaced, choose the **Equal Spacing** command from the same section of the dialog box.

4. The vertical spacing of the controls can be set by either entering the increment amount into the **Vertical** edit box, or by using the up/down arrows to change the grid increment to the value desired. For this example, set the vertical increment to 5.

5. Select the **Align** command to exit the align dialog box and to align the controls.

The **Align Options** dialog box can be closed without changing the controls positions by selecting the **Ok** command. The controls can be aligned using the current align options by selecting the **Align** command from the **Edit** menu item or by using the F9 shortcut key.

Drawing Lines Around The RadioButtons

To set the group of radiobuttons apart from the rest of the front panel, draw lines around the four buttons and add a label.

1. From the **Controls** menu, select the Line control.
2. Using the mouse, position the cursor at the start point for the line and click and drag the control to the desired length.
3. Repeat the above line draws until you have something that looks like Figure 4-3.

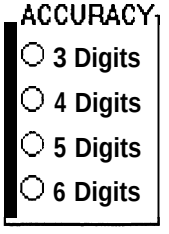


Figure 4-3: Radiobutton Group with Lines and a Label

4. The lines can be moved and resized the same as other controls, To resize, select one of the visible drag handles using the mouse and drag the end of the line to the new position.
5. Add a label to the line box so that your radiobutton group appears like the one shown in the figure above.

Adding a Range Selection to the Panel

The DM5010 panel should have a way to set the input range of the instrument. The input range is a selection of one-of-many functions. For this selection choose the **Listbox** control to provide range selection. The Listbox control provides a list of items to select from. Only one item on the list can be selected at any one time.

1. Create a list box control in the DCV view.
2. Set the control parameters to:

Name	"dcvrange"
Type	STRING
Label	RANGE

3. The **Listbox** control usually requires you to set the optional parameters for the control. In this example, set the text that appears in the list box control and set the text that is sent upon selection of the list box control text.

4. Open the options dialog box by selecting **ToScriptList** from the **Parameters** selection box.

5. Enter the following text into the **Display String** edit box. Figure **4-4** shows what the **Display String** edit box should look like when you are done.

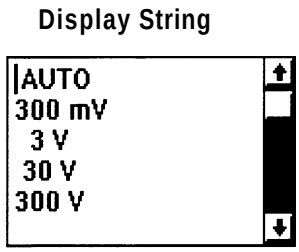


Figure 4-4: Display Strings of the Listbox Control

6. Next, you need to define the text that will be sent per each selection in the **Display String** edit box. Enter the text into the **ToScript String** edit box as shown in Figure 4-5.

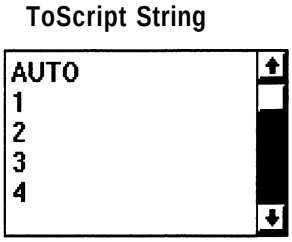


Figure 4-5: The Listbox Control's ToScript Strings

NOTE

There are no quote marks required. FPE adds the necessaty quotes when the script is generated.

7. For variety, set the **Listbox** control size by entering the number of rows and columns into the row and column edit boxes in the options dialog box:

   a. Enter **12** in the **Columns** edit box.

   b. Enter 4 in the **Rows** edit box.

c. Next, set the default position for the **Listbox** control to 1. This causes the list box to appear with the first item on the list selected when TekTMS/IPG utilizes the DC voltmeter panel.

d. Enter "1" in the **Default Position** edit box.

8. Choose the **Ok** command to close the options dialog boxes and return to the main menu. A list box control similar to the one shown in Figure 4-6 should appear.

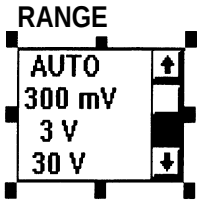


Figure 4-6: The Range List Box Control

The measurement readout needs to be updated whenever a range change is made. The **UpdateList** variable for the **Range** list box control is set to perform the update function.

9. Set the **UpdateList** for the **RANGE** list box control to include "dcvreading".

Adding a Filter Value for the Measurements

The DM5010 instrument has the capacity to average a number of readings before updating the measurement value. To do this, add an average enable switch and an input control to set the number of readings to average, as described in these steps:

1. Use the **Checkbox** control to enable/disable the average function and use the **Editbox** control to provide for the input of the number of samples to average.

2. Create these controls with the following control parameters set:

Checkbox control	
Name	`"dcvfilt"`
Type	`INTEGER`
Label	`Ave`
Editbox control	
Name	`"dcvfiltval"`
Type	`INTEGER`

The two controls should look something like Figure 4-7

Figure 4-7: The Average On/Off and Sample Number Controls

Making Control Groups

For the filter function of the DC voltmeter, you need the **Average On/Off** control and the **Filter Value** control to share the same code scenes. A ControlGroup associates the controls within the group to the same scene code.

1. Select the **Ave** check box and the filter input edit box into a common ControlGroup.
2. To make a ControlGroup, hold down the left mouse button and drag the mouse to create a box around the controls to be grouped. Figure 4-8 shows the controls selected.

Figure 4-8: Filter Controls Selected

3. Choose the **ControlGroup Make ControlGroup** command to place the two controls into the same control group, thus providing common setting and measure scene code. There is no confirmation of the command.
4. To see if a control is a member of a ControlGroup, first select the control, then choose the **ControlGroup Show ControlGroup** command. If there are other controls within the same ControlGroup, a selection rectangle is drawn around all the controls in that group.
5. To enable immediate updating of the DC voltmeter readout whenever the **Ave** control is activated, you need to add "dcvreading" to the **Ave** controls UpdateList.

The Go Button

You now have the controls to set the voltmeter range, accuracy, and filter mode and to readout the measured voltage. With these controls, there is no way to trigger a reading from the voltmeter without changing the voltmeter settings — youneed an acquire control. This control causes the DC voltmeter to place the current measurement into the readout control. Use the pushbutton control for this purpose.

1. Create the control and set the control parameters as listed.

Name	`"dcvacquire"`
Type	`INTEGER`
Label	`Acq`

2. The **Acq** pushbutton control requires that the "dcvreading" be added to its UpdateList. By adding "dcvreading" to the UpdateList each time the pushbutton control is activated, the readout is updated.

The Controls Code

Now that the front panel has a number of controls visible, the next step is to describe the purpose of each control. This section describes the scene code for each control using the TekTMS ISL language.

To define the code for the readout control:

1. Select the textbox control labeled **Measurement,** then choose the **Params Set Scene** and **Params Measurement Scene** commands. These commands open two edit windows, providing a way to edit the scene code for the selected control. You can also open the two edit windows by selecting **Set Scene** and **Measure Scene** from the **Parameters** selection box.

2. You can move to any control's scene code by selecting the control while the edit windows are open. The control's variable name is shown in the edit window title bar.

The scene edit windows have a main menu that applies to the edit session. The **Statements** menu item contains a shell for most of the ISD language constructs.

1. For this example, move the cursor into the measurement edit window and click the mouse. The edit window now has the focus as indicated bye a blinking cursor in the main window.

2. Choose the **Statements SendToInst** command. The text **cont->"";** appears in the scene window with the cursor between the two quote signs. Without moving the cursor enter "SEND".

3. Next, place the cursor on a new line and select the **Statements ReceiveFromInst** command. The text **inst->"";** appears in the window. Add a ";" to the text, then move the cursor before the first quote sign and enter the text "dcvreading,". The Measurement Scene code should look like the following:

```
cont-> "SEND";
inst->dcreading,";";
```

4. Choose the **Save** command to save the scene code for the **dcvreading** control. Use the **Save** command to save your scene code in subsequent code generation steps.

5. All the active controls have scene code. Each one of the following sub-steps describes the changes for an edit window for a given scene type and control name.

   a. Enter the SET block `dcvthree` control code as follows:

```
if (dcvthree) then
cont-> "DIGIT 3";
endif
```

   b. Enter the MEASURE block `dcvthree` control code as follows:

```
tempvar tmp:STR;
dcvthree = 0;
cont-> "DIGIT?"
inst-> "DIGIT",tmp,";";
if (tmp=="3") then
dcvthree = 1;
endif
```

   c. Enter the SET block `dcvfour` control code as follows:

```
if (dcvfour) then
cont-> "DIGIT 4";
endif
```

   d. Enter the MEASURE block `dcvfour` control code as follows:

```
tempvar tmp:STR;
dcvfour = 0;
cont-> "DIGIT?"
inst-> "DIGIT",tmp,";";
if (tmp=="4") then
dcvfour = 1;
endif
```

   e. Enter the SET block `dcvfive` control code as follows:

```
if (dcvfive) then
cont-> "DIGIT 5";
endif
```

   f. Enter the MEASURE block `dcvfive` control code as follows:

```
tempvar tmp:STR;
dcvfive = 0;
cont-> "DIGIT?"
inst-> "DIGIT",tmp,";";
if (tmp=="5") then
dcvfive = 1;
endif
```

g. Enter the SET block dcvsix control code as follows:

```
if (dcvsix) then
cont-> "DIGIT 6";
endif
```

h. Enter the MEASURE block dcvsix control code as follows:

```
tempvar tmp:STR;
dcvsix = 0;
cont-> "DIGIT?"
inst-> "DIGIT",tmp,";";
if (tmp=="6") then
dcvsix = 1;
endif
```

i. Enter the SET block dcvrange control code as follows:

```
cont-> "RANGE",dcvrange;
```

j. Enter the MEASURE block dcvrange control code as follows:

```
cont-> "RANGE?";
inst-> "",dcvrange,";";
```

k. Enter the SET block dcvfilt control code as follows:

```
if (dcvfilt==1) then
cont-> "FILTERVAL",dcvfiltval,";FILTER ON";
else
cont-> "FILTER OFF";
endif
```

l. Enter the MEASURE block dcvfilt control code as follows:

```
tempvar filton:STR;
cont-> "FILTER?;FILTERVAL?";
inst-> "",filton,";","",dcvfiltval,";";
if (filton=="ON") then
dcvfilt = 1;
else
dcvfilt = 0;
```

m. Enter the SET block dcvacquire control code as follows:

```
cont-> "DCV";
```

Setting the Bus

The instrument front panel is now described. This allows you to select the range, number of digits, and the sample filter value for a DC voltage measurement function using a DM5010. You must now indicate the type of bus connecting the DM5010. For this example, use the GPIB bus, as follows:

1. Choose the **Pararns BusNotes** command from the main menu. By selecting the **GPIB** command in the **Bus Selection** dialog box, you set the bus type to GPIB.
2. If you wish to set the GPIB parameters to something other than the default values, select the **Options** command to open the **GPIB Parameters** dialog box. For this instrument front panel, accept the default GPIB bus parameters.
3. Choose the **Ok** command to exit the **BusNotes** dialog box and return to the main menu.

ISD ASCII Listing

The FPE provides an easy path to an ASCII listing of the current script. To list the current script, choose **Options Open Notepad** command. The editor generates a temporary listing of the script and displays the script using the NotePad Windows application.

Checking for TekTMS/IPG Compatibility

You now have a complete script for making DC voltmeter measurements from a DM5010. To preserve the panel, you need to write it to an ISD file. To use the panel to control the DM5010, the TekTMS/IPG is required. The current version of FPE does not have any instrument interface capabilities, thus the requirement for TekTMS/IPG to prove the functionality of the front panel. The FPE does have the capability, however, of determining if the ISD script is readable by TekTMS/IPG.

To check if the ISD is readable, perform a compatibility check. Choose **Pararns Check ISD** command. The script is checked and a **Script Passed** message appears if the script is compatible. If there are errors, a message indicating the error appears.

Creating an AC Panel

If you wish to add an AC measurement instrument panel that has the same layout as the one you just built, you could copy all the controls in the first view and paste them into the second view.

After the controls are pasted to the second view, you must go through the controls and change their names and then edit the scene code to reflect the changed names. All controls must have unique names within and ISD file. This tutorial does not attempt to create a second panel, but you may wish to try this feature of the FPE.

Loading the DC Measurement Front Panel

To use the new front panel for developing test steps using the DM5010, you need to load TekTMS/IPG. Refer to the Interactive Procedure Generator User Manual to determine how to use the new instrument front panel. This concludes the FPE tutorial.

TekTMS/FPE Procedures

This section explains how to use a mouse with the Front Panel Editor. For many programs, using a mouse is considered a short cut method; however, for the FPE application, a mouse is necessary for a successful editing session.

Using a Mouse

In general, a mouse is used to point at and select items. The FPE application uses a mouse as do other Microsoft Windows applications.

You select menu items and commands with a mouse cursor and click (press the left button) to initiate an action.

Input focus or selection can be set within dialog boxes using a mouse.

In addition to the normal mouse actions, the FPE application uses a mouse to create, position, and select controls within a view.

Creation

Controls are created with a mouse by placing the mouse creation cursor over an empty spot in a view and then click-drag to control size. The type of control created is selected in the **Controls** menu item.

Position

The position of a control within the view is determined at creation by the position of the mouse cursor when you click the left button.

A control can be moved with the pointer cursor by first selecting the control with a left button click and, with the left button down, dragging the control to the new position.

Selection, One at a Time

A control is selected by positioning the mouse pointer cursor over a control and clicking the left mouse button. Control selection is shown by eight drag handles (small black squares) drawn around the selected control. This type of selection applies to a single control.

Selection of a Group

A group of controls can be selected by using the pointer cursor to draw a selection rectangle around the controls. To drag select, position the cursor on a corner of the controls you want to select, press the left mouse button, drag a selection box around the controls, and release the left mouse button

Double Click

A double click on a control within a view opens the **Controls Parameter** dialog box.

The double click on the control performs the same action as selecting a control and choosing **Params Variables** command.

Right Button

The right button selects between control creation mode and point mode. The mode switches when the cursor is over an empty part of the view and the right button is clicked. When switching to creation mode, the cursor changes to the creation cursor and the control that will be created is the selected control.

The right button can also switch off the duplicate and the copy/paste modes. The action is the same as selecting **Edit Duplicate Control** when Duplicate Control is enabled.

Shift + Left Button Click

Use the SHIFT+left button click to select controls within a view into a group. The SHIFT+left button click performs the same grouping as the left button drag selection of controls. SHIFT+left button click is the only way to select controls that are not physically adjacent into a common group. The group is indicated by a selection rectangle surrounding the selected controls. Once the group is selected, the commands that normally apply to a group of controls are available.

Creating and Editing Instrument Front Panels

The following sections describe some of the basic steps used to edit or create an ISD script using TekTMS/FPE. In general, creating a new ISD script requires the creation of front panel views, the controls within the views, checking for compatibility, and then saving the script to a file.

The FPE may also be used to modify an existing script. The script file is read and edited using the editor menu commands. The modified script is checked for compatibility as with a new script, and then the changes are saved in an ISD file.

Creating and Editing Views

Follow these steps when you want to create a new view for the front panel script:

1. Select the **View New View** command from the main menu and a new panel dialog box appears.
2. Enter the view name in the **New View** dialog box, then choose the Ok command when done. A new view window appears with the given view name as the window title. The view is ready for the creation of new controls within the view.

Other views can be selected by selecting the **Views Next View** command from the main menu, or by selecting the desired view name from the list in the **Views** menu item.

The PgDn and PgUp keys will move you to the next or previous view on the list. PgDn and PgUp are accelerator keys for moving from view to view.

Creating and Editing Controls

Follow these steps to create new controls in a front panel:

1. Select the type of control desired from the **Controls** toolbox or menu list. The mouse cursor changes to the tool cursor indicating that the FPE is in creation mode.
2. Move the cursor to the desired position in the current view and click and drag the creation rectangle to the desired size of the new control, then release the button. The control is drawn, replacing the creation rectangle upon release of the mouse button.

   The parameters for the control are available by double clicking the left mouse button over the main body of the new control, or by choosing the **Params Variables** command from the main menu.
3. Enter the control variable name and select the variable type. The control variable name and type are required for a valid script file.

The controls optional parameters are available through the **Parameter** drop-down dialog box. Each control type has a set of valid optional parameters. Only valid parameters are offered for selection.

After the control parameters are set, the scene code for the new control can be edited. The scene edit windows are available through the selection of **Params Setting Scene** or **Params Measurement Scene** command.

In the **Scene Edit** windows, edit the scene code using the Instrument Script Language (ISD). The script language is detailed in the Instrument Script Language Description topic in *Section* 6, *Instrument Script Language Descriptions.*

If there is a requirement for multiple controls to be associated with a common scene code block, a ControlGroup must be created to define the control association. A ControlGroup is created by first selecting the controls that make up the group and the selecting the **ControlGroup Make Controlgroup** command to create the control grouping.

The new control can be checked for TekTMS/IPG compatibility by selecting the **Params** menu item and choosing the **Check ISD** command. A "Script Passed" message appears if the full script is compatible. If there is an error, an error message will appear indicating the error.

Repeat the above steps as many times as required to build the instrument front panel.

Creating a Learn Scene

A Learn scene provides a way to create IPG test procedure steps that will reset an instrument to a previous known state. Follow these steps to create a learn scene:

1. Select the **Params Learn Scene** command from the main menu. The command selection opens the **Learn Scene** dialog box providing for the input of the Learn variable name and Learn type; either ASCII or Binary.

   An ASCII learn type contains the instrument control data in an ASCII format. If desired, you can modify the instrument command string using an ASCII editor. The Binary Learn type contains the instrument command string in a binary format that is not easily read.

