

Series 2040 Test Systems

CYCLOPS

User Manual

Part Number 4200-0171
Version 3.1

Table Of Contents

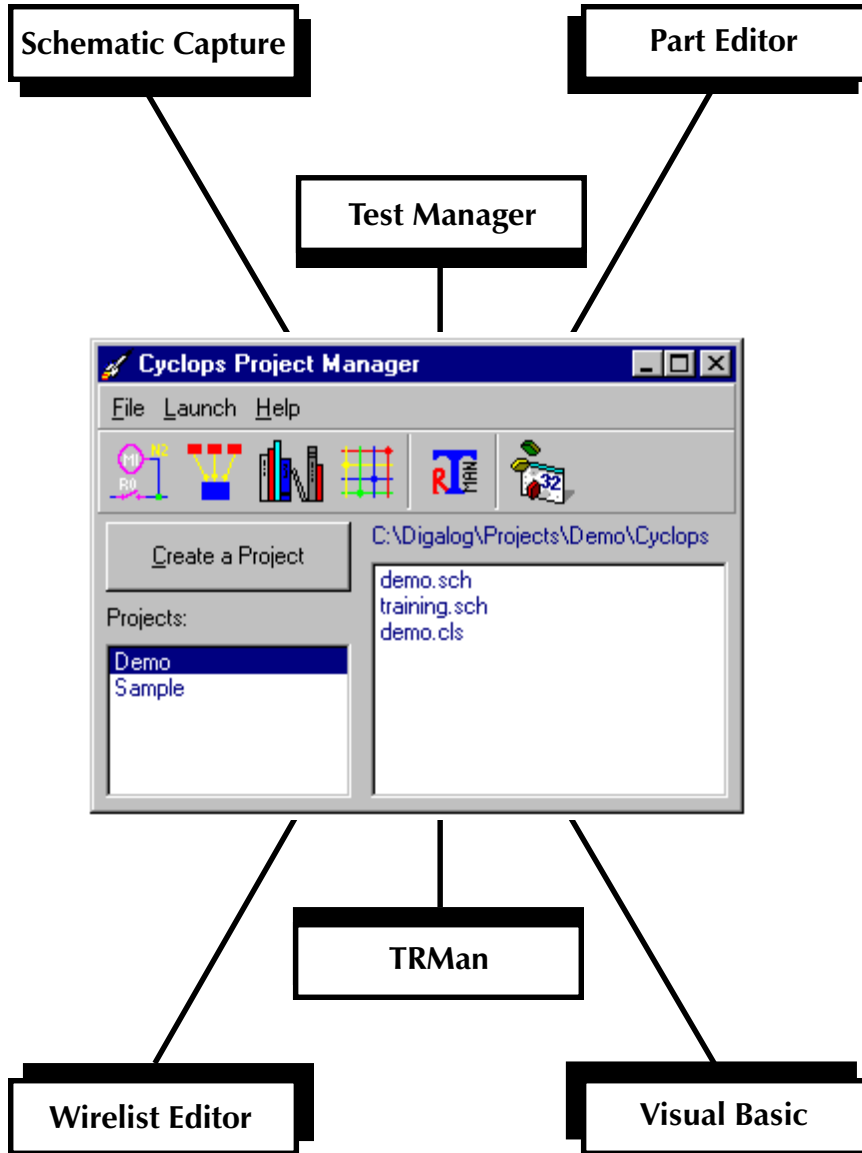
OVERVIEW	5
OVERVIEW	7
Digalog Environment Variable	8
Digalog Directory Structure	8
Cyclops Filename Extensions	9
CYCLOPS PROJECT MANAGER	12
Menu Bar	13
Schematic Capture	15
Status Bar	20
Digalog Resources Toolbar	20
Parts	20
TEST MANAGER	23
TEST MANAGER TAB	26
BASIC CODE WINDOW	29
CYX EXECUTIVE API	32
Changeable Code	32
CYX Executive Form	32
Provided Unchangeable Code	33
cyxdata.cls	33
cycldata.cls	34
cyx.bas	34
Interface to CYX executive	34
CYX executive API initialization	35
CYX executive API execution	35
CYX executive API data logging	35
fixture.bas	37
niglobal.bas, vbib32.bas	37
analog32.bas	37
dlimisc32.bas	37
Directory Structure	38
TOOL DIALOGS	41
COMMON INTERFACE	41
Example Code Window	42
Adjustable Digital I/O	43
Amplitude Measurement System	44
Arbitrary Waveform Generators	46