2. To edit the code for the learn scene, select the **Scene** command in the **Learn Scene** dialog box. Selection of the **Scene** command opens a **Learn Scene** edit window. The **Edit** window provides menu items to assist in the editing of the learn scene code. The complete description of these menu commands can be found in the **Menu Commands** section of this menu.

3. Upon completion of the learn scene code, select the **Save** command and exit the **Edit** window and the **Learn Scene** dialog box.

Selecting the Bus

Follow these steps to select the bus and do the setup:

1. Select the **Params** menu item and choose the **BusNotes** command. A dialog box will appear providing a choice of GPIB, RS232, VXIinternal, VXI5520, CDS, and MXI buses. Select one bus in the dialog box.

2. Each bus has parameters that can be set. If the **Options** command is not selected, the default bus parameters will become the current bus parameters for the script. If you wish bus parameters other than the default values, select the **Options** command and enter the bus parameters in the **Options** dialog box that appears.

3. After all the parameters are set, select the **Ok** command to exit the **Parameters** dialog box.

Aligning Controls

Controls can be moved and resized in the current view by using a mouse. In addition to manually positioning controls there is a set of control alignment commands available in the Edit menu item.

To align controls, first select the controls to be aligned. Use either the mouse to drag-select the controls or the Shift + left mouse button to select the controls if they are not adjacent.

The selected controls will be aligned using the alignment parameters set in the Align Options dialog box. Open the Align Options dialog box by selecting the Edit Align Options command from the main menu.

The Align dialog box provides the means to select the alignment point, the increment used, and whether to equally space the controls. Select the appropriate options and the close the dialog box. If the dialog box is closed by selecting the Align command, the selected controls will be aligned using the set options. The Align Options dialog box may also be closed without aligning the controls by selecting the Ok command.

If the align options have already been set, then select the Edit Align command from the main menu to align the controls and return you to the main menu.

Duplicating and Copying Controls

You can generate multiple copies of a control either by duplicating a single control or by copying and pasting one or more controls. With duplication only a single control can be copied at a time. The **Copy**/**Paste** commands allow single or multiples of controls to be copied and pasted. The Duplicate command does not use the Windows clipboard to store the control, whereas the **Copy**/**Paste** command does use the clipboard.

The Duplicate command provides an easy and quick way to generate multiple copies of a control. Using the Control+D accelerator keys in conjunction with the mouse, you can copy a control many times with a minimal number of key/mouse strokes. The complete control and all its parameters are duplicated.

To duplicate a control, follow these steps:

1. Select the control using the mouse cursor and then click the mouse.
2. After the control is selected, select the Duplicate command by using the Control+D keys or by selecting the Edit Duplicate command from the menu. The Duplicate Control command is a toggle command, so selecting the command will select the opposite state. A check mark next to the Duplicate command indicates the duplicate mode is on; no check mark indicates the Duplicate mode is off.
3. Once the duplicate mode is on, use the mouse to position the cursor to the desired position to create the new control. While the duplicate mode is on, a placement rectangle appears. The placement rectangle tracks the cursor movement to show where the new control will be drawn when you press the left mouse button.

4. Repeat the key and mouse actions to make multiple duplications of the selected control.

5. The duplicate mode is turned off each time a new control is created. The duplicate mode can also be terminated by clicking the right mouse button, or by selecting the **Duplicate** menu command.

The **Copy**/**Paste** command differs from the **Duplicate Control** command in that it allows you to copy a group of controls. The complete control parameters, including its ControlGroup associations, are copied. A problem can arise after a Copy/Paste operation because all controls within a given script must have unique variable names. If a control or group of controls is copied to the same script, the variable names must be changed to provide a unique variable name for each control.

The **Copy**/**Paste** command can be used to move controls between ISD scripts and FPE sessions, because the command uses the Windows clipboard to store the copied controls. For example, a complete view from an ISD script can be copied, the script closed and another one opened, and then you can paste the controls into the newly opened ISD script.

To copy a control or group of controls, follow these steps:

1. Select the control or controls using the mouse. If copying a group of controls, drag a selection rectangle around the desired controls. If the controls are not physically grouped, use the Shift+Left button click to select the controls. The Selection rectangle will enclose all the controls in the group.

2. Select the **Edit Copy Control** command to copy the controls. Selecting this command puts the controls onto the Windows clipboard in a data format that only the FPE application understands. Once the controls are on the Windows clipboard, the **Paste** command is enabled.

3. Select the **Edit Paste Control** command to paste the control(s) to the selected view. The selection of the **Paste** command creates a placement rectangle that tracks the mouse and shows the boundaries of the new controls. Position the placement rectangle with the mouse and press the left mouse button to paste the new controls in the view.

4. If the controls are pasted into the same ISD, you will need to change the variable names so they are unique to have a valid ISD.

Building ControlGroups

A ControlGroup is an ISL block structure that groups controls with common settings and measurement scene code. The detailed description of the Controlgroup block can be found in the Instrument Script Language Description topic in Section 6, Instrument Script Language Descriptions.

To build a controlgroup, the member controls must first be selected using the mouse. The controls are selected by dragging a selection rectangle around the controls to be included in the controlgroup. Here are the steps to follow:

1. If the controls are not physically adjacent, use the Shift + left button to select the group of controls. Otherwise use the mouse selection method to select controls into a group.
2. Once all the desired controls are selected, choose the **ControlGroup Make ControlGroup** command from the main menu. The controls are now selected into a Controlgroup with common code scenes.
3. To show the controls that are in the same control group, first select a control using the mouse click.
4. Select the **ControlGroup Show ControlGroup** command from the main menu. The selection of this command causes a selection rectangle to be drawn around all the controls within the same controlgroup as the initial control.

A controlgroup can be broken up by selecting a control, showing the controlgroup and then choosing the **ControlGroup Break ControlGroup** command from the main menu.

Checking for TekTMS/IPG Compatibility

The primary goal of creating a front panel with the Front Panel Editor is to make a panel that is usable for generating test steps in TekTMS/IPG. An additional goal is to have interactive control of the instrumentation before the generation of the test steps. Therefore, the FPE-generated script must be compatible with TekTMS/IPG.

To insure compatibility, FPE provides a command that performs a check of the current script. To check a current script, choose the **Params Check ISD** command from the main menu. If the script is compatible, the message **Script Passed** is displayed. If there are errors in the script, an error message appears indicating the error.

For large scripts, the check process can take a considerable amount of time. The script is actually written out to a temporary file and then read back into the editor.

The compatibility check is performed at every **Save** command providing the assurance that all successfully saved sessions are compatible ISD files.

Opening a Notepad Session

The Windows NotePad application can be opened from the FPE application by selecting **Options Open Notepad** command. An ISD script file is generated and is displayed in the NotePad.

The **Open Notepad** command is provided to allow easy access to an ASCII edit session of the current script.

If the notepad application is already open, the **Open Notepad** command will not open an additional copy of the notepad. Close the current notepad session and choose the **Open Notepad** command to regenerate the current script in ASCII text form.

Opening a TekTMS/IPG Session

TekTMS/IPG can be opened from the FPE application by the selection of the **Options Start TekTMS** command.

TekTMS/IPG may then be used to load in the current script file. The current script changes will not be reflected in the IPG session unless the script has been saved and reloaded into IPG. The ISD is not automatically loaded when TekTMS/IPG is opened from within the FPE.

Instrument Script Language Descriptions

Script Basics

This section provides a detailed description of the block structure of the Instrument Script Language (ISL) shown in Figure 6-1. ISL is used in developing scripts. This section explains the interaction between blocks, the constructs used in blocks, construct syntax, and the ISL special functions.

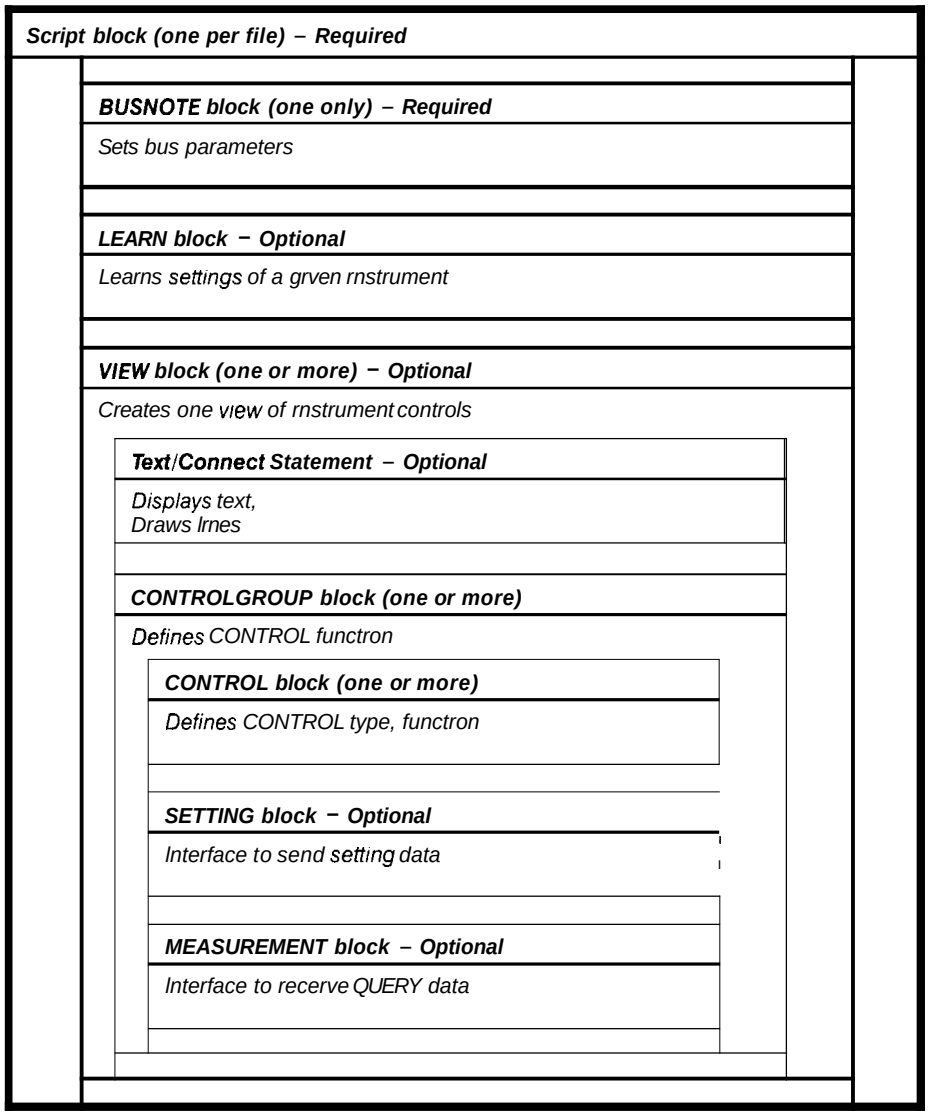


Figure 6-1: ISL Block Structure

The Framework

A script is a text file that can be created using a text editor or the front panel editor. The syntax is block structured: all blocks are identified by keywords. The termination of every block is marked with the keyword END. Related information is encapsulated in blocks, as shown in Figure 6-1 on the previous page. All items in a script, (e.g., keywords, variable names, etc.) are case insensitive; therefore, uppercase and lowercase may be mixed within an entry as desired (e.g., `script` is the same as `SCRIPT`).

General

A line beginning with the keyword **REM** or **REMARK** is treated as a comment. Each remark line must have REM or REMARK at its beginning.

A section may be indented and spaced in any style you want to improve readability. The syntax of the script is free format; i.e., any number of contiguous spaces, tabs, or new-lines are equivalent to a single space.

The following is a simple example of a script: it defines only one CONTROL in one view. A script can have many controls and many views, but, for introducing a complete script, this example is sufficient.

```
REM  THE FOLLOWING IS AN EXAMPLE OF A SIMPLE SCRIPT TO
REM  ILLUSTRATE THE VARIOUS BLOCKS USED IN A SCRIPT.
REM  NOT ALL BLOCKS ARE SHOWN IN THIS EXAMPLE.
REM  THE TEXT, CONNECT, AND LEARN BLOCKS ARE NOT USED IN
REM  THIS EXAMPLE. THEY WILL BE EXPLAINED IN OTHER
REM  EXAMPLES.

REM  YOU BEGIN WITH A 'SCRIPT' BLOCK. IT CONTAINS ALL
REM  SCRIPTDATA.

SCRIPT "EXAMPLE1"

 REM  NEXT IS THE BUSNOTE BLOCK. IT TELLS WHICH
 REM  COMMUNICATIONS BUS TO USE AND SETS VARIOUS
 REM  COMMUNICATIONS PARAMETERS.

 GPIB
  EOM = 10;
 END

 REM  IF A SCRIPT HAS ANY CONTROLS, A 'VIEW' BLOCK IS
 REM  REQUIRED.

 VIEW "CHANNELA"

  REM  EACH CONTROL IS IN A 'CONTROLGROUP' BLOCK. YOU CAN
  REM  PLACE MORE THAN ONE CONTROL IN A 'CONTROLGROUP'
  REM  BLOCK. YOU MUST HAVE AT LEAST ONE 'CONTROLGROUP'
  REM  IN A VIEW.
```

```
CONTROLGROUP
 REM  A CONTROL (THE GRAPHICAL PART) IS DEFINED IN A
 REM  'CONTROL' BLOCK. YOU ALSO DEFINE THE NAME OF THE
 REM  CONTROL VARIABLE, ITS DATA TYPE, AND ITS LOCATION
 REM  IN THE 'CONTROL' BLOCK.

 CONTROL FREQUENCYA:FLOAT EDITBOX @1,2

  REM  THE SIZE, TITLE, AND DEFAULT VALUES OF A CONTROL
  REM  ARE DEFINED WITHIN THE 'CONTROL' BLOCK.

  NUMROWS 1;
  NUMCOLS 12;
  CONTROLTITLE "FREQUENCY A";
  STRING "10000";
 END

 REM  A 'SETTING' BLOCK TAKES INFORMATION FROM THE
 REM  FRONT PANEL OR TEST PROGRAM FOR ALL OF THE C
 REM  ONTROL VARIABLES DEFINED IN A 'CONTROLGROUP'
 REM  BLOCK.

 SETTING
  CONT -> "FREQA: ",FREQUENCYA,";";
 END

 REM  A 'MEASUREMENT' BLOCK TAKES INFORMATION FROM THE
 REM  CONTROL VARIABLES DEFINED IN A 'CONTROLGROUP'
 REM  BLOCK AND RETURNS IT TO THE FRONT PANEL OR TEST
 REM  PROGRAM.

 MEASUREMENT
  CONT -> "FREQA?";
  INST -> "FREQA:",FREQUENCYA,";";
 END
END
END
END
```

The example shows some of the syntax points. All lines beginning with the keyword REM are comments.

Script Block

The SCRIPT block is the complete script (or it can be defined as the outermost block containing a script). The keywords for this block are **SCRIPT** and **END.** SCRIPT is followed by the script identifier. The script identifier is a character string enclosed within double quotes.

The SCRIPT block, in its entirety, communicates with the instrument as a driver. A script block consists of a number of internal blocks: the *BUSNOTE* block, the LEARN block, and the VIEW block. The LEARN block and VIEW block are optional, as shown in the following example script:

```
SCRIPT "SCRIPTID1"

 GPIB
  EOM = 13,10;
  EOI = 1;
  TIMEOUT = 1000;
 END
END
```

This example defines a complete script. There are no controls defined, but a simple script such as this one can be used to establish a communications path between a test program and an instrument for TRANSFER DATA steps in an IPG test procedure (i.e., VARIABLE TO INSTRUMENT or INSTRUMENT TO FILE transfers). Refer to Section 5, Advanced Applications, in the Interactive Procedure Generator Users Manual, under the Communicating with an Instrument which does not have a Driver *(.ISD)* File topic for additional information. The example script can also be used to gain access to the TALK/LISTEN function of a front panel. Refer to the *Talk/Listen* description in Section *4,* Menus, of the Interactive Procedure Generator User Manual, under the Instrument Front Panel Menu description, for further information.

BUSNOTE Block

The purpose of the BUSNOTE block is to select the bus type and advise the script interpreter of interface setting parameters for the corresponding bus. This subsection describes the parameters and the corresponding settings that can be made within the BUSNOTE block. Only one bus (GPIB, VXI, etc.) can be used. The valid keywords for a BUSNOTE block and their definitions are as follows:

- **GPIB** — National Instruments GPIB interface for the PC
- **VXIINTERNAL** — Tektronix and Colorado Data Systems embedded VXI controllers
- **VX5520** — Tektronix VX5520 GPIB interface and Resource Manager for VXI
- **RS232** — COM1 or COM2 ports on the PC
- **MXI** — National Instruments MXI PC to VXI interface.
- **CDSBUS** — Colorado Data System 53 Series embeded controllers.

The general syntax for a BUSNOTE block is shown in the following example:

```
GPIB

  EOM = number, number;
  EOI = 1 (YES) | 0 (NO);
  TIMEOUT = number;

END
```

Note that the equal sign (=) and the semicolon (;) are required in the syntax. There is a different set of parameters for each bus type. The parameters for each bus type and their description follows.

GPIB Parameters

EOM (End Of Message) — This parameter specifies a character or characters automatically appended to the end of a device-dependent message on output. The characters to be appended are the ASCII equivalent of the numbers specified in the argument. EOM characters found in the input from an instrument will terminate the Read. The End of Message syntax is as follows:

`EOM = n,m;`	0 <= *n* <= 255 0 <= *m* <= 255

The following are examples of EOM:

`EOM = 0;`	(turns off EOM function)
`EOM = 10;`	(LF termination)
`EOM = 13,10;`	(CR/LF termination)

The above examples illustrate three possibilities: zero (OFF); one character specified; and two characters specified.

EOI (End Of Input) — This parameter determines if EOI is asserted on the last byte of messages sent by the controller to the instrument, or by the instrument to the controller. The following examples show the use of EOI:

`EOI = 1;`	(EOI is asserted)
`EOI = 0;`	(EOI is not asserted)

TIMEOUT — This parameter is specified in milliseconds. It determines the wait period before a timeout is reported. Since some GPIB control software does not support a continuous range, the script interpreter will use the closest value that is at least as great as the specified value. The following example shows the use of TIMEOUT:

`TIMEOUT = 1E4;`	(TIMEOUT value of 10 seconds)

Default SETTINGs — The default settings for the parameters EOM, EOI and TIMEOUT are as follows:

```
EOM = 0;
EOI = 1;
TIMEOUT = 10000;
```

RS232 Parameters

```
RS232
 EOM = 0;
 BAUD = number; (300,600, 1200 ... 9600,19200)
 PARITY = 0 (EVEN) | 1 (ODD) | 2 (NONE) ;
 FLOWCONTROL = 0 (NONE) | 1 (XON/XOFF)) | 2 (DTR/DSR —
                hardware) | 3 (RTS/CTS);
 STOPBITS = 1 | 2 | 3 (is 1.5);
 DATABITS = 4 | 5 | 6 | 7 | 8;
END
```

Default SETTINGs — The default settings for the RS232 parameters are as follows:

```
BAUD = 1200;
PARITY = 2; (NONE)
FLOWCONTROL = 1; (XON/XOFF)
STOPBITS = 1;
DATABITS = 7;
EOM = 0;
```

VX5520 Bus Parameters

The parameters for the VX5520 Interface are the same as GPIB (substitute VX5520 for GPIB) as in the following example script:

```
VX5520
 EOM = 13,10;
 EOI = 1; (1 = YES, 0 = NO)
 TIMEOUT = 1E4; (TIMEOUT value of 10 seconds)
END
```

Default SETTINGs — The default settings for the VX5520 Bus parameters are as follows:

```
EOM = 0;
EOI = 1;
TIMEOUT = 10000;
```

NOTE

It is important to use a *BUSNOTE* type of *VX5520* if you are using a *VX5520* as the *VXI* Resource Manager and a *GPIB* interface to *VXI* instruments.

VXI Internal Bus Parameters

The parameters for VXI internal bus are shown in the following example:

```
VXIINTERNAL
 EOM = 0;
END
```

Default SETTINGs — The default settings for the VXI Internal Bus parameters are as follows:

```
EOM = 0;
```

MXI Bus Parameters

The parameters for MXI bus are shown in the following example:

```
MXI
 EOM = 10;
 TIMEOUT = 10000;
END
```

Default SETTINGs — The default settings for the MXI Bus parameters are as follows:

```
EOM = 0;
TIMEOUT = 10000;
```

CDS 53 Series Parameters

The parameters for CDS 53 Series parameters are shown in the following example:

```
CDSBUS
 EOM = 13,10;
 TIMEOUT = 5000;
END
```

Default SETTINGs — The default settings for the CDS 53 Series bus parameters are as follows:

```
EOM = 0;
TIMEOUT = 10000;
```

LEARN Block

The LEARN block provides a way to create IPG test procedure steps that will reset an instrument to a state you specify on subsequent executions of the test procedure. You do this simply by creating a LEARN step that contains a SETTING block and a MEASUREMENT block.

The SETTING block is executed when a test procedure with a LEARNED SETTING step is executed.

The MEASUREMENT block is executed when requested by the user during test program creation in IPG.

The LEARN SETTING block does not describe a graphical, interactive control, but is used to store the instrument state in a test program for later retrieval.

The following examples show the use of the LEARN block:

```
REM  THIS EXAMPLE SHOWS AN ASCII LEARN SETTING BLOCK FOR
REM  A TEKTRONIX DM5010 DIGITAL MULTIMETER. THERE IS NO
REM  GRAPHICAL DESCRIPTION, AS THIS CONTROL DOES NOT
REM  APPEAR ON THE SCREEN.

LEARN LEARNSTR:STR ASCII
 SETTING
  REM  SEND THE ENTIRE LEARNED STATE STRING TO THE
  REM  INSTRUMENT.
  CONT -> LEARNSTR;
 END
 MEASUREMENT
  REM  QUERY THE INSTRUMENT FOR ITS ENTIRE SETTING STATE
  REM  AND STORE THE DATA IN THE CONTROL VARIABLE
  REM  LEARNSTR.
  CONT -> "SET?";
  INST -> LEARNSTR;
 END
END

REM  THIS EXAMPLE SHOWS A BINARY LEARN SETTING BLOCK FOR
REM  A TEKTRONIX DM5010 DIGITAL MULTIMETER. THERE IS NO
REM  GRAPHICAL DESCRIPTION, AS THIS CONTROL DOES NOT
REM  APPEAR ON THE SCREEN.

LEARN LEARNSTR:STR BINARY
 SETTING
  REM  SEND THE ENTIRE LEARNED STATE STRING TO THE
  REM  INSTRUMENT.
```

```
  CONT -> LEARNSTR;
 END
 MEASUREMENT
  REM  QUERY THE INSTRUMENT FOR ITS ENTIRE SETTING STATE
  REM  AS A BINARY STRING (THIS IS SHORTER THAN AN ASCII
  REM  STRING) AND STORE THE DATA IN THE CONTROL VARIABLE
  REM  LEARNSTR.
  CONT -> "LLSET?";
  INST -> LEARNSTR;
 END
END
```

The difference between an ASCII LEARN block and a BINARY LEARN block, is that you can edit the contents of the ASCII string when creating a LEARNED SETTING step, but you cannot edit the contents of a BINARY string.

When one of the example scripts is used, the MEASUREMENT block gets a measurement from an active instrument and places it into the variable specified in the script. When a LEARN SETTING step is created in the IPG test procedure and the test procedure is run, only the SETTING block is executed, allowing you to set the instrument.

The LEARN block is used with the **Learn SETTING...** menu item from the **Step** menu selection in the IPG Instruments Front Panel (for further information, refer to the Instrument Front Panel Menu description in Section ***4,*** Menus, of the IPG Users Manual).

VIEW Block

The keywords for a VIEW block are **VIEW** and **END.** The keyword VIEW is followed by the *VIEW* identifier. The VIEW identifier is a character string enclosed within double quotes. In a script, there may be any number of VIEW blocks. One VIEW block corresponds to one logical instrument view.

The VIEW block consists of a number of statements and blocks. The first entries may be **TEXT** or **CONNECT** statements (or both). The blocks in the VIEW block are **CONTROLGROUP** blocks.

A logical instrument view contains graphical controls such as PUSHBUTTONS, CHECK BOXES, etc. These graphical controls allow interaction with the hardware and the bus.

There are several approaches to assigning functions to different views (The issue of fully functional vs. application-oriented front panels is explained in Section 1, General Information.):

- Don't put too many controls into one View as it creates the same problems as when actual hardware front panels have too many controls in a small space. Keep the View simple.

  A fully functional or general purpose approach might parcel views by the actual partitioned functionality of the instrument front panel (e.g., a DSO might have a Channel 1 view, a Channel 2 view, a Timebase view, etc.).

- An application-oriented approach might put all the functions needed to perform a particular action in one view, and those for another action in another view. (e.g., in a DSO, all the functions required to acquire a waveform on channel 1 might be presented in one view, and all those needed to acquire a waveform on channel 2 might be presented in another view. The time base and trigger functions would be included as required and duplicated on the two views).

  On an interactive front panel, a group of measurements might be placed in a different view from control settings so that when the measurements are updated, the controls are not updated.

- Implementation of a set of controls might be simplified by using several views. For example, if the DM5120 had one range control that controlled the ranges for all functions, it might be included in a view for each function (DCV, ACV, OHMS, etc.), and be given an appropriate label corresponding to the function associated with the view.

- Don't build a view that requires the entire display screen (maximized).

Multiple Views

Multiple views in the same script should use approximately the same surface area for the view layout to preclude the user resizing the window to see all of the controls each time a new view is selected. When views are not the same size, the first view (default) should be the largest so the user doesn't accidently miss controls that might be outside the window of the initial display area. Also any controls that are duplicated from view to view should be laid out in the same relative position.

Display Types

If an instrument front panel has been developed for an EGA display and it fills a large part of the display, it may not be completely displayed on a CGA display. Also, a VGA display provides a smaller amount of display area than a comparable EGA display. The user should be aware that different graphics adapters provide different display resolutions, and if a front panel is to be used on systems with different graphics adapters, it should be designed for the system with the lowest resolution (smallest display area).

View Layout

All controls must fit within a 27 character high by 80 character wide area (Instrument Script Language character coordinates). This assures that all controls are visible on the smallest available window display, which is the 640 x 480 VGA mode that provides 640 x 410 pixels in the white space of a maximized front panel window. A 640 x 480 VGA mode displays characters in 8W x 15H cells, which reduces the number of rows of characters that can be displayed. Controls should be spaced far enough apart on a front panel so that they are perceived as separate controls. Each control should be surrounded by several characters of space to promote "readabil-

ity" of the panel. You may find it easier to layout the front panel to scale on graph paper before implementing it in a script. The units for front panel layout are the number of characters from the top left corner of the display.

Coordinate System

The IPG software uses character spaces to set coordinates. Starting from the top left corner of your display, the coordinates are 0,0. They move right horizontally to the maximum number of columns; and down vertically to the maximum number of rows. Coordinates can be entered into the script, or can be function values. The coordinates can also be decimal numbers. For example, you can place a control at 3.3,4.5, if desired. You can use several digits of precision if you desire, but tenths of character sizes are usually adequate.

TEXT or Connect Statement

The TEXT or **CONNECT** statement (either is started with a separate keyword) can appear in a VIEW block. A TEXT or CONNECT statement is used to display text or to draw lines on the screen of the logical instrument front panel view. The body of each statement may contain one or more sets of TEXT and CONNECT points. TEXT or CONNECT statements must come before CONTROLGROUP blocks.

Following are examples of TEXT and CONNECT statements:

```
TEXT "text to be displayed" @ 5,5;
CONNECT (5,10),(10,10);
```

This example defines text to be displayed beginning at character spaces (5, 5); and specifies drawing a line between display coordinates (character space points) 5,10 and 10,10.

More than one TEXT or CONNECT statement can be included in a VIEW block.

NOTE

Notice that these statements require a terminating semicolon.

TEXT — The following syntax displays one line of text:

```
TEXT "stringoftext" @ x,y;
```

As an example:

```
TEXT "THIS IS TEXT" @ 5, 10;
```

This example causes THIS IS TEXT to be displayed, beginning at character space location 5, 10.

Connect — The following syntax draws a line between end points:

```
CONNECT (X1, Y1), (X2, Y2) [, (X3, Y3), (X4, Y4) [,...]];
```

where the x and y values of the points are given in character spaces. They are relative to the upper left hand corner of the user window, where x = 0, and y = 0.

Examples:

```
CONNECT (5,10),(10,10);
```

This example draws a line between the two points: (5, 10) and (10, 10).

```
CONNECT (3,3),(10,3),(10,10),(3,10),(3,3);
```

This example causes a line to be drawn from the point (3, 3) to (10, 3) then to (10, 10) to (3, 10) and then back to (3, 3). The result is a rectangle.

CONTROLGROUP Block

A CONTROLGROUP block is a collection of at least one CONTROL block and, optionally, one SETTING and one MEASUREMENT block that define the control functions in an instrument view. The CONTROLGROUP must contain at least one CONTROL block, but it may have more than one. A CONTROLGROUP block without either a SETTING or MEASUREMENT block will not be able to send data to or receive data from an instrument. Each CONTROLGROUP block specified has the ability to define a SETTING and/or a MEASUREMENT block to perform when the control is manipulated interactively or used in a test program. In a SETTING block, data is taken from the front panel or test procedure and sent to the SETTING block. In a MEASUREMENT block, data is gathered from the instrument and returned to the front panel or test procedure.

It is important to remember that SETTING blocks accept data and MEASUREMENT blocks return data. SETTING blocks do not update the front panel control, while MEASUREMENT blocks do not use the value of the front panel control in the measurement.

Another important consideration about scripts is that they do not execute sequentially. They do not start at the first line and execute to the last line of the script. Each control definition will execute either the SETTING or MEASUREMENT block independently, depending on the level of front panel to instrument interaction, type of instrument step executed, and which controls are manipulated.

The following is an example of a CONTROLGROUP block with CONTROL, SETTING, and MEASUREMENT blocks defined:

```
CONTROLGROUP
 CONTROL FREQUENCYA:FLOAT EDITBOX A2,2
  CONTROLTITLE "FREQUENCY A";
  STRING "10000";
  NUMCOLS 12;
  NUMROWS 1;
```

```
  END
  SETTING
   CONT -> "FREQA: ",FREQUENCYA,";";
  END
  MEASUREMENT
   CONT -> "FREQA?";
   INST -> "FREQA: ",FREQUENCYA,";";;
  END
 END
```

In this example, when the front panel is in "full Interaction" mode, and the ENTER keyboard key is press when the input focus is on the EDITBOX control, the SETTING block will send the string `FREQA: 10000;` to the instrument. If the control is saved as a SETTING step in a test procedure, and the value of the control was `10000` when the step was saved, the same string will be sent to the instrument when this step is executed. The value of the CONTROL variable `FREQUENCYA` will be set to `10000` and the SETTING block will be executed.

If the front panel is in "full interaction" mode when the ENTER key is pressed, the SETTING block is executed, and then the MEASUREMENT block is executed which updates the value of the EDITBOX to reflect the value returned from the instrument in response to the `FREQA?` query. If the response from the instrument is `FREQA: 10000.0;`, then the front panel EDITBOX will be updated with the value 10000.0.

More detailed information about the various blocks and instrument dialogs is presented later in this section. For further information about the front panel window, refer to the Interactive Procedure Generator Users Manual.

Control Block

Since a CONTROL block describes only one control, each control on an instrument front panel display must have a separate CONTROL block. The first line of a CONTROL block contains the keyword CONTROL followed by a CONTROL ID (control variable), a colon (:), a variable type (INTeger (a long), FLOAT (a double), or STRing), the control type (TEXTBOX, EDITBOX, etc.), and the controls reference position (relative X, Y coordinate).

A CONTROL variable name is limited to 15 alphanumeric characters, and must begin with an alpha character. The following example defines a TEXTBOX control at location `2,20` that accepts and shows integer data. `COUNTA` is the name of the CONTROL.:

```
CONTROL COUNTA:INT TEXTBOX @2,20
```

Control ID

Once you declare a **Control ID,** it becomes a global variable that can be accessed by any subsequent SETTING or QUERY block (MEASUREMENT block) in the script. A common use of this global variable feature is for defining a control that can display an error message for an operator using

the front panel interactively. The following example script makes use of this feature by defining a control to display an error message that is defined and implemented by another control.

```
SCRIPT "SCRIPT1"
 GPIB
 END
 VIEW "VIEW1"
  REM  DEFINE THE ERROR DISPLAY CONTROL.
  CONTROLGROUP
   CONTROL ERRMSG:STR TEXTBOX @ 1, 2
    TITLE "ERROR MESSAGE";
    NUMROWS 1; NUMCOLS 20;
   END
   QUERY
   REM  FORCE AN UPDATE OF THE CONTROL VALUE.
   ERRMSG = ERRMSG;
   END
  END
  CONTROLGROUP
   CONTROL TWO:INT CHECKBOX @ 3, 5
    STRING "CLICK HERE";
    REM  CAUSE THE ERRMSG QUERY BLOCK TO EXECUTE.
    UPDATELIST ERRMSG;
   END
   SET
    REM  GIVE THE ERROR MESSAGE A VALUE.
    ERRMSG = "ERROR 0";
   END
  END
 END
END
```

In this example, a TEXTBOX control to display an error message is declared using the Control ID ERRMSG and front panel TITLE "ERROR MESSAGE". This error message display control must be the first control defined in the VIEW or script, so that any errors detected in subsequent steps can be detected.

Next, a CHECKBOX control using the Control ID TWO and front panel title "Click Here" is declared to cause the system to locate the MEASUREMENT step for ERRMSG, the measurement to be taken, and execute the assignment of the error message and the updating of the TEXTBOX display.

The QUERY block is required., since **UPDATELIST** only updates blocks that contain a QUERY block. If it is missing, the control will not be updated by the UPDATELIST ERRMSG statement.

Script Design Considerations — There is a restriction that must be considered when one control sets or reads the value of the Control ID of another control. When writing a script, a Control ID can be set only if it has been previously defined with a CONTROL statement earlier in the script.