ARB	48
ARBFreq	48
ARBSin	49
ARBProg	49
ARBPulse	49
Auxiliary Relays.....	50
Idle	50
D/A Converters	51
Digital I/O.....	51
Drivers	52
Receivers	53
Clocking	54
Isolation Amplifiers	55
Matrix Relays	56
MRly.....	57
MODMRly	57
MRlyReset	57
Measurement Display Electronics	58
TrigA	59
Trig1	59
Open Collector I/O	60
OCEn	60
OCRail	60
OCData	60
OCRead	61
OCPut	61
OCGet.....	61
OCclk	61
OCStrobe	61
Patchboard ID	61
Programmable Power Supplies	62
Relay Multiplexer	63
Selftest Multiplexer	64
TClear	65
Time Measurement System	66
TCount	67
DTime	67
Freq.....	67
Ratio.....	68
Trigger Matrix.....	68

PART EDITOR.....	71
Part Editor Grid	71
Pin Editor Grid	72
Part Size Display	72
Part Display	72
Menu Bar	72
File Menu	72
Edit Menu	73
Help Menu	73
WIRELIST EDITOR.....	75
Wire Editor Grid	75
Net Column	75
Connections Column	75
Wire Size Column	76
Comments	76
Menu Bar	76
File Menu	76
Edit Menu	76
Help Menu	76
TRMAN (Tester Resource Manager)	77
Menu Bar	77
MICROSOFT® VISUAL BASIC®	83
VISUAL BASIC FUNDAMENTALS.....	83
MENU BAR	84
File Menu	84
Edit Menu	84
View Menu	85
Project Menu	86
Format Menu	86
Debug Menu	87
Run Menu	87
Tools Menu	87
Add-Ins Menu	88
Window Menu	88
Help Menu	88
TOOLBAR	89
PROJECT WINDOW.....	92
TOOLBOX.....	94
PROPERTIES WINDOW	94
FORM LAYOUT WINDOW	95

FUNCTIONAL CALLS	95
cyldata.cls	97
Cyx.bas	99
Cyresult.bas	101
Glossary Of Terms.....	103

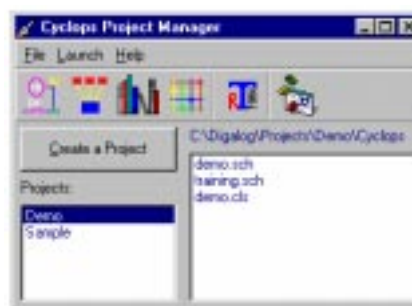
Overview



OVERVIEW

The Cyclops Test Development Environment is used in conjunction with the Digalog 2040 Series Tester and the Microsoft® Visual Basic® Development System to create and maintain test programs for the functional testing of products. It contains the necessary programs for creating fixtures and developing test programs for company products. The current release of Cyclops contains:

- Cyclops Project Manager
- Schematic Capture
- Test Manager
- Part Editor
- Wire Editor
- Tester Resource Manager
- Visual Basic



The Cyclops Project Manager is used for the creation and maintenance of projects. It contains a toolbar of selected programs that can be run from the Project Manager.

The Schematic Capture is used to create schematics for fixture design, and to select tester parts to be controlled. It can be used to display status information during the development and debugging of test programs. Finally, it can also create a wirelist for fixture construction.

The Test Manager organizes and executes sequences of tests as designated by the programmer.

The Part Editor allows the creation of part libraries to be used with the Schematic Editor.

The Wire Editor allows extra information (primarily wire size) to be added to the wirelist. This information may be used in constructing a fixture.

The Tester Resource Manager allows the modification and generation of the 2040 Tester resources available to Cyclops and other Digalog programs.

The Microsoft Visual Basic® Development System is the environment that is used to create and compile the test executives generated by the Test Manager.