CONTROLGROUP Types

Each CONTROL block defines a single control, and defines the variable and variable type used for communications with that control.

There are several types of front panel controls. The CONTROL block defines one of the controls and control data types listed in Table 6-1.

Table 6-1: CONTROL Block

CONTROL	CONTROL Data Type
TEXTBOX	INT, STR, FLOAT
EDITBOX	INT, STR, FLOAT
LISTBOX	INT, STR
CHECKBOX	INT, STR
RADIOBUTTON	INT, STR
PUSHBUTTON	INT, STR
WAVEFORMDISPLAY	STR, WAVE

The INTeger value for a CHECKBOX, RADIOBUTTON, or PUSHBUTTON is either 1 (ON) or 0 (OFF). The string value for these controls can be any string necessary for the function of the control SETTING or MEASUREMENT blocks.

Data values for a TEXTBOX or EDITBOX can be any value required by the control function.

The INTeger value for a LISTBOX is the index of the selected LISTBOX item. The indices run from 1 to the number of entries in the LISTBOX. The string value of the LISTBOX is either the string displayed in the LISTBOX, or the corresponding TOSCRIPT string.

CONTROLGROUP Type Construct Definitions — Table **6-2** defines the control constructs used in the following CONTROLGROUP type discussions.

Table 6-2: Construct Definitions

Name	Syntax	Description
CONTROL-TITLE	CONTROLTITLE "titlestring";	titlestring is the string to be displayed above the TEXTBOX, EDITBOX, or LISTBOX (no default)
DEFAULTPOS	DEFAULTPOS n;	n is the default selected row in a LISTBOX (default 1)
NUMCOLS	NUMCOLS n;	n is the number of columns wide of the TEXTBOX, EDITBOX, or LISTBOX (default 10)

Table **6-2:** Construct Definitions (Cont.)

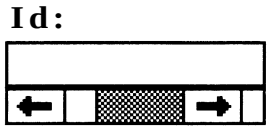
Name	Syntax	Description
NUMROWS	NUMROWS n;	n is the number of rows high of the TEXTBOX, EDITBOX, or LISTBOX (default 1)
ONEOFGROUP	ONEOFGROUP "*groupname*"	groupname is the name of the group of controls to which a CHECKBOX or RADIOBUTTON belongs (no default)
SCROLLHORZ	SCROLLHORZ *YES*<N>|*NO*;	YES specifies a horizontal scroll bar for a TEXTBOX, EDITBOX, or LISTBOX; NO specifies no horizontal scroll bar (default NO)
SCROLLVERT	SCROLLVERT *YES*<*N*>|NO;	YES specifies a vertical scroll bar for a TEXTBOX, EDITBOX, or LISTBOX; NO specifies no vertical scroll bar (default NO)
STATE	STATE *ON*<N>|OFF;	The default state of a CHECKBOX or RADIOBUTTON (default OFF)
STRING	STRING "datastring";	datastring is the string to display in the TEXTBOX, EDITBOX, or LISTBOX; or the string to be displayed as the label of the CHECKBOX, PUSHBUTTON, or RADIOBUTTON (no default)
TITLE	TITLE "titlestring";	titlestring is the string to be displayed above the TEXTBOX, EDITBOX, or LISTBOX (no default)
TOSCRIPT	TOSCRIPT "string,["string",[...]]	string is the string that will be placed in the CONTROL variable (~) in a LISTBOX
TOSCRIPTOFF	TOSCRIPTOFF "offstring";	offstring is a string that will be placed in the CONTROL variable when the CHECKBOX or RADIOBUTTON is turned OFF or deactivated (no default)
TOSCRIPTON	TOSCRIPTON "datastring";	datastring is a string that will be placed in the CONTROL variable when the PUSHBUTTON in manipulated (no default)
	TOSCRIPTON "onstring";	*onstring* is a string that will be placed in the CONTROL variable when the CHECKBOX or RADIOBUTTON is turned ON or activated (no default)

Table 6-2: Construct Definitions (Cont.)

Name	Syntax	Description
UPDATELIST	UPDATELIST *controlid* [,*controlid* [, ...]];	controlid is the name of another control in this view to update when the control whose script contains the UPDATELIST is manipulated. To update a control, the MEASUREMENT block actions for the control whose script contains the UPDATELIST are executed. (no default)

TEXTBOX

A **TEXTBOX** is a rectangular area in which text is displayed and can be used for measurement displays on a front panel. If you do not need to modify the measurement value reported from the instruments, use a TEXTBOX. The following illustration is an example TEXTBOX graphic.



The following script was used to create the example TEXTBOX graphic:

```
CONTROLGROUP
 CONTROL ID:STR TEXTBOX @ 5, 10
  NUMROWS 1;
  NUMCOLS 10;
  SCROLLHORZ YES;
  CONTROLTITLE "ID:";
 END
 MEASUREMENT
  CONT ->"ID?";
  INST ->ID;
 END
END
```

In the above example, a CONTROL block with a TEXTBOX is defined with its number of columns and rows. The TEXTBOX has a horizontal scroll bar and a control title `ID:`; and is empty. When the front panel is updated, the TEXTBOX will display the contents of the control variable ID, which is the output returned by the MEASUREMENT block. The variable and type are declared with the control type (in the above example `CONTROL ID:STR TEXTBOX @ 5, 10`).

The available constructs in a TEXTBOX are:

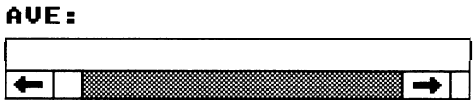
CONTROLTITLE
NUMCOLS
NUMROWS
SCROLLHORZ
SCROLLVERT
STRING
TITLE

The definitions of these constructs are given in Table 6-2, in this section.

The variable type can be STR, INT, or FLOAT.

EDITBOX

An **EDITBOX** is a rectangular area in which you can enter and edit text and can be used for setting and reporting numeric values for an instrument (i.e., the frequency of a signal generator). The following illustration is an example of the EDITBOX graphic.



The following script was used to create the example EDITBOX graphic:

```
CONTROLGROUP
 CONTROL NUM:FLOAT EDIT @ 10, 10
  NUMROWS 1;
  NUMCOLS 23;
  CONTROLTITLE "AVE:";
  SCROLLHORZ YES;
 END
 SETTING
  CONT ->"AVE",NUM,";";
 END
 MEASUREMENT
  CONT ->"AVE?";
  INST ->"AVE",NUM,";";
 END
END
```

The example defines an EDITBOX with its number of columns and rows. It has a horizontal scroll bar and the title AVE: displayed above the box. Initially, the EDITBOX is empty. The variable and type are declared with the control type (in the example, `CONTROL NUM:FLOAT EDIT @ 10, 10`).

When the SETTING block is evaluated, the number in the EDITBOX is assigned to the CONTROL variable NUM. The number is encoded with the string constant and sent to the instrument. For example, if NUM = 100, then "AVE 100" is sent.

When the MEASUREMENT block is evaluated, the number to be displayed in the EDITBOX is taken from the CONTROL variable NUM. The instrument is queried for the value of AVE and its string in the form "AVE NUM";. The value of NUM is extracted from the instrument response.

The available constructs for an EDITBOX are:

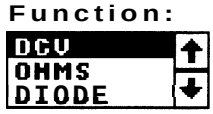
CONTROLTITLE	SCROLLVERT
NUMCOLS	STRING
NUMROWS	TITLE
SCROLLHORZ	UPDATELIST

The definitions of these constructs are given immediately following the RADIOBUTTON description, in this section.

The variable type can be STR, INT, or FLOAT.

LISTBOX

A LISTBOX is a scrollable list of strings representing instrument controls or measurements and can be used to give the front panel user a choice of a limited number of settings for an instrument. The following illustration is an example of a LISTBOX graphic giving a choice of operating modes for a digital multimeter:



The following script is used to create the example LISTBOX graphic:

```
CONTROLGROUP
 CONTROL PARAM1:STR LISTBOX @ 5, 10
  NUMROWS 3;
  NUMCOLS 8;
  CONTROLTITLE "FUNCTION:";
  SCROLLVERT YES;
  STRING "DCV","OHMS","DIODE","ACV","ACDC";
 END
 SETTING
  CONT ->PARAM1;
 END
 MEASUREMENT
  CONT ->"FUNC?;
  INST ->"FUNC",PARAM1,";";
```

```
  END
 END
```

The example defines a LISTBOX with the number of columns, rows, a vertical scroll bar, a title FUNCTION and the strings displayed in the LISTBOX. The variable and type are declared with the control type (in the example, CONTROL PARAM1:STR LISTBOX @ 5, 10).

When the SETTING block associated with a LISTBOX is evaluated, the string selected in the LISTBOX is assigned to the control variable PARAM1.

When the MEASUREMENT block is evaluated, the string to be highlighted will be taken from the control variable PARAM1. If the variable does not match any string in the STRING block, no string in the LISTBOX will be highlighted. The match is case sensitive.

The available constructs for a LISTBOX are:

CONTROLTITLE	SCROLLVERT
DEFAULTPOS	STRING
NUMCOLS	TITLE
NUMROWS	UPDATELIST

And, if the control type is STR:

TOSCRIPT

The definitions of these constructs are given immediately following the RADIOBUTTON description, in this section.

The variable type can be STR or INT.

PUSHBUTTON

A Pushbutton is a box that contains a string representing an action and is used for actions that require no response (i.e., the initialization function of an instrument). A Pushbutton has only a SETTING block, and when it is manipulated (pushed) is always in the ON state.

Following is an example of a PUSHBUTTON graphic:

The following script was used to create the example PUSHBUTTON graphic:

```
CONTROLGROUP
 CONTROL INIT:STR PUSHBUTTON @5,15
  STRING "INIT";
  TOSCRIPTON "INIT";
 END
 SETTING
  CONT ->INIT;
 END
END
```

This example script defines a PUSHBUTTON that will have the text INIT inside it. The variable and type are declared with the control type (in the example, CONTROL INIT:STR PUSHBUTTON @5,15).

When the SETTING block associated with the PUSHBUTTON is evaluated, the string defined by TOSCRIPTON is assigned to the control variable.

In the following alternative example, the PUSHBUTTON uses a control variable of the type INT.

```
CONTROLGROUP
 CONTROL INIT:INT PUSHBUTTON @5,15
  STRING "INIT";
 END
 SETTING
  CONT ->"INIT"
 END
END
```

This example script also defines a PUSHBUTTON that will have the text INIT inside it. Again, the variable and type are declared with the control type (in the example, CONTROL INIT:STR PUSHBUTTON @5,15). This example takes advantage of the fact that when a PUSHBUTTON is manipulated, the value of the control variable is always 1, and simply sends the constant string INIT to the instrument.

A PUSHBUTTON should not have a MEASUREMENT block. If it does, the MEASUREMENT block will not be evaluated.

The available constructs for a PUSHBUTTON are:

STRING
UPDATELIST

And, if the control type is STR:

TOSCRIPTON

The definitions of these constructs are given in Table 6-2, in this section.

The variable type can be STR or INT.

CHECKBOX

A **CHECKBOX** is a small square with an identifying character string to the right. When a CHECKBOX is activated, an "X" appears in the square. A CHECKBOX is used to indicate the state of a control, either ON or OFF. A CHECKBOX might be used to indicate whether the output of a signal generator is turned ON or OFF.

Following is an example of a CHECKBOX graphic:

The following script was used to create the example CHECKBOX graphic:

```
CONTROLGROUP
 CONTROL LFR:STR CHECKBOX @10,20
  STRING "LFR";
  TOSCRIPTON "LFR ON";
  TOSCRIPTOFF "LFR OFF";
 END
 SETTING
  CONT ->LFR;
 END
 MEASUREMENT
  CONT ->"LFR?";
  INST ->LFR,";";
 END
END
```

This example defines a CHECKBOX with the text LFR displayed next to it. The variable and type are declared with the control type (i.e., CONTROL LFR:STR CHECKBOX @ 10,20).

When the SETTING block is evaluated, if the CHECKBOX is activated, the string defined by the key word **TOSCRIPTON** is assigned to its control variable; if the CHECKBOX is not activated, the string defined by the key word **TOSCRIPTOFF** is assigned to the control variable.

When the MEASUREMENT block is evaluated, the control variable will be used to update the state of the CHECKBOX. If the control variable matches the string defined by the key word **TOSCRIPTON,** the CHECKBOX will be activated; if it does not match, the CHECKBOX will not be activated.

The following alternative example of a CHECKBOX script uses a control variable of the type INIT:

```
CONTROLGROUP
 CONTROL LFR:INT CHECKBOX @5,15
  STRING "LFR";
 END
 SETTING
  IF (LFR == 1) THEN
   CONT -> "LFR ON";
  ELSE
   CONT -> "LFR OFF";
  ENDIF
 END
 MEASUREMENT
  TEMPVAR TEMPSTR:STR;
  CONT -> "LFR?";
  INST -> TEMPSTR,";";
  IF (TEMPSTR == "LFR ON") THEN
   LFR = 1
  ELSE
   LFR = 0
  ENDIF
```

```
  END
END
```

This example also defines a CHECKBOX with the text LFR displayed next to it. The variable and type are declared with the control type (i.e., `CONTROL LFR:INT CHECKBOX @ 5,15`).

This example takes advantage of the fact that when a CHECKBOX is manipulated, the value of the control variable is 1 if the CHECKBOX is activated, and 0 if it is not activated. The SETTING block checks the value of the control variable and sends the appropriate command to the instrument. The MEASUREMENT block checks the value of the query and sets the control variable to either 1 (ON) or 0 (OFF). Notice that the MEASUREMENT block uses a temporary string variable to help with the comparison.

The available constructs for a CHECKBOX are:

ONEOFGROUP
STATE
STRING
UPDATELIST

And, if the control type is STR:

TOSCRIPTOFF
TOSCRIPTON

The definitions of these constructs are given immediately in Table 6-2, in this section.

The variable type can be STR or INT.

RADIOBUTTON

A **RADIOBUTTON** is a small circle with a character string to the right of the circle. When selected, the circle contains a black dot. RADIOBUTTONs are used only in groups to represent mutually exclusive selections. A RADIOBUTTON is used to indicate the state of a control, either ON or OFF. A group of RADIOBUTTONs show the state of mutually exclusive instrument controls or states. For example, a group of RADIOBUTTONs could be used to show which function of a function generator is currently active (i.e., sine, square, triangle, etc.).

Following is an example of the RADIOBUTTON graphic:

The following script was used to create the example RADIOBUTTON graphic:

```
CONTROLGROUP
 CONTROL LFR:INT RADIOBUTTON @10,20
  STRING "LFR";
```

```
  REM  RADIOBUTTONS MUST CONTAIN A ONEOFGROUP DEFINITION.
  ONEOFGROUP "FREQ";
 END
 SETTING
  IF (LFR == 1) THEN
   CONT ->"LFR ON";
  ELSE
   CONT ->"LFR OFF";
  ENDIF
 END
 MEASUREMENT
  TEMPVAR FROMINST:STR;
  CONT ->"LFR?";
  INST ->FROMINST;
  IF (FROMINST == "LFR ON;") THEN
   LFR = 1;
  ELSE
   LFR = 0;
  ENDIF
 END
END
```

The example defines a RADIOBUTTON with the text `LFR` next to it. The variable and type are declared with the control type (i.e., `CONTROL LFR:INT RADIOBUTTON @ 10, 20`).

When the SETTING block is evaluated, if the RADIOBUTTON is ON, a `1` is assigned to the control variable; if the RADIOBUTTON is OFF, a `0` is assigned to the control variable.

When the MEASUREMENT block is evaluated, the control variable is used to update the state of the RADIOBUTTON. If the output variable is `1`, the RADIOBUTTON will be ON; otherwise, the RADIOBUTTON will be OFF.

NOTE

The *RADIOBUTTON* being turned *OFF* has its *SETTING* block executed first; then the *SETTING* block of the button being turned *ON* is executed.

The available constructs for a RADIOBUTTON are:

ONEOFGROUP
STATE
STRING
UPDATELIST

And, if the control type is STR:

TOSCRIPTOFF
TOSCRIPTON

The definitions for these constructs are on the following page.

The variable type can be STR or INT.

If a RADIOBUTTON is type STRING (STR) then the TOSCRIPTON and TOSCRIPTOFF constructs must be used.

WAVEFORMDISPLAY

The **WAVEFORMDISPLAY** control shows a variable of the type WAVEFORM and gives you control over how the waveform is acquired and viewed.

The ISD Script Language contains two functions, WAVEFORMTOADIF and WAVEFORMTOVAR, to convert most instrument specific data formats into a WAVEFORM variable that can be used and displayed by TekTMS. The WAVEFORMDISPLAY control does not appear on the screen, but appears as an icon at the bottom of the screen when an instrument view containing a waveform control is shown. The display can be viewed by expanding the icon. If no waveform has been captured the display field is blank. For further information regarding waveform display, refer to the *Interface Program Generator Users Manual.* The contents of the variable cannot be modified by the display.

The data type of a WAVEFORMDISPLAY control is either WAVE or STR. The variable and type are declared with the control type. If you use the data type of WAVE you must use the function WAVEFORMTOVAR to convert the instrument specific data. If you use the data type of STR, you must use the function WAVEFORMTOADIF to convert the instrument specific data. The control string is the name of the file that contains the acquired waveform data. The preferred type to use is WAVE. The STR type is mainly for compatibility with previous versions of TekTMS. With the WAVE type you can also capture XY or Envelope data from an instrument and put it into the WAVE variable.