Once the project is loaded, the program can be modified to suit the particular needs of the current product or project. Visual Basic must be installed for Cyclops to function properly.

Digalog Environment Variable

The [DIGALOG] environment variable is used by Cyclops to locate where the necessary files and directories are located. It is usually set to:

set DIGALOG=C:\DIGALOG

This means that all of the Digalog specific files are usually located in the directory C:\DIGALOG. Cyclops depends on this variable and the directory structure as defined below.

Digalog Directory Structure

The Digalog directory structure is set up so that the programs that are generated by Cyclops know where the necessary component files are located. This includes: executables, libraries, initialization files, bitmaps, projects, etc. The root of this structure is defined by the [DIGALOG] environment variable. All of the subdirectories defined below are located within the [DIGALOG] directory.

\bin - This directory is where all of the Digalog utilities and programs are located. All of the Cyclops executables are also located here. If programs are used by more than one project, this would be a good place to put them. Programs specific to a particular project should be placed in the project's own subdirectory.

\doc This directory is where all of the Digalog help files and other documentation is located.

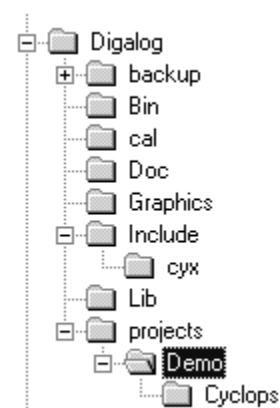
\include - The \include directory is where all of the Digalog specific Visual Basic modules are located. There are subdirectories that contain more specific Visual Basic modules defined below. Any modules or initialization files global to all projects should also be placed in this directory.

\include\cyx - The \include\cyx directory is where the Cyclops executive source resides. If the executive needs to be updated for use by all projects, then this is the place to update. When the Test Sequence Manager creates the executive, there are specific files it copies from this directory into the current project.

\lib - The \lib directory is where the global part libraries are kept. These libraries are used by the Schematic Capture Program. The libraries digalog.plb and part.plb are loaded from this directory.

\projects - The \projects subdirectory contains a subdirectory for each project created by the Cyclops Project Manager. All files relevant for each project are stored under its project subdirectory.

\projects\DEMO\cyclops - The \projects\DEMO\cyclops subdirectory is where all of the files that Cyclops creates are placed for a project named DEMO. The DEMO subdirectory represents a typical project name used. For example, \projects\DEMO\ is the project for the Cyclops DEMO program, and has a subdirectory \projects\DEMO\cyclops where all of the Cyclops specific files are stored for that specific project.



Cyclops Filename Extensions

Cyclops Filename Extensions are important to understand because the programs rely on these extensions to determine the types of files that are located throughout the Digalog directory structure. Cyclops programs will not recognize files with different extensions than the ones described below.

.plb - The .plb file extension is used for the part libraries for the Schematic Capture program. It contains information on drawing the parts and part pins, as well as the attributes of the parts (described later). There are two part libraries read from the \lib subdirectory. Other libraries read may be located in the \cyclops subdirectory of individual projects.

\lib\digalog.plb - The \lib\digalog.plb file contains parts that are specific to 2040 Testers. It also contains the attributes that can be displayed for each part in this file.

\lib\part.plb - The \lib\part.plb file is a general purpose library for parts used inside the Patchboard fixture like resistors, capacitors, etc.

\projects\demo\cyclops*.plb - These libraries are generated from the Part Editor. These are all loaded when the Schematic Capture program is run. They contain parts specific to a project named DEMO.

.sch - The .sch file extension is for schematic files. These files contain information for the connections, parts, labels, and junctions on the schematic. This is the extension used by the Schematic Capture program and cannot be changed.

.tcl - The .tcl file extension is associated with test classes. These files are created by the Test Manager and share the same format as Visual Basic® class modules. They contain all the code and additional information necessary to conduct a set of tests and evaluate the results. These files will be used in the test executive generated by the Test Manager.

.wir -The .wir file extension is used for a wirelist that is generated from the Schematic Capture program. It is a wire list of the schematic with the same name with the **.wir** extension.