Unlike other controls, the position specified for the WAVEFORMDISPLAY graphic is relative to the upper left hand corner of the front panel, since the graphic is a separate window rather than part of the front panel.

Figure 6-2 is an example of a WAVEFORMDISPLAY graphic.

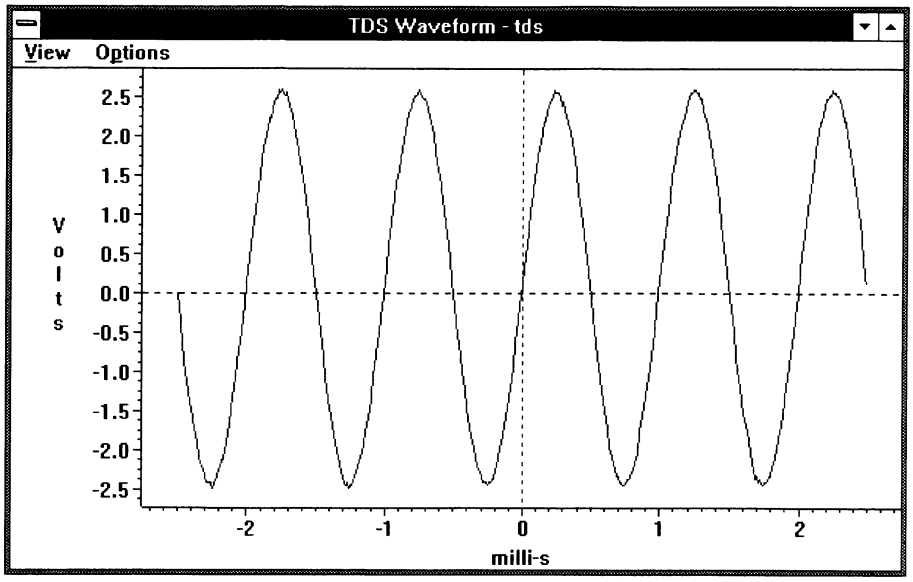


Figure 6-2: Example of a WAVEFORMDISPLAY Graphic

The only available construct for a WAVEFORMDISPLAY control is CONTROL-TITLE or TITLE.

The following formulas are used to calculate the real value of the point:

- Implicit dimension: value[*n*] = scale * n + offset where n goes from 0 to points−1.
- Explicit dimension: value[*n*] = scale * data[*n*] + offset where data[*n*] is nth data point on that dimension.

WAVEFORMTOVAR Function — The following paragraphs describe how to use the WAVEFORMTOVAR function in IPG.

The WAVEFORMTOVAR function converts raw digitized waveform data into a waveform variable.

WAVEFORMTOVAR takes a number of parameters that determine the actions taken by the routine. The SCRIPT invocation is shown in the following example:

```
WAVEFORMTOVAR (INFILE,INFORMAT,WAVEORDER,INTYPE,
    XSCALE,XOFFSET,YSCALE,YOFFSET,POINTS,BITS,BYTES,
    BYTEORDER,READOFFSET,BYTEFORMAT,XLABEL,YLABEL);
```

The Parameters are defined as follows:

- **INFILE** — Astring containing the name of the input raw digitized waveform data.
- **INFORMAT** — An integer which defines the input format. The possible formats are:

0 ASCII format

1 Binary Signed format.

2 Binary unsigned format.

- **WAVEORDER** — An integer indicating the order that data is stored in the input file — this is used only for xy and env inTypes.

  0 by Tuple (the data is organized so that the data points are listed together i.e. x1y1x2y2)

  1 by Dimension (the data for the first dimension is given then the next)

- **INTYPE** — An integer which defines the type of waveform input. The possible formats are:

  0 yt (normal waveform – explicit data for only one dimension)

  1 xy (explicit data for both dimensions)

  2 env (two explicit dimensions with the same implicit dimension — max value first)

  3 env (two explicit dimensions with the same implicit dimension – min value first)

- **XSCALE** — A floating point number giving the X scale factor used to convert the raw data into usable information.

- **XOFFSET** — A floating point number giving the X offset to add to the raw data.

- YSCALE — A floating point number giving the Y scale factor used to convert the raw data into usable information.

- **YOFFSET** — A floating point number giving the Y offset to add to the raw data.

- **POINTS** — An integer number giving the number of data points in one dimension.

- **BITS** — An integer giving the number of bits of data per data point. This is typically the resolution of the instrument (i.e. 8 for an 8 bit digitizer, 10 for a 10 bit digitizer, 16 for Tektronix 11400 Series digitizers).

- **BYTES** — An integer giving the number of bytes per data point in the input data. The possible values are:

  1 one byte per data point.

  2 two bytes per data point.

  4 four bytes per data point. Either integer or floating point format.

  8 eight bytes per data point. Floating point format only.

- **BYTEORDER** — An integer that specifies the order of the bytes in the data point when the input format is binary. This field is used not used when there is only one data byte per point. The possible values are:

0 Least Significant Byte first. (Swapped bytes).

1 Least Significant Byte last.

- **READOFFSET** — An integer giving the number of bytes to skip at the beginning of the input before attempting to read the waveform data points. This allows you to skip header information which as 'CURVE %nn' as is common with Tektronix digitizers.
- **BYTEFORMAT** — an integer which specifies the format of the input data if the input format is binary. The possible values are:

  4 integer format.

  8 floating point.
- **XLABEL** — A string containing the label for the X axis.
- **YLABEL** — A string containing the label for the Y axis.

The following WAVEFORMDISPLAY control script is from a Tektronix TDS540 DSO script:

```
CONTROLGROUP

REM THIS CONTROL IS OUTPUT ONLY, SO IT ONLY HAS A
REM MEASUREMENT BLOCK.

  CONTROL WFMDISPL:WAVE WAVEFORMDISPLAY @ 2.43, 2.00
    CONTROLTITLE "TDS WAVEFORM";
  END

  MEASUREMENT

    REM DEFINE ALL OF THE TEMPORARY VARIABLES USED BY THE
    REM WAVEFORMTOVAR FUNCTION.

  TEMPVAR RAWFILE:STR,INMODE:INT,INTYPE:INT,XSCALE:FLOAT,
  XOFF:FLOAT,YSCALE:FLOAT,YOFF:FLOAT,YZERO:FLOAT,
  NUMPTS:INT,BITS:INT,BYTES:INT,BORDER:INT,ORDER:INT,
  READOFF:INT,BFORMAT:INT,TRIGPTS:INT,XLABEL:STR,
  YLABEL:STR;

  REM SET THE INPUT FILE NAME. DATA IS READ FROM THE
  REM TDS540 INTO A DATA FILE AND THE
  REM RESULT OF THE WAVEFORMTOVAR FUNCTION
  REM IS A WAVEFORM VARIABLE.
  REM GIVE THE RAW DATA FILE ANY LEGAL NAME THAT DOES
  REM NOT CONFLICT WITH NAMES TO BE USED IN TEKTMS.
  REM THE VARIABLE WFMDISPS IS A CONTROL VARIABLE AND
  REM CONTAINS THE STRING GIVING THE INPUT SOURCE.
  REM THIS HAS BEEN SET BY A SETTING STEP OR
  REM FRONT PANEL UPDATE.

  RAWFILE = "~TDS.WAV";

  REM SETUP THE TDS AND QUERY THE INSTRUMENT FOR THE DATA
  REM REQUIRED TO CREATE THE WAVEFORM.
```

```
    CONT->"DATA:SOURCE ",WFMDISPS,";";
    CONT->"DATA:ENCDG ASCII;";
    CONT->"WFMPRE:", WFMDISPS, ":XUNIT?;";
    INST->"\"",XLABEL,"\"";
    CONT->"WFMPRE:", WFMDISPS, ":XINCR?;";
    INST->XSCALE;
    CONT->"WFMPRE:", WFMDISPS, ":YUNIT?;";
    INST->"\"",YLABEL,"\"";
    CONT->"WFMPRE:", WFMDISPS, ":YMULT?;";
    INST->YSCALE;
    CONT->"WFMPRE:", WFMDISPS, ":YZERO?;";
    INST->YZERO;
    CONT->"WFMPRE:", WFMDISPS, ":YOFF?;";
    INST->YOFF;
    CONT->"WFMPRE:", WFMDISPS, ":NR_PT?;";
    INST->NUMPTS;
    CONT->"WFMPRE:", WFMDISPS, ":PT_OFF?;";
    INST->TRIGPTS;

    REM INPUT FORMAT IS ASCII,INTYPE IS YT, NUMBER OF BITS
    REM IS 8. BORDER AND BFORMAT CAN HAVE ANY VALUE.

    INMODE = 0;
    INTYPE = 0;
    ORDER = 0;
    BITS = 8;
    BYTES = 1;
    BORDER = 1;
    BFORMAT = 4;

    REM ASCII DATA STARTS AT THE BEGINNING OF THE FILE.

    READOFF = 0;

    REM CALCULATE THE OFFSETS BASED ON DATA FROM THE TDS.

    XOFF=(XSCALE*TRIGPTS)*-1;
    YOFF=YZERO-(YSCALE*YOFF);

    REM QUERY FOR THE WAVEFORM DATA AND WRITE IT TO A FILE.

    CONT->"CURVE?;";
    INST->RAWFILE:%F;

    REM ASSIGN THE RESULT OF THE WAVEFORMTOVAR FUNCTION TO
    REM THE CONTROL VARIABLE.

    WFMDISPL = WAVEFORMTOVAR(RAWFILE,INMODE,ORDER,INTYPE,
       XSCALE,XOFF,YSCALE,YOFF,NUMPTS,BITS,BYTES,BORDER,
       READOFF,BFORMAT,XLABEL,YLABEL);

    END
END
```

Note that the variable type is WAVE and that no ADIF file is written at this time. The Control can not be defined as a STR. To get an ADIF file you need to do a Variable to file transfer step using the waveform variable assigned when a meaurement is done using this control.

WAVEFORMTOADIF FUNCTION — The following paragraphs describe how to use the **WAVEFORMTOADIF** function in IPG.

The WAVEFORMTOADIF function converts raw digitized waveform data into ADIF format data. It takes an input waveform file and outputs an ADIF format data file containing the waveform. This release supports ADIF *Version* 1.00.

WAVEFORMTOADIF takes a number of parameters which determine the actions taken by the routine. The script invocation is shown in the following example:

```
WAVEFORMTOADIF (INFILE,INFORMAT,OUTFILE,OUTFORMAT,
    XSCALE,XOFFSET,YSCALE,YOFFSET,POINTS,BITS,BYTES,
    BYTEORDER,READOFFSET,BYTEFORMAT,XLABEL,YLABEL);
```

The parameters are defined as follows:

- **INFILE —** Astring containing the name of the input raw digitized waveform data.
- **INFORMAT** — An integer which defines the input format. The possible formats are:

  0 ASCII format.

  1 Binary signed format.

  2 Binary unsigned format.

- **OUTFILE** — A string containing the name of the output ADIF file.

  OUTFORMAT — Aninteger which defines the output format. The possible formats are:

  0 ASCII format.

  1 Binary format. All ADIF binary formats are signed.

  The parameter OUTFORMAT is not used in Version 2.5. All ADIF files are written in binary format (32 bit floating point). This parameter is included for compatibility with previous versions of TekTMS.

- **XSCALE —** A floating point number giving the X scale factor used to convert the raw data into usable information.
- **XOFFSET —** Afloating point number giving the X offset to add to the raw data.
- **YSCALE** — A floating point number giving the Y scale factor used to convert the raw data into usable information.
- **YOFFSET —** Afloating point number giving the Y offset to add to the raw data.

- **POINTS —** An integer giving the number of data points in the raw input data file.
- **BITS —** An integer giving the number of bits of data per data point. This is typically the resolution of the instrument (i.e., 8 for an 8 bit digitizer, 10 for a 10 bit digitizer, 16 for the Tektronix 11400 Series digitizers).
- **BYTES —** An integer giving the number of bytes per data point in the input data. The possible values are:
  - 1 one byte per data point.
  - 2 two bytes per data point.
  - 4 four bytes per data point. Either integer or floating point format.
  - 8 eight bytes per data point. Floating point format only.
- **BYTEORDER —** An integer that that specifies the order of the bytes in the data point when the input format is binary. The possible values are:
  - 0 Least Significant Byte first. (Swapped bytes.)
  - 1 Least Significant Byte last.
- **READOFFSET** — An integer giving the number of bytes to skip at the beginning of the input file before attempting to read waveform data points. This allows you to skip header information such as "CURVE %NN" as is common with Tektronix digitizers.
- **BYTEFORMAT —** An integer that specifies the format of the input data if the input format is binary. The possible values are:
  - 4 integer format.
  - 8 floating point.
- **XLABEL —** A string containing the label for the X axis.
- **YLABEL —** A string containing the label for the Y axis.

The following WAVEFORMDISPLAY control script is from the Tektronix 2430A DSO script:

```
CONTROLGROUP

  REM  THIS CONTROL IS OUTPUT ONLY, SO IT ONLY HAS A
  REM  MEASUREMENT BLOCK.

  CONTROL WFM:STR WAVEFORMDISPLAY @2.5,2.0
   CONTROLTITLE "2400 WAVEFORM";
  END

  MEASUREMENT

   REM  DEFINE ALL OF THE TEMPORARY VARIABLES USED BY THE
   REM  WAVEFORMTOADIF FUNCTION.
```

```
TEMPVAR RAWFILE:STR,ADIFFILE:DTR,INMODE:INT,
        OUTMODE:INT,XSCALE:FLOAT,XOFF:FLOAT,
        YSCALE:FLOAT,YOFF:FLOAT,NUMPTS:INT,BITS:INT,
        BYTES:INT,BORDER:INT,READOFF:INT,BFORMAT:INT,
        XLABEL:STR,YLABEL:STR;

REM  SET THE INPUT AND OUTPUT FILE NAMES. DATA IS READ
REM  FROM THE 2430A INTO A DATA FILE AND THE RESULT OF
REM  THE WAVEFORMTOADIF FUNCTION IS A DATA FILE. GIVE
REM  THE RAW DATA AN EXTENSION OF .WAV AND THE ADIF
REM  FILE AN EXTENSION OF .WFD.

RAWFILE = "2400.WAV";
ADIFFILE = "2400.WFD";

REM  SET THE INPUT DATA FORMAT FOR THE RAW WAVEFORM
REM  DATA AS BINARY SIGNED AND THE OUTPUT AS BINARY

INMODE = 1;
OUTMODE = 1;

REM  SET THE INPUT BYTE SIZE, BYTE NUMBER, ORDER, AND
REM  FORMAT.

BITS = 8;
BYTES = 1;
BORDER = 1;
BFORMAT = 4;

REM  SET THE READ OFFSET TO 3 BYTES (SKIP THE %NN AT
REM  THE BEGINNING OF THE DATA FILE).

READOFF = 3;


REM  SET THE DATA RETURN FORMAT TO JUST NUMBERS AND
REM  THEN QUERY THE INSTRUMENT FOR THE DATA REQUIRED TO
REM  CREATE THE ADIF FORMAT DATA.

CONT -> "PATH OFF;LONG OFF;";
CONT -> "DATA ENC:RIB;WFMPRE? XIN,XUN,PT.0,YUN,YMU,YOF,
          NR.P";

REM  READ THE SCALE, OFFSET, NUMBER OF DATA POINTS, AND
REM  AXIS LABELS.

INST -> XSCALE,","XLABEL,","XOFF,","YLABEL,","
          YSCALE,","YOFF,","NUMPTS;

REM  SCALE THE OFFSET CORRECTLY.
```

```
    XOFF = XOFF*XSCALE*-1;
    YOFF = YSCALE*YOFF*-1;

    REM  QUERY FOR THE WAVEFORM DATA AND WRITE IT TO A
    REM  FILE.

    CONT -> "CURVE?";
    INST -> RAWFILE:%F;

    REM  TURN THE RAW INSTRUMENT DATA INTO AN ADIF FILE FOR
    REM  USE BY THE WAVEFORMDISPLAY AND PULSE PARAMETER
    REM  FUNCTIONS.

    WAVEFORMTOADIF (RAWFILE,INMODE,ADIFFILE,OUTMODE,XSCALE,
                    XOFF,YSCALE,YOFF,NUMPTS,BITS,BYTES,
                    BORDER,READOFF,BFORMAT,XLABEL,YLABEL);
    REM  ASSIGN THE WAVEFORMDISPLAY CONTROL VARIABLE THE
    REM  NAME OF THE FILE CONTAINING THE JUST CAPTURED AND
    REM  CONVERTED WAVEFORM.

    WFM = ADIFFILE;

    END
  END
```

Notice that the variable type is STR and that the variable is assigned the name of the ADIF file. The type of IPG variable associated with this control would be WAVEFORM, but in the script, it is treated as a string.