.arb - The ARB waveform file extension.

.arp - The ARB arbpulse waveform file extension.

.mde - MDE waveform file extension.

.vbp - The .vbp file extension is for Visual Basic® projects.

.bas - The .bas file extension is for Visual Basic® modules.

.cls - The .cls file extension is for Visual Basic® classes.

.frm - The .frm file extension is for Visual Basic® forms.

.frx - The .frx file extension is for Visual Basic® form “stash” file (binary).

CYCLOPS PROJECT MANAGER

The Cyclops Project Manager is the first program that starts in the Cyclops Test Development Environment. It manages projects and programs. There is a toolbar that contains all of necessary Cyclops programs. One of the two large windows shows a projects list of available Cyclops projects, and the other shows a file list of the currently selected project. Only projects that contain a \cyclops subdirectory are shown. Any project that doesn't contain this directory can easily be added with the "Create Project" button.



Toolbar - The toolbar contains all of the Cyclops applications.

Schematic Capture - The Schematic Capture program is the first program on the toolbar, and is used to create schematics, provide visual feedback while debugging sequences, and generating fixture wirelists.

Test Manager - The Test Sequence Manager is used to create, manage, and debug all code to test an individual project. After the code has been written and debugged, a standard test executive may be generated as a stand-alone program which incorporates this code.

Part Editor - The Part Editor is the third program, and creates the part libraries used by the Schematic Capture program.

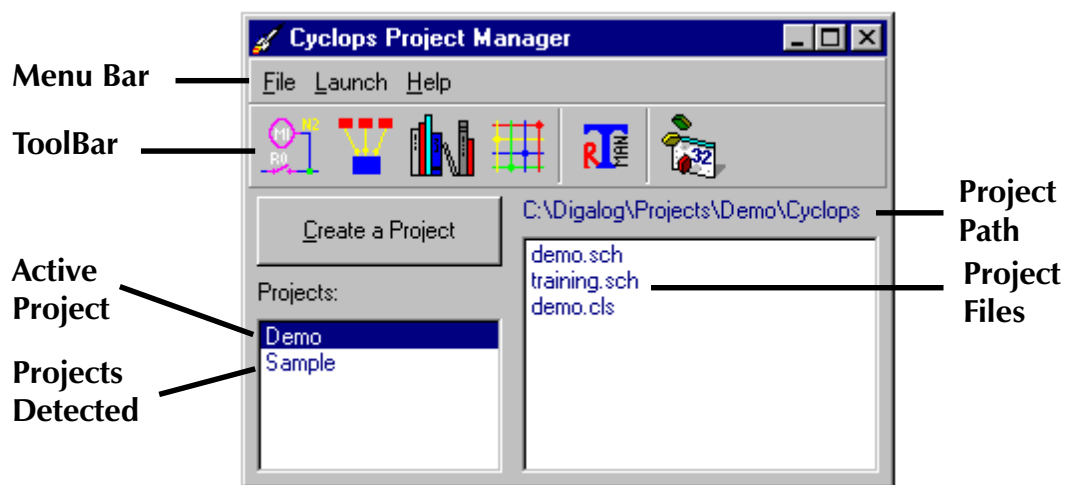
Wirelist Editor - The Wirelist Editor is the fourth program, and allows the programmer to insert additional information in the wirelist.

Tester Resource Manager - The Tester Resource Manager is the next program, and allows the programmer to view the resources available on an individual tester. This information is stored in the Windows® Registry.

Microsoft® Visual Basic® - The Visual Basic icon is the final program on the toolbar, and is used to compile the test executives.

Available Projects - The Projects window lists all of the projects in the \projects directory that contain a \cyclops subdirectory. This is the directory where all Cyclops programs store all of their files for a project by default. The button above the window is used to create a project or add a \cyclops subdirectory to an existing project. This button pops up a dialog that lists the remaining non-Cyclops projects and a text entry window for the name of the project to be created or added to. If an entry is selected in the list, then a subdirectory called \cyclops is created under that project. If a new project is entered into the text box, the project is created under the \projects name and a \cyclops directory is created under that project.