Control Type Summary

Table 6-3 is a quick reference guide showing the available control type and parameters:

Table 6-3: Control Type Summary

	CHECK BOX	EDIT BOX	LIST BOX	PUSH BUTTON	RADIO BUTTON	TEXT BOX	WAVE FORM
CONTROLTITLE		X	X			X	X
DEFAULTPOS			X				
NUMCOLS		X	X			X	
NUMROWS		X	X			X	
ONEOFGROUP	X				X		
SCROLLHORZ		X				X	
SCROLLVERT		X	X			X	

Table 6-3: Control Type Summary (Cont.)

	CHECK BOX	EDIT BOX	LIST BOX	PUSH BUTTON	RADIO BUTTON	TEXT BOX	WAVE FORM
STATE	X				X		
STRING	X	X	X	X	X	X	
TOSCRIPT			X				
TOSCRIPTOFF	X			X	X		
TOSCRIPTON	X			X	X		
UPDATELIST	X	X	X	X	X		X

SETTING and MEASUREMENT Blocks

SETTING blocks and **MEASUREMENT** blocks are alike. Both consist of dialogs and statements. The IPG procedures can only send data to a SETTING block and can only receive data from a MEASUREMENT block. These blocks must be located at the end of their CONTROLGROUP blocks, and the SETTING block must precede a MEASUREMENT block

A SETTING or MEASUREMENT block can contain actions that are either dialogs or statements. The following is a list of dialogs and statements:

Dialogs

CONT – > Controller Dialog
INST – > Instrument Dialog

Statements

Assignment	General purpose expression evaluation and variable assignment
IF THEN ENDIF	IF (condition) THEN (actions) ENDIF
IF THEN ELSE ENDIF	IF (condition) THEN (actions) ELSE (actions) ENDIF
WHILE DO END	WHILE (condition) DO (actions) END
GPIB commands	Low level GPIB bus commands
Variable declarations	Definitions of variables local to a SETTING or MEASUREMENT block
Function call	Calls to various ISL functions

More information on dialogs and statements is provided in the following sections of this manual.

The dialogs and statements in a SETTING or MEASUREMENT block are executed in order, starting at the beginning of the block. In some applications, the sequential execution of all dialogs and statements may not be desired. An IF-THEN-ELSE statement can be used within a SETTING or MEASUREMENT block to control statement execution.

IF Conditional Structure

The syntax for an IF conditional structure is as follows:

```
IF condition THEN dialogs OR statements;
ELSE dialogs OR statements;
ENDIF
```

OR

```
IF condition THEN dialogs OR statements;
ENDIF
```

The IF Statement allows decision processing. When the condition result in an IF statement is TRUE, the dialogs or statements in the initial IF statement are executed. When the condition result is FALSE, the dialogs or statements in the ELSE portion of the construct are executed, or, in the case of no ELSE, the dialogs or statements in the IF construct are bypassed without execution. Therefore, the IF Conditional Structure would be used to choose between alternative actions, based on the result of the condition.

Dialogs and statements are the heart of a script. SETTING and MEASUREMENT blocks use the dialogs and statements to communicate with and execute commands for instrument operation from the controller. Specific instrument address information comes from the **Instrument Select** information entered when an ISD file is selected. Communications parameters come from the BUSNOTE section of the script.

Condition

The syntax for condition could be any one of the following:

```
(EXPRESSION)
(EXPRESSION RELATIONAL OPERATOR EXPRESSION)
(STRING VARIABLE RELATIONAL OPERATOR STRING VARIABLE)
```

NOTE

Expressions used in these conditions must evaluate to a zero (FALSE) or a nonzero (TRUE) value.

Where EXPRESSION is a mathematical expression and RELATIONAL OPERATOR is a relational operator (==, >, <, >=, <=, or <>). A STRING VARIABLE is either a string constant or a variable name of type STR.

A condition is TRUE if it evaluates to a non-zero value, and FALSE if it evaluates to zero. A comparison of a string to a another string evaluates to TRUE if they are identical and FALSE if they are not. The comparison is case sensitive.

Dialogs

Dialogs are used for controller and instrument instructions. Each dialog type is designed to carry out a different kind of task.

There are two dialog types: Controller Dialog and Instrument Dialog. For each dialog type there is a unique keyword.

CONT **–>** Controller dialog

INST **–>** Instrument dialog

A dialog is limited to using CONT and INST.

Controller Dialog — prepares and sends data, or the contents of a file, to the instrument. The string to be sent is built by evaluating each OUTPUT-EXPRESSION and appending the results to form the string. This string is then sent to the instrument. The text for a controller dialog consists of one or more OUTPUT_EXPRESSIONS. The format for the controller dialog is as follows:

```
CONT -> OUTPUT-EXPRESSION [,OUTPUT-EXPRESSION[,....]];
```

Where OUTPUT-EXPRESSION contains:

- ***STRING constants*** — One or more ASCII characters enclosed in double quotes. "ABC", "VOLTS", and "ID?" are examples of string constants. The double quotes are required around the string. A string without quotes is interpreted as a variable reference.

  Variables — FREQUENCYA, CH1VOLTS, and DMMVALUE are examples of variables.

- ***Variable:%format*** — CH1VOLTS:%6.3G and DMMVALUE:%8.2E are examples of variables with format specifiers.

- ***Expressions*** — Any valid mathematical calculation, string concatenation, or function call. 2*VOLTS, VOLTSDIV/4.3, and VALUEQUERY & "VOLTS" are examples of expressions.

- The ***commas*** separating the OUTPUT_EXPRESSIONS and the ending semicolon are required.

- ***Function calls*** — You can use the SKIP(N) or the CHR(N) functions in a controller dialog.

Following are some examples of controller dialogs:

```
CONT -> "VOLTS?";
```

This example will send the string "VOLTS?" to the specified instrument.

```
CONT -> "CH1 VOLTS: ",VOLTS:%8.2G,";",;
```

In this example, if the variable VOLTS contained the number 12.34, the string "CH1 VOLTS: 12.34;" will be sent to the specified instrument. Notice that there are leading spaces before the number 12.34. The 8 in the dialog is the number of spaces in the width of the format, and the 2 is the

number of digits after the decimal point. The format type G tells the controller dialog to output a floating point number. The :% characters indicate that the following character is a format descriptor.

If the variable VOLTS contained the number 12.3456, the string sent would be "CH1 VOLTS: 12.35;". Notice the leading spaces and the rounding of the number to fit the specified format.

More information on format types and format modifiers is given later in this section.

The following is an example of a function call in a controller dialog:

```
CONT -> "MODE BINARY ",CHR(43),CHR(45);
```

The controller dialog in this example will send the string "MODE BINARY +-" to the specified instrument.

Instrument Dialog — reads data from the instrument. The format for an instrument dialog is as follows:

```
INST -> INPUT-EXPRESSION [INPUT-EXPRESSION [,....]];
```

Where an INPUT-EXPRESSION contains:

- ***STRING constants*** — One or more ASCII characters enclosed in double quotes. "ABC", "VOLTS", and "ID?" are examples of string constants. The double quotes are required around the string. A string without quotes is interpreted as a variable reference. String constants in an instrument dialog are used to indicate exact string matches in the response from the instrument. When a string constant is found in an instrument dialog, the string read from the instrument is scanned for a match with the constant. If a match is found, the rest of the dialog takes data starting with the character after the matched string. If a match is not found, an error is reported.
- ***Variables*** — FREQUENCYA, CHIVOLTS, and DMMVALUE are examples of variables.

  Variable:%format — CH1VOLTS:%6.3G and DMMVALUE:%8.2E are examples of variables with format specifiers.

  Expressions — String concatenation. Such as VALUEQUERY & "VOLTS" is allowed.
- ***Function calls*** — You can use the SKIP(N) or CHR(N) functions in an instrument dialog.

The commas separating the INPUT-EXPRESSIONS and the ending semicolon are required.

Following are some examples of instrument dialogs. In each example, the instrument string read is:

```
CH1 VOLTS:12.345;CH1 AMPS:0.2345;CH1 TERM:50;
```

Where VOLTS and AMPS are FLOATS and TERM is an INT.

```
INST -> "VOLTS:",VOLTS;
```

In this example, the instrument dialog scans the string read from the instrument from the string "VOLTS:", then reads the following characters as a floating point number. The conversion of characters from a string into a number will stop when the ";" character after the number 12.345 is read. The variable VOLTS will contain the value 12.345 after the instrument dialog is executed.

```
INST -> "AMPS:",AMPS:%6.3G;
```

In this example, the instrument dialog will scan the string read from the instrument for the string "AMPS:", then read the following characters as a floating point number with a width of 6 characters and no more than 3 characters after the decimal point. The variable AMPS will contain the value 0.234 after the instrument dialog is executed. The next character that would be processed is the ";", because the width specifies a format of 6 characters. The character 5 is skipped.

```
INST -> "VOLTS:",VOLTS,"AMPS:",AMPS,"TERM:",TERM;
```

In this example, the instrument dialog will scan the string read from the instrument for three different strings and extract the VOLTS, AMPS, and TERM values from the string.

The following is an example of a function call in an instrument dialog:

```
INST -> SKIP(20),DMMVALUE;
```

In this example, the instrument dialog will read a string from the instrument, skipping the first 20 characters, then place the remainder of the data into the variable DMMVALUE using the default format for the data type declared for DMMVALUE.

You can use the **SKIP** characters function in an instrument dialog to skip over unneeded characters. For example, if an instrument returns an ASCII binary representation of switch closures, then you could skip over unwanted characters to get to the one character that indicated whether or not the switch was open or closed. If an instrument with 16 switches returned a string of the format WORDF:B #B0000000000000001;, to indicate that Switch 1 was closed and all other switches were open, you could read just the Switch 1 setting with the MEASUREMENT block using the SKIP characters function in the following example:

```
CONTROLGROUP
 CONTROL SWITCH1:INT CHECKBOX @5,3
  STRING "1";
 END
 SETTING
  CONT -> "CLOSE:F1.A1"
 END
 MEASUREMENT
  CONT -> "WORDF:BINARY; WORD?";

  REM  SKIP OVER THE 'WORDF:B #B' HEADER AND THE FIRST
```

```
    REM  15 SWITCH SETTINGS TO GET TO THE DIGIT NEEDED.
    REM  READ IT AS A DECIMAL NUMBER SO THAT 1
    REM  (CLOSED/TRUE) OR 0 (OPEN/FALSE) IS RETURNED.

    INST -> SKIP (25),SWITCH1;%1D;
  END
END
```

Statements

There are five types of statements you can use in programming SETTING and MEASUREMENT blocks. These types are:

Temporary variable declarations
GPIB commands
Assignment statements
IF statements
WHILE statements

Temporary Variables — are the variables specified in the **TEMPVAR** statement. These variables can be used to store values during the execution of a SETTING or MEASUREMENT block. Variables declared with the TEMPVAR statement are local to the SETTING or MEASUREMENT block where they are declared and can not be accessed by another SETTING or MEASUREMENT block. This contrasts with control variables, which are global in a script. Temporary variables are used where it is desirable to temporarily store data without declaring a separate CONTROL block.

TEMPVAR is the temporary variable statement keyword, specifying a list of variables that are made available in the SETTING or MEASUREMENT block. The syntax of the TEMPVAR statement is:

```
TEMPVAR VARNAME:VARTYPE (,VARNAME:VARTYPE);
```

In the previous syntax definition, the statement type is TEMPVAR and VARNAME is the variable name. VARNAME is limited to 15 characters. VARNAME is followed by a colon (:), followed by VARTYPE, which is the variable type (i.e., INT, FLOAT, or STR), and must be specified for each VARNAME and separated from VARNAME by a comma. Following is an example:

```
TEMPVAR VOLTS:INT, MODE:STR;
```

In this example, the TEMPVAR statement declares variables named VOLTS, of the type <F1>INT,<F1> and MODE, of the type STR.

GPIB Commands — are meaningful only when the script is configured for an IEEE 488 compatible bus. These commands can be included in scripts that use a communications bus other than GPIB, but no action will be taken when they are executed. A GPIB command in a script with a VXIINTERNAL, MXI, CDSBUS, or RS232 BUSNOTE is treated as a NOOP (no operation) statement.

The syntax for a GPIB command is:

```
INTERFACEMSG [PARAM];
```

The statement TEXT for a GPIB command consists of one or more of the INTERFACEMSG messages in Table 6-4. A parameter may follow the INTERFACEMSG, if required. The INTERFACEMSG messages referred to here have been specified in IEEE 488.

Table 6-4: GPIB Commands

Command (InterfaceMsg)	Parameter (param)	Description
ATN	n	A single byte of the value n sent to the bus with attention asserted.
ATN	"STRING"	A string constant is sent to the bus with attention asserted.
DCL		Device *CLear* command sent out on the bus.
GTL		Go To Local is sent to the device.
GET		Group Execute Trigger.
IFC		*InterFace* Clear line is pulsed.
LLO		Local Lockout command is sent over the bus.
REN	TRUE	Remote *ENable* is asserted if TRUE.
REN	FALSE	Remote *ENable* is dropped if FALSE.
SDC		Selective Device Clear command is sent to the device.
TIM	num	The TIMEOUT value is redefined to be num (in milliseconds). This overrides the value specified in the Note section or the one specified by an earlier TIM command for the duration of the block containing the TIM command.
UNL		*UNListen* command is sent out on the bus.
UNT		*UNTalk* command is sent out on the bus.

Following are some GPIB command examples:

```
REN TRUE;
ATN 10; (The number here represents an ASCII character.)
SDC;
IFC;
TIM 10;
```

You can perform a serial poll with the ISL special function **SERIALPOLL.** SERIALPOLL allows the user to create a GPIB instrument front panel control to poll and display instrument status. The format is:

```
X = SERIALPOLL();
```

Where x can be any integer variable.

Following is an example using SERIALPOLL to create a TEXTBOX to display status whenever an **Update!** menu item is selected in an IPG test procedure, or the automatic Refresh Rate mode is selected:

```
REM DISPLAY INSTRUMENT STATUS ON FRONT PANEL UPDATE.

CONTROLGROUP
 CONTROL STATUS:INT TEXTBOX @2,3
  TITLE "STATUS";
 END
 MEASUREMENT
  STATUS = SERIALPOLL();
 END
END
```

Assignment Statement — allows evaluation of any expression. An expression is created from one or more of the elements that have a value combined to represent a new combined value. Another way of explaining an expression is to describe it as one side of an algebraic formula. Several things to note about expressions are:

Expressions can be very simple (one element) or extremely complicated.

- Usually, all values are separated from each other by an operator.

  Using meaningful names makes the expression much easier to understand.

- Values can be of different types, and the proper combination of these types is necessary for correct results.

The Assignment Statement has the following syntax:

```
VAR = expression;
```

Table 6-5 lists the available operators for expressions.

Table 6-5: Expression Operators

Operator	Description	Type
+	plus	FLOATand INT, unary
–	minus	FLOAT and INT, unary
*	multiply	FLOAT and INT
/	divide	FLOAT and INT
>=	greater than or equal	INT, FLOAT, or STR
<=	less than or equal	INT, FLOAT, or STR
>	greater than	INT, FLOAT, or STR
<	less than	INT, FLOAT, or STR
==	equal	INT, FLOAT, or STR
<>	not equal	INT, FLOAT, or STR
=	assignment	INT, FLOAT, or STR
and	logical AND	INT
or	logical OR	INT
not	logical NOT	INT
band	bitwise AND	INT (assumed to be unsigned)
BOR	bitwise OR	INT (assumed to be unsigned)
BXOR	bitwise EXCLUSIVE OR	INT (assumed to be unsigned)
BNOT	bitwise NOT	INT (assumed to be unsigned)
&	STRING concatenation	STR

Following are some examples of Assignment statements:

`A = A+B;` where A and B are numeric values

`B = 34/VOLTS;` where B and VOLTS are numeric values

`COMMAND = "CMD" & COMMANDTYPE;` where `COMMAND` and `COMMANDTYPE` are string variables

`POLLRESPONSE = SERIALPOLL();` where `POLLRESPONSE` is an integer variable

`LOGICALVALUE = (NOT(A AND B)) OR C;` where `LOGICALVALUE, A, B,` and `C,` are integer variables

`BITVALUE = (BNOT(A AND B)) OR C;` where `BITVALUE, A, B,` and `C` are integer variables

Parentheses can be used to define the order of evaluation of an expression. The default order of precedence for evaluation is as follows (highest precedence to lowest precedence):

not, BNOT	logical negation and bitwise negation
*, /	multiplication and division
+, -	addition and subtraction
&	string concatenation
<=, >=, >, <	less than or equal, greater than or equal, less than, greater than
==, <>	equal and not equal
band	bitwise AND
BOR, BXOR	bitwise OR and bitwise EXCLUSIVE OR
and	logical AND
or	logical OR
=	assignment

IF **Statement** — allows decision processing. The syntax for an IF statement is as follows:

```
IF expression THEN dialogs OR statements;
 ELSE dialogs OR statements;
ENDIF
```

OR

```
IF expression THEN dialogs OR statements;
ENDIF
```

The *expression* in an `IF expression THEN` clause must evaluate to a zero (FALSE) or a non-zero (TRUE) number. By definition, a comparison of STRINGs, less than, greater than, less than or equal, greater than or equal, equal, not equal, bitwise, and logical operators result in a zero or non-zero number. When the *expression* in an IF statement results in a TRUE condition, the *dialogs OR statements* in the initial `IF` statement are executed. When the *expression* results in a FALSE condition, the *dialogs OR statements* in the `ELSE` portion of the construct are executed, or, in the case of no `ELSE`, the *dialogs OR statements* of the `IF` construct are bypassed without execution. Therefore, an `IF` statement would be used to choose between alternative actions, based on the result of the *expression.*

You can nest IF statements.

WHILE Statement — allows repeated dialog evaluation. The syntax for a WHILE statement is as follows:

```
WHILE expression DO dialogs OR statements;
END
```

The *expression* in a `WHILE expression DO` clause must evaluate to a zero (FALSE) or a non-zero (TRUE) number. By definition, a comparison of strings, less than, greater than, less than or equal, greater than or equal, equal, not equal, bitwise, and logical operators result in a zero or non-zero number. The *dialogs OR statements* in the WHILE statement will continue to execute until the *expression* results in a FALSE condition.

You can nest WHILE statements.

Functions

CHR

The number to ASCII character conversion function is used in a controller dialog to send binary character strings to an instrument. The syntax for the function is `CHR(n)`, where n is a number from 0 – 255. You can use this function to send binary or format characters to an instrument. For example, you may have an instrument that needs a binary pattern of 00000010 to set it to a particular state. You could use the following SETTING block to accomplish this:

```
CONTROLGROUP
 CONTROL CH1RESET:INT PUSHBUTTON @24,12
  STRING "CH1 RESET";
 END
 SETTING
  CONT -> CHR(2),CHR(2);
 END
END
```

TIMEDELAY

Another function that is useful in a SETTING or MEASUREMENT block is the **TIMEDELAY** function. It can be used anytime you need to wait for a process to complete, such as closing a relay. The syntax for the TIMEDELAY function is `TIMEDELAY (n);`, where n is in milliseconds. The following is an example of the TIMEDELAY function.

```
SETTING
 CONT -> "CLOSE:F1.A1";
 TIMEDELAY(100);
END
```

Display

A third useful function is the display of a message to the operator. The syntax for this function is **DISPLAY**(*stringdata*), where stringdata is a string constant, string variable, or string expression. This function lets you display a message to the operator on the screen in a dialog box during script execution. It can be used to say that a value is out of range, or that an instrument function requested did not execute. For example, if the voltage output range specified is out of bounds, you could send a message to the operator using the following example script:

```
CONTROLGROUP
 CONTROL VOLTS:FLOAT EDITBOX @20,10
 END
 SETTING
  IF (VOLTS > 24 OR VOLTS < 0.1) THEN
   DISPLAY("THE VOLTAGE MUST BE SET BETWEEN 24 AND 0.1
VOLTS");
  ELSE
   CONT -> "VOLTS: ",VOLTS;
  ENDIF
 END
 MEASUREMENT
 CONT -> "VOLTS?";
 INST -> "VOLTS: ",VOLTS;
 END
END
```

FASTDCWRITE

If you have instruments that support the VXI Fast Data Channel protocol, you can use the FASTDCWRITE function in a SETTING block to send the contents of a file to an instrument. The syntax for the FASTDCWRITE function is as follows:

```
FASTDCWRITE(FILENAME,FDCBASEADDR,FDCSIZE,CMDSTR)
```

Where: FILENAME is the name of the file to read from (a string); FDCBASEADDR is the base address of the instrument FDC memory (an integer); FDCSIZE is the size of the FDC memory (an integer); and CMDSTR is the Word Serial command to send to the instrument to start an FDC transfer (a string).

The following is a simple Script example using the fast data channel functions:

```
SCRIPT "FDCTEST"

 VXIINTERNAL
 END

 VIEW
  CONTROLGROUP
```

```
  CONTROL FDCREAD:STR EDITBOX @5,2
  CONTROLTITLE "FDCREAD:";
  NUMCOLS 14;
 END

 SETTING

 REM  QUERY THE INSTRUMENT FOR THE FDC PROTOCOL
 REM  PARAMETERS.
  TEMPWAR BASEADDR:INT,SIZE:INT;
  CONT -> "HEADER OFF";
  CONT -> "FDCBASE?";
  INST -> "#H",BASEADDR:%H;
  CONT -> "FDCSIZE?";
  INST -> SIZE;
  CONT -> "HEADER ON";

  REM  READ FROM THE INSTRUMENT AND PLACE THE DATA INTO
  REM  A FILE.
  FASTDCREAD(FDCREAD,BASEADDR,SIZE,"FDCLOAD?");
  END
 END

CONTROLGROUP
 CONTROL FDCWRITE:STR EDITBOX @5,6
  STRING "DEF.ED0" ;
  CONTROLTITLE "FDCWRITE:";
  NUMCOLS 14;
 END

 SETTING
 REM  QUERY THE INSTRUMENT FOR THE FDC PROTOCOL
 REM  PARAMETERS.
  TEMPVAR BASEADDR:INT,SIZE:INT;
  CONT -> "NEW";
  CONT -> "FDCBASE?";
  INST -> "#H",BASEADDR:%H;
  CONT -> "FDCSIZE?";
  INST -> SIZE;
  CONT -> "HEADER ON";
  REM  WRITE THE DATA FROM THE FILE TO THE INSTRUMENT.
  FASTDCWRITE(FDCWRITE,BASEADDR,SIZE,"FDCLOAD");
 END
 END
END
```

Notice that FASTDCWRITE uses a SETTING block to transfer data. You need the name of the file to read data from, and the only way to get data from the front panel or test program to the instrument is by using a SETTING block.

FASTDCREAD

If you have instruments that support the VXI Fast Data Channel protocol, you can use the **FASTDCREAD** function in a SETTING block to read data from an instrument and place it into a file. The syntax for the FASTDCREAD function is as follows:

```
FASTDCREAD(FILENAME,FDCBASEADDR,FDCSIZE,CMDSTR)
```

Where FILENAME is the name of the file to read from (a string); FDCBA-SEADDR is the base address of the instrument FDC memory (an integer); FDCSIZE is the size of the FDC memory (an integer); and CMDSTR is the Word Serial command to send to the instrument to start an FDC transfer (a string).

The following is a **Script** example using the fast data channel functions:

```
SCRIPT "FDCTEST"

 VXIINTERNAL
 END

 VIEW
  CONTROLGROUP
   CONTROL FDCREAD:STR EDITBOX @5,2
   CONTROLTITLE "FDCREAD:";
   NUMCOLS 14;
  END

SETTING

  REM  QUERY THE INSTRUMENT FOR THE FDC PROTOCOL
  REM  PARAMETERS.

   TEMPVAR BASEADDR:INT,SIZE:INT;
   CONT -> "HEADER OFF";
   CONT -> "FDCBASE?";
   INST -> "#H",BASEADDR:%H;
   CONT -> "FDCSIZE?";
   INST -> SIZE;
   CONT -> "HEADER ON";

   REM  READ FROM THE INSTRUMENT AND PLACE THE DATA INTO
   REM  A FILE.

   FASTDCREAD(FDCREAD,BASEADDR,SIZE,"FDCLOAD?");
   END
  END

 CONTROLGROUP
  CONTROL FDCWRITE:STR EDITBOX @5,6
   STRING "DEF.ED0" ;
   CONTROLTITLE "FDCWRITE:";
```

```
    NUMCOLS 14;
  END

  SETTING

  REM  QUERY THE INSTRUMENT FOR THE FDC PROTOCOL
  REM  PARAMETERS.

   TEMPVAR BASEADDR:INT,SIZE:INT;
   CONT -> "NEW";
   CONT -> "FDCBASE?";
   INST -> "#H",BASEADDR:%H;
   CONT -> "FDCSIZE?";
   INST -> SIZE;
   CONT -> "HEADER ON";

   REM  WRITE THE DATA FROM THE FILE TO THE INSTRUMENT.

   FASTDCWRITE(FDCWRITE,BASEADDR,SIZE,"FDCLOAD");
  END
 END
END
```

Notice that FASTDCREAD is not in a QUERY bock. You need the name of the file to write data to, and the only way to get data from the instrument to the front panel or test program by using a SETTING block.

READLENGTH

The **READLENGTH** function sets the number of data bytes to read with the next INST dialog. The default value for the number of data bytes read is –1, or all bytes sent by an instrument. An instrument that talks without terminating the data string causes problems with unwanted timeouts. An instrument that does not stop sending data also causes problems. The READLENGTH function applies only to the next INST dialog. The INST dialog resets the READLENGTH value to –1 (read all data) when it is complete. The READLENGTH function does not set the absolute number of data bytes to read. It only put an upper limit on the number of bytes read. Fewer data bytes may be read depending on the data being sent from the instrument. The following is an example of the READLENGTH function.

```
SETTING
  CONT -> "*IDN?";
  READLENGTH(12);
  INST -> VALUE1;
END
```

If an instrument has more data ready to send than will be read by the INST dialog as modified by the READLENGTH function, the status of the extra data strings will be determined by the instrument. In some cases the data is discarded and in some cases, th e data will be sent to the next INST dialog.

Variable Types

There are four variable types available in scripts: integers, floats, STRING and waveform. The variable name is followed by the variable type (`:INT` for integer, `:FLOAT` for floats, `:STR` for String and `:WAVE` for waveform). Variable names must start with an alphabetic character and be no more than 15 characters in length.

Integers

Integers require 4 bytes for storage. The range is -2^{31} — 2^{31}-1.

Floats

Floats require 8 bytes for storage and use the IEEE format. The approximate range (as specified for Microsoft C) is 2.2E-308 — 1.8E+308.

Strings

A string is any number of bytes, each byte being a character. The number of bytes in a string is limited to the amount of memory available.

Literal strings (enclosed in " ") are limited to 255 characters. If a literal string exceeds this limit, an error is reported and parse fails. Strings longer than 255 characters can be built up in script using the ampersand (`&`) concatenation operator. Several special printing and non-printing characters can be included in a literal string by using the backslash character (`\`) followed by the desired character. These special characters and their string entries are as follows:

"	`\"`
LF (Line Feed	`\R`
HT (Horiz Tab)	`\T`
CRLF (Return Line Feed)	`\n`

Carriage returns are ignored in literal strings except when explicit `\n` appears.

Waveforms

Waveform variables are a special type of data and can be used only in very specific situations. You can either assign one waveform variable to another waveform variable or assign it to the function WAVEFORMTOVAR. Any other use of the waveform variable causes an error.

Variable Formats

In controller and instrument dialogs, you can precisely specify the format for variables.

Controller Dialog Formats

In a controller dialog, the format can be specified for a string variable.

To specify the format, each variable is followed by a colon, followed by the format. The format specification begins with a % and ends with a format character. Between the % and the format character one may specify the modifiers, width, and precision.

The following are examples of specifying formats:

```
CONT -> "VOLTS ",VOLTS:%9.5E;
CONT -> "AMPS ",AMPS:%7.3G;
```

The following list shows the format types that are available, what they are used for, and the default field widths:

%nD — integer, the numbers 0 – 9, + and –. n specifies the number of digits. Default width of 10 if n is not specified.

%*n.m*G — floating point, integers, and numbers with decimal points. n defines the total field size, including the decimal point, and m defines the maximum number of characters following the decimal point. Numbers of greater precision than specified will be rounded. Default width of 19 if n is not specified.

%*n.m*E — integers, floating point, and exponential numbers. Scientific notation characters are 0 – 9, + and –, E or e. n defines the total field size, including the decimal point, m defines the maximum number of characters following the decimal point. Numbers of greater precision than specified will be rounded. Default width of 24 if n is not specified.

%*n*B — binary, the numbers 0 and 1. The number is assumed to be unsigned and is interpreted as right justified. n specifies the number of digits. Default width of 32 if n is not specified.

%*n*O — octal, the numbers 0 – 7. n specifies the number of digits. Default width of 11 if n is not specified.

%*n*H — hexadecimal, the numbers 0 – 9 and the letters A – F. Input is assumed to be unsigned. n specifies the number of digits. Default width of 8 if n is not specified.

%nS — string. n specifies the number of characters. All characters are accepted if n is not specified. No default width.

%F — file name. No default width.

If the specified format is incompatible with the type of the variable, an error is reported.

Binary, Octal, and Hexadecimal formats are used to write ASCII string representations of these types of numbers; they do not write binary representations of these formats.

If n is not specified, the processing is Free Format. Output is based on the minimum number of bytes required for that format.

If the format is not specified, the type of the variable is taken as the default format and the format is Free Format.

You can use appropriate width or width precision modifiers with these format types. The following are examples of specifying field widths and using precision modifiers with format types:

```
CONT -> VAR1:%6D;
CONT -> VAR1:%12.6E;
```

If the format width is specified but is not large enough to accommodate the number, an error is reported.

If the width is specified and the number of digits for the number is less than the width, the remaining characters are discarded.

The following is an example of specifying field widths, using a precision modifier, and using a formatting modifier:

```
CONT -> "VOLTS ",VOLTS:%+5.2G;
```

In this example, + is the formatting modifier, 5 is the specified width, 2 is the precision modifier, and G specifies the format type.

The available formatting modifiers are:

Z	Leading zero fill instead of the default space fill
+	Always output sign.
<	Left justify instead of the default right justify.

Following are examples in the use of formatting modifiers (assume the variable VOLTS contains the value 1.1):

```
CONT -> "VOLTS ",VOLTS:%<5.2G; output STRING "VOLTS 1.10 "
                                (contains a trailing space)
CONT -> "VOLTS ",VOLTS:%+5.2G; output STRING "VOLTS +1.10"
CONT -> "VOLTS ",VOLTS:%Z5.2G; output STRING "VOLTS 01.10"
CONT -> "VOLTS ",VOLTS:%+Z5.2G; output STRING "VOLTS
+01.10"
```

Notice that you can combine formatting modifiers.

The file name format is used with a string variable and indicates that the entire transaction is between a file and an instrument. Using %F in a controller dialog will transfer the contents of a file to an instrument. If you use the %F format, it must be the only format specified in the dialog, or an error will be reported.

Following is an example of using the `%F` *format:*

```
SETTING
 DATAFILE = "TEST1.DAT";
 CONT -> DATAFILE:%F;
END
```

Table 6-6 lists the available formats for controller dialogs:

Table ***6-6:*** *Controller Dialog Format*

CONT	**B**	*D*	**E**	*F*	*G*	**H**	*O*	**S**
STRing				X				X
INTeger	X	X	X		X	X	X	
FLOAT		X	X		X			
Z	X	X	X		X	X	X	
+		X	X		X			
<	X	X	X		X	X	X	X
n	X	X	X		X	X	X	X
m			X		X			
Default Width	*32*	*10*	*24*		19	**8**	11	

NOTE

IPG converts all controller dialogs, except those using the `%F` *format, to strings before sending them to the instrument. Therefore, it is very important to properly specify modifiers if and when they are used. For example, if the value* `10.1` *was to be sent to the instrument, and the modifier 2.1G was specified, only* `10` *would be sent. This occurs because IPG truncates data based on the modifier format specified. When truncation occurs, a warning message is generated by IPG. In this case, the proper modifier should have been* `4.1G`*. This problem can be prevented by using the* `%G` *modifier format rather than the* `N.MG` *modifier format.*

Instrument Dialog Formats

In an instrument dialog, the format can be specified for a variable.

To specify the format, each variable is followed by a colon, followed by the format. The format specification begins with a % and ends with a format character. Between the % and the format character one may specify the modifiers, width, and precision.

The following are examples of specifying formats:

```
INST -> STATUS:%6B;
INST -> COUNT:%4D;
```

The following list shows the format types that are available, what they are used for, and the default field widths:

%nD	integer, the numbers 0 – 9, + and –. n specifies the number of digits. Default width of 10 if n is not specified.
%*n.m*G	floating point, integers, and numbers with decimal points. n defines the total field size, including the decimal point, and m defines the maximum number of characters following the decimal point. Default width of 19 if n is not specified
%*n.m*E	integers, floating point, and exponential numbers. Scientific notation characters are 0 – 9, + and –, E ore. n defines the total field size, including the decimal point, m defines the maximum number of characters following the decimal point. Default width of 24 if n is not specified.
%nB	binary, the numbers 0 and 1. The number is assumed to be unsigned and is interpreted as right justified. n specifies the number of digits. Default width of 32 if n is not specified.
%nO	octal, the numbers 0 – 7. n specifies the number of digits. Default width of 11 if n is not specified.
%nH	hexadecimal, the numbers 0 – 9 and the letters A – F. Input is assumed to be unsigned. n specifies the number of digits. Default width of 8 if n is not specified.
%nS	string. n specifies the number of characters. All characters are accepted if n is not specified — no default width.
%F	file name (no default width)

If the specified format is incompatible with the type of the variable, an error is reported.

Binary, Octal, and Hexadecimal formats are used to read ASCII string representations of these types of numbers; they do not read binary representations of these formats.

If n is not specified, the processing is Free Format. Input is accepted until either a maximum number of bytes have been received, or a non-valid character is received for that field.

If the format is not specified, the type of the variable is taken as the default format and the format is Free Format.

You can use appropriate width or width and precision modifiers with these format types. The following are examples of specifying field widths and using precision modifiers with format types:

```
CONT -> VAR1:%6D;
CONT -> VAR1:%12.6E;
```

If the format width is specified but is not large enough to accommodate the number, an error is reported.

If the width is specified and the number of digits for the number is less than the width, the remaining characters are discarded.