Project Files - The project files window is the list of available files in the \cyclops directory of the current project. These files have extensions corresponding to each of the Cyclops applications listed on the previous page. When one of the Cyclops Programs saves a new file, this list is automatically updated.

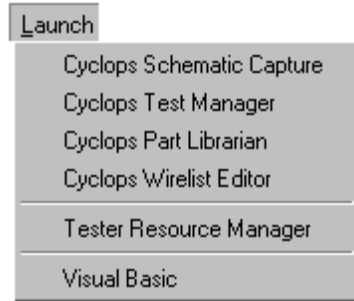


Menu Bar

File Menu - The File menu only has options to refresh the Available Projects window or Exit the Project Manager.

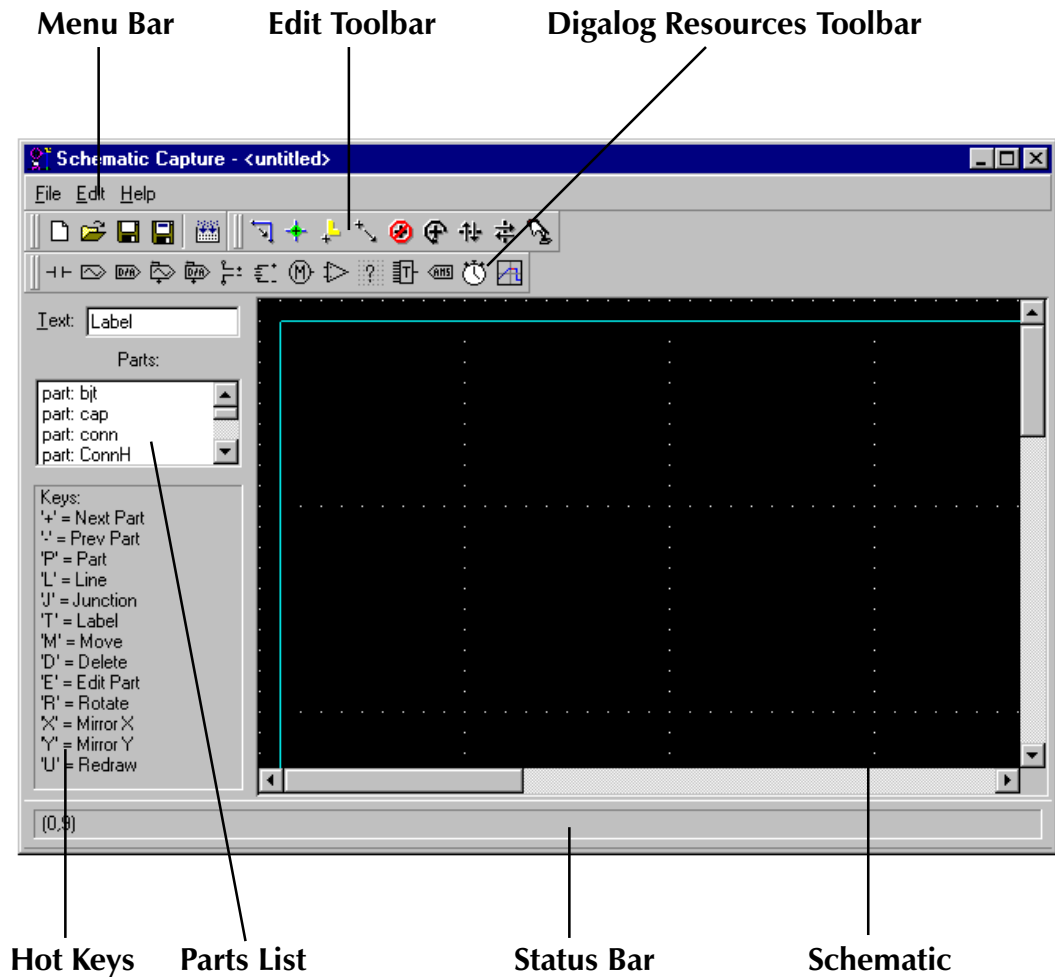
Launch Menu - This menu contains options to launch any of the programs from the toolbar as shown to the right.