The file name format is used with a string variable and indicates that the entire transaction is between a file and an instrument. Using `%F` in an Instrument dialog will place all of the data read from the instrument into a data file. If you use the `%F` format, it must be the only format specified in the dialog, or an error will be reported.

Following is an example of using the `%F` format:

```
QUERY
 DATAFILE = "RESPONSE.DAT";
 INST -> DATAFILE:%F;
END
```

Table 6-7 lists the available formats for Instrument dialogs:

Table 6-7: Instrument Dialog Format

INST	B	D	E	F	G	H	O	S
STRing				X				X
INTeger	X	X	X		X	X	X	
FLOAT		X	X		X			
n	X	X	X		X	X	X	X
m			X		X			
Default Width	32	10	24		19	8	11	

Common Problems when Using Instrument Dialog Formatting

Not Enough Data Received — Parsing of an instrument dialog takes place from left to right. If the string received from the instrument does not contain enough bytes to completely parse the Instrument dialog, an ERROR is reported. Following is an example:

```
MEASUREMENT
  TMPVAR VAR1:STR;
  INST >> VAR1:%5S;
END
```

Assuming that the string returned from the instrument is `ON,` an error is reported because the string returned from the instrument contains only two characters and the Instrument dialog requires that five bytes be available.

Too Much Data Received — If the string returned from the instrument contains more bytes than required to parse an instrument dialog, then the excess bytes are ignored (the rest of the buffer is flushed). Following is an example:

```
MEASUREMENT
  TMPVAR VAR1:INT;
  INST >> VAR1;
END
```

Assuming that the string returned from the instrument is `12 VOLTS,` `VAR1` is assigned the value `12,` and the rest of the string, `VOLTS,` is ignored.

Non-Numeric Character in a Numeric Format Handling — If the width (for example, n) is specified in the format, then a non-valid character, if received before n characters have been received, delimits the input. A non-valid character is a character that is not compatible with the format; e.g., non-digit for `%D` format, or non-binary for `%B` format. If the first character is a non-valid character, numeric conversion is stopped at the invalid character and no characters are processed.

ISL Keywords

Table 6-8 provides an alphabetical list of the reserved keywords in the Instrument Script Language. You can not use these keywords for any other purpose, such as naming a variable; they are reserved for ISL use.

Table 6-8: ISL Keywords Listed Alphabetically

&	DEFAULTPOS	NORMALCLOSE	TEXT
*	DISPLAY	NORMALOPEN	TEXTBOX
+	DO	NOT	THEN
	EDIT	NUMCOLS	TIM
/	EDITBOX	NUMPOS	TIMEDELAY
:%	ELSE	NUMROWS	TIMEOUT
<	END	OFF	TITLE
<=	ENDIF	ON	TMO
=	EOI	ONEOFGROUP	TMPVAR
==	EOM	OPEN	TO
>	FASTDCREAD	OR	TOLEFTOF
=>	FASTDCWRITE	PARITY	TORIGHTOF
AND	FLOWCONTROL	PUSHBUTTON	TOSCRIPT
ASCII	FALSE	QUERY	TOSCRIPTOFF
ATN	FILE	RADIOBUTTON	TOSCRIPTON
BAND	FLOAT	READLENGTH	TRUE
BAUD	GET	REM	UNL
BINARY	GPIB	REMARK	UNT
BNOT	GTL	REN	UPDATE
BOR	HELP	RS232	UPDATELIST
BXOR	HSCROLL	SCANSTRING	VIEW
CDSBUS	IF	SCRIPT	VSCROLL
CHECKBOX	IFC	SCROLLHORZ	VXIEVENT
CHR	INST	SCROLLVERT	VX5520
CLOSE	INSTRUMENT	SDC	VXIINT
CONNECT	INT	SERIALPOLL	VXIINTERNAL
CONT	LEARN	SET	WAVE
CONTROLLER	LISTBOX	SETTING	WAVEFORM-DISPLAY
CONTROL	LLO	SKIP	WAVEFORM-TOADIF
CONTROL-GROUP	MEASURE	STATE	WAVEFORMTO-VAR
CONTROLTITLE	MEASURE-MENT	STOPBITS	WHILE
DATABITS	MIDSTRING	STR	YES
DCL	MXI	STRING	
DEFAULT	NO	TEMPVAR	

Table 6-9 lists the ISL keywords by their use.

Table **6-9:** ISL Keywords Listed **by** Group

BUSNOTE	BAUD	TROL	TIMEOUT
	CDSBUS	GPIB	TMO
	DATABITS	MXI	VX5520
	EOI	PARITY	VXIEVENT
	EOM	RS232	VXIINT
	FLOWCON-	STOPBITS	VXIINTERNAL
CONTROL Block	CHECKBOX	NORMALCLOSE	STRING
	CLOSE	NORMALOPEN	TEXTBOX
	CONTROL	NUMCOLS	TITLE
	CONTROLTITLE	NUMPOS	TOLEFTOF
	DEFAULT	NUMROWS	TORIGHTOF
	DEFAULTPOS	OFF	TOSCRIPT
	EDIT	ON	TOSCRIPTOFF
	EDITBOX	ONEOFGROUP	TOSCRIPTON
	FALSE	OPEN	TRUE
	FILE	PUSHBUTTON	UPDATE
	FLOAT	RADIOBUTTON	UPDATELIST
	HSCROLL	SCANSTRING	VSCROLL
	INT	SCROLLHORZ	WAVE
	LISTBOX	SCROLLVERT	WAVEFORM-
	MIDSTRING	STATE	DISPLAY
	NO	STR	YES
CONTROL- GROUP Block	CONTROL- GROUP		
Format	:%		
Functions	CHR	SERIALPOLL	WAVEFORM- TOADIF
	FASTDCREAD	SKIP	
	FASTDCWRITE	TIMEDELAY	WAVEFORMTO- VAR
	READLENGTH		
GPIB Commands	ATN	IFC	TIM
	DCL	LLO	UNL
	GET	REN	
	GTL	SDC	UNT
LEARN Block	ASCII	BINARY	LEARN
Miscellaneous	HELP	REM	REMARK

Table 6-9: ISL Keywords Listed by Group (Cont.)

Operators	&	<=	BAND
	*	=	BNOT
	+	==	BOR
	−	>	BXOR
	/	=>	NOT
	<	AND	OR
Script Block	SCRIPT		
SETTINGs and **MEASUREMENT** Blocks	CONT	IF	SET
	CONTROLLER	INST	SETTING
	DISPLAY	INSTRUMENT	TEMPVAR
	DO	MEASURE	THEN
	ELSE	MEASURE-	TMPVAR
	END	MENT	TO
	ENDIF	QUERY	WHILE
VIEW Block	CONNECT	TEXT	VIEW

Appendix A: Glossary

Bus

The communications channel used to control programmable instrumentation (VXIbus, GPIB, RS232, MXI, CDS53).

Control Selection

Control selection indicates the control that has the focus for any edit functions. FPE requires a control to be selected before an edit action may be performed. Control selection is indicated by eight drag handles surrounding the control.

Controlgroup

A Controlgroup is a group of instrument controls. ControlGroup is an ISD specific term that refers to a type of ISD block. A ControlGroup groups instrument controls together that have a common setting and measurement scene block.

Drag Handle

Drag handles are the eight inverted rectangles that appear around a selected control. The handles indicate the mouse cursor hot point for changing the size of the control.

Focus

A concept of identifying where the current action will occur. The focus area is typically identified by a dotted rectangular box of a blinking cursor.

Group Select/Selection

A group selection is a selected group of controls within a control panel view. A group selection is performed using the mouse to click and drag a selection rectangle around the desired controls. A group selection is indicated by a dotted rectangle around the group of controls.

IFP

Instrument Front Panel. An Instrument Front Panel is a graphical representation of an instrument panel. Although the name indicates front panel, you may use it to define the rear panel also.

ISD

Instrument Software Definition files. Files that contain Instrument Script Language.

ISL

Instrument Script Language. The language that describes instrument control panels.

Measurement Block

The IPG can only receive data from the Measurement Block.

Scene Block

Scene block contains the code to define the instrument controls behavior. There are three scene blocks, settings, measurement, and learn scenes.

Settings Block

The IPG can only send data to the Settings Block

Script

Scripts are instrument driver files that define instrument control panels.

Tool Mode

Tool mode is the FPE's control creation mode. Editor must be in tool mode to create new controls. Tool mode is indicated by the crosshair cursor.

Point Mode

Point mode is the FPE's selection mode. With FPE in point mode you can move and select controls. The point mode is indicated by the arrow cursor.

Panel

A panel is a window or view that contains graphical representations of instrument controls. Panel comes from the term instrument front panel.

Appendix B: TDM5120.ISD Script Printout

A printout of TDM5120.ISD is provided here so you can examine the entire script while developing new scripts or modifying scripts.

```
REM Tektronix DM5120 Digital Multimeter Script "DM5120"

   GPIB
   End

  Learn var:STR
    Setting
      cont -> var;
    End
    Measurement
      cont -> "SET?";
      inst -> var;
    End
  End

  View "ACV"

    Text "DM5120 AC VOLTMETER" @ 5,1.0;
    Text "ACCURACY" @ 3.75,6.5;
    Connect (12.5,7),(14,7),(14,14),(2,14),(2,7),(3,7);

    ControlGroup
      Control acvacq:INT PushButton @ 18.75,15

      REMARK ...acq for acquire

        String "Acq";
        UpDateList acvreading;
      End
      Setting
        cont -> "ACV";
      End
    End

    ControlGroup
      Control acvreading:FLOAT TextBox @ 2.5,4.5
        NumRows 1; NumCols 20;
        ControlTitle "Measurement";
      End
      Measurement
        cont -> "SEND";
        inst -> acvreading,";";
      End
```

```
End

ControlGroup
  Control acvthree:INT RadioButton @ 2.5,7.5
    String "3 Digits";
    OneOfGroup "accuracy";
    UpDateList acvreading;
  End
  Set
    if (acvthree==1) then
      cont -> "DIGIT 3";
    endif
  End
  Query
    tempvar tmp:STR;
    acvthree = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="3") then
      acvthree = 1;
    endif
  End
End

ControlGroup
  Control acvfour:INT RadioButton @ 2.5,9
    String "4 Digits";
    OneOfGroup "accuracy";
    UpDateList acvreading;
  End
  Setting
    if (acvfour==1) then
      cont -> "DIGIT 4";
    endif
  End
  Measurement
    tempvar tmp:STR;
    acvfour = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="4") then
      acvfour = 1;
    endif
  End
End
ControlGroup
  Control acvfive:INT RadioButton @ 2.5,10.5
    String "5 Digits";
    OneOfGroup "accuracy";
    UpDateList acvreading;
  End
```

```
  Setting
    if (acvfive==1) then
      cont -> "DIGIT 5";
    endif
  End
  Measurement
    tempvar tmp:STR;
    acvfive = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="5") then
      acvfive = 1;
    endif
  End
End

ControlGroup
  Control acvsix:INT RadioButton @ 2.5,12.
    String "6 Digits";
    OneOfGroup "accuracy";
    UpDateList acvreading;
  End
  Setting
    if (acvsix==1) then
      cont -> "DIGIT 6";
    endif
  End
  Measurement
    tempvar tmp:STR;
    acvsix = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="6") then
      acvsix = 1;
    endif
  End
End

ControlGroup
  Control acvfilt:INT CheckBox @ 2.5,15.0
    String "Ave";
    UpDateList acvreading;
  End
  Control acvfiltval:INT Edit @ 8.75,15.0
    NumRows 1;NumCols 5;
  End
  Setting
    if (acvfilt==1) then
      cont -> "FILTERVAL ",acvfiltval,";FILTER ON";
    else
      cont -> "FILTER OFF";
```

```
      endif
    End

    Measurement
      tempvar filton:STR;
      cont -> "FILTER?;FILTERVAL?";
      inst -> " ",filton,";"," ",acvfiltval,";";
      if (filton=="ON") then
         acvfilt=1;
      else
         acvfilt=0;
      endif
    End
  End

  ControlGroup
    Control acvrange:STR ListBox @ 18.75,8.0
      NumRows 5;NumCols 5;
      ControlTitle "RANGE";
      String " AUTO","300 mV",
                  "  3 V"," 30 V","300 V";
      ToScript "AUTO","1",
                    "2","3","4";
      UpDateList acvreading;
    End
    Setting
      cont -> "RANGE ",acvrange;
    End
    Measurement
      cont -> "RANGE?";
      inst -> " ",acvrange,";";
    End
  End
End

View "DCV"

  Text "DM5120 DC VOLTMETER" @ 5,1.0;
  Text "ACCURACY" @ 3.75,6.5;
  Connect (12.5,7),(14,7),(14,14),(2,14),(2,7),(3,7);

  ControlGroup
    Control dcvacquire:FLOAT PushButton @ 18.75,15
      String "Acq";
      UpDateList dcvreading;
    End
    Setting
      cont -> "DCV";
    End
  End
```

```
ControlGroup
  Control dcvreading:FLOAT TextBox @ 2.5,4.5
    NumRows 1; NumCols 20;
    ControlTitle "Measurement";
  End
  Measurement
    cont -> "SEND";
    inst -> dcvreading,";";
  End
End

ControlGroup
  Control dcvthree:INT RadioButton @ 2.5,7.5
    String "3 Digits";
    OneOfGroup "accuracy";
    UpDateList dcvreading;
  End
  Set
    if (dcvthree) then
      cont -> "DIGIT 3";
    endif
  End
  Query
    tempvar tmp:STR;
    dcvthree = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="3") then
      dcvthree = 1;
    endif
  End
End

ControlGroup
  Control dcvfour:INT RadioButton @ 2.5,9
    String "4 Digits";
    OneOfGroup "accuracy";
    UpDateList dcvreading;
  End
  Setting
    if (dcvfour) then
      cont -> "DIGIT 4";
    endif
  End

  Measurement
    tempvar tmp:STR;
    dcvfour = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="4") then
```

```
      dcvfour = 1;
    endif
  End
End

ControlGroup
  Control dcvfive:INT RadioButton @ 2.5,10.5
    String "5 Digits";
    OneOfGroup "accuracy";
    UpDateList dcvreading;
  End
  Setting
    if (dcvfive) then
      cont -> "DIGIT 5";
    endif
  End
  Measurement
    tempvar tmp:STR;
    dcvfive = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="5") then
      dcvfive = 1;
    endif
  End
End

ControlGroup
  Control dcvsix:INT RadioButton @ 2.5,12.
    String "6 Digits";
    OneOfGroup "accuracy";
    UpDateList dcvreading;
  End
  Setting
    if (dcvsix) then
      cont -> "DIGIT 6";
    endif
  End
  Measurement
    tempvar tmp:STR;
    dcvsix = 0;
    cont -> "DIGIT?";
    inst -> "DIGIT ", tmp, ";";
    if (tmp=="6") then
      dcvsix = 1;
    endif
  End
End
```

```
ControlGroup
      Control dcvfilt:INT CheckBox @ 2.5,15.0
        String "Ave";
        UpDateList dcvreading;
      End
      Control dcvfiltval:INT Edit @ 8.75,15.0
        NumRows 1;NumCols 5;
      End

      Setting
        if (dcvfilt==1) then
          cont -> "FILTERVAL ",dcvfiltval,";FILTER ON";
        else
          cont -> "FILTER OFF";
        endif
      End
      Measurement
        tempvar filton:STR;
        cont -> "FILTER?;FILTERVAL?";
        inst -> " ",filton,";"," ",dcvfiltval,";";
        if (filton=="ON") then
           dcvfilt=1;
        else
           dcvfilt=0;
        endif
      End
    End

ControlGroup
      Control dcvrange:STR ListBox @ 18.75,8.0
        NumRows 5;NumCols 5;
        ControlTitle "RANGE";
        String " AUTO","300 mV",
                    "  3 V"," 30 V","300 V";
        ToScript "AUTO","1",
                      "2","3","4";
        UpDateList dcvreading;
      End
      Setting
        cont -> "RANGE ",dcvrange;
      End
      Measurement
        cont -> "RANGE?";
        inst -> " ",dcvrange,";";
      End
    End
  End
End
```

Appendix B: TDM5120.ISD Script Printout

Appendix C: Software Performance Report

This Software Performance Report is for your use in reporting any problems you experience when using TekTMS/IPG. It provides us with a way to track problems with a particular system and it ensures that we provide you and other customers with a prompt solution to the problem. Please supply the following information:

Customer (Company Name): _____

User (Person making report): _____

Address: _____

City: _____ State: _____ Zip: _____

Country: _____ Telephone: _____

System Description (Product Name, S/W Version, Serial Number): _____

Product Name: _____ PC Model/Make: _____

Version: _____ DOS Version: _____

Serial No: _____ Windows Version: _____

Other Software (TSR's, ...)

Problem Description: _____

_____

_____

_____

_____

(If possible, include exact steps to recreate the problem.)

Itemize attached documentation (i.e., listings, diskettes, ...)

Mail the report to:

Tektronix, Inc.
Instrument Controllers and Software Marketing Dept.
MS 47-665
P.O. Box 500
Beaverton, OR 97075-9965

Or FAX to:

(503) 627-2933

Marked for ICS Marketing, 47-665

Appendix C: Software Performance Report

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

U

V

W

Z