Help Menu - This menu contains selections for displaying help topics for the various features and options of the Project Manager. It also has an "About" screen for determining the version number of the program.



Schematic Capture

The Cyclops Schematic Capture program is used to create a fixture schematic used to build the fixture that connects a 2040 Series Tester to a Product. It is also used to select various 2040 tester components to control during test development. It can create a plot or print of the schematic as well as a wirelist suitable for building a fixture. When a full path specification of a .sch schematic file is entered, the file is loaded into the Schematic Capture program. When first started, it loads the \lib\digalog.plb and \lib\part.plb, and all part libraries in the \cyclops subdirectory of the currently selected project.



While loading the digalog.plb file, it checks the Microsoft® Windows® Registry for available Digalog Testhead boards. If a particular board is not found, the resource parts associated with that board will not be shown in the toolbar. It continues until all of the parts are loaded. A user can add/delete/modify parts, connections, junctions and labels on the schematic.


There is a toolbar at the top of the main window with various edit functions as shown on the previous page. The status bar at the bottom of the window displays various status information. A toolbar is displayed containing all of the Digalog specific parts. The Cyclops Test Manager also has a toolbar with similar elements. A listbox is located down the left side of the main window listing all of the user parts. The list contains entries with the library name, a semicolon, and the part name for each part. Finally, the schematic window is shown with the current fixture schematic. There is a set of hotkeys for various operations in the Schematic Capture program. They are described with their operation just under the part list on the left of the main window. The hotkeys are not case sensitive. Hot keys are not recognized when the keyboard focus is on the text box above the part list.

File Menu



Toolbar



 **New** - Creates a new schematic File.



Open - Opens an existing schematic file.



Save - Saves the current schematic file.



Save As - Saves the current file under a new name.

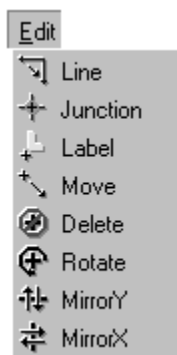


Size - Sets the drawing size of the schematic.



Create Wire List - Creates a wirelist of the current schematic.

Edit Menu



Toolbar



Line Mode - This button allows the creation of interconnections between various parts on the schematic. When selected, the user can click and drag lines. All lines on the schematic are orthogonal (horizontal and vertical). When adding new lines by dragging diagonally, two lines are created. When the horizontal or vertical plane of the first point is crossed, the plane that was crossed becomes the first line. This helps when connecting two diagonal points because there are always two ways to connect.

The hot key “L” will change the current mode to the line mode also.



Junction Button - The junction button allows the placement of junctions on the schematic. When this mode is selected, the user can place a junction with the click of the mouse. It is used to connect crossing lines which are normally not connected.

The hot key “**J**” will change the current mode to the junction mode also.



Label Button - The label button allows the placement of labels on the schematic. When the lower left corner of a label is on a line, the label becomes the name of the net used in the wirelist. Two labels with the same name tie both nets together. When a label is placed on crossing lines, the lines are tied together just like a junction. There is a text entry box above the part list to the left of the schematic window for entering the label text. When the mouse is clicked in the schematic window, the label is placed. If a label exists at the location where the click occurs, the label is changed to the text in the text box. You can tell when you are replacing a label because the color of the text changed to red when the mouse pointer is over it.

The hot key ‘**T**’ will change the current mode to the label mode also. Notice ‘**L**’ was taken by the Line command.



Move Button - The move button allows any object in the schematic to be moved. Objects are highlighted in red to show which object can be moved when a click-and-drag operation is done. When the mouse moves over a object, it turns to red. When the mouse leaves the object, it changes back to its original color. When lines are moved, they are not resized.

The hot key ‘**M**’ will change the current mode to the move mode also.



Delete Button - The delete button allows any object in the schematic to be deleted. Objects get highlighted (as in move) when the mouse pointer moves over them and change back when the mouse pointer leaves them. A click of the mouse deletes an object when it is highlighted.

The hot key ‘**D**’ will change the current mode to the delete mode also.



Rotate Button - The rotate button allows only parts in the schematic to be rotated. Parts get highlighted like as in the move and delete modes. When the mouse is clicked on a highlighted part, the part will rotate counterclockwise 90° around the placement corner.

The hot key 'R' will change the current mode to the rotate mode also.



Mirror Y Button - The "mirror y" button is used to flip a part on its y axis. The text still remains normal when this command is used. Parts get highlighted as in the move and delete modes. When the mouse is clicked on a highlighted part, the part gets mirrored on its y axis.

The hot key 'Y' will change the current mode to the "mirror y" mode also.



Mirror X Button - This is the same as "mirror y" except on the x axis instead.

The hot key 'X' will change the current mode to the "mirror x" mode also.



Edit Button - The edit button is used when the Test Manager is started along with the Schematic Capture program. Parts are highlighted as in the move and delete commands. When the mouse is clicked on a highlighted Digalog part, it sends a command to the Test Manager to show the control panel for that part with the proper part number used.

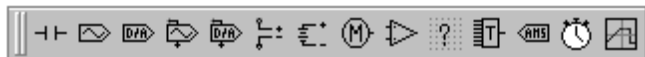
The hot key 'E' will change the current mode to the edit mode.

Other Hot Keys - '+' and '-' keys - These keys are used to change the current part reference. It will only go to available references. Used references are skipped. When all parts are used, no more can be placed on the schematic.

'U' hot key - This key is used to redraw the schematic.

Status Bar

The Status area displays the current mode and the current coordinates that the mouse pointer is at in the schematic window. It also displays which resource from the toolbox is currently selected.



Digalog Resources Toolbar

The toolbar contained in the Schematic Capture program displays the usable Digalog specific resources. The Tester Resource Manager defines the available resources in the current system, which is stored in the Windows® Registry. The valid boards have specific icons (i.e. parts) that can be placed on the schematic. These icons are located on the toolbar. Only the resources listed in the registry are displayed. Parts are described in more detail later in this manual.



















Parts

A part in the Schematic Capture program is one of two types: User Parts or Digalog Resource Parts. The main difference is that User Parts are inanimate and Digalog Parts can display tester status information and are interactive. This status information is contained in the attributes on Digalog Parts, and display status information such as the voltage set on a D/A part.

User Parts - User Parts are defined in `\lib\part.plb` for global parts available to all projects and in any `.plb` files in the `\cyclops` subdirectory of the current project for parts specific to a particular project. The Part Editor is able to create and modify parts libraries (`.plb` files). These parts are static and don't change during test development.

Digalog Parts - Digalog Parts are defined in `\lib\digalog.plb` with parts specific to various board types in the 2040 Series Analog Tester. These parts are used to interconnect a UUT to the tester patch panel. These parts have pin numbers that match the tester patch panel. Correct pin numbers for a specific part will be automatically assigned for any board placed in the Testhead. If the board slot is changed by the Tester Resource Manager, and the Schematic Capture program is restarted, the pin numbers of the part will change to reflect the new configuration. Finally, Digalog Parts have attributes used to display status information when using the Test Manager for test development.

The Individual Digalog Parts are as follows:

-  Places the next available Auxiliary Relay onto the schematic.
-  Places the next available Auxiliary FET onto the schematic.
-  Places the next available ARB onto the schematic.
-  Places the next available D/A Converter onto the schematic.
-  Places an ARB reference onto the schematic.
-  Places a D/A reference onto the schematic.
-  Places a DIO Clock onto the schematic.
-  Places the next available DIO byte onto the schematic.
-  Places the next available ADIO byte onto the schematic.
-  Places a Programmable Power Supply onto the schematic.
-  Places a Patchboard Power Supply onto the schematic.
-  Places the next available Relay Multiplexer channel onto the schematic.
-  Places the next available Matrix Relay channel onto the schematic.
-  Places the next available Matrix Relay bus onto the schematic.
-  Places the Amplitude Measurement System onto the schematic.
-  Places the Time Measurement System onto the schematic.
-  Places the Measurement Display Electronics onto the schematic.
-  Places the next available OCIO byte onto the schematic.



Places a graphic onto the schematic to select an internal or external rail voltage of the next available bank of OCIO drivers.



Places a UART interface onto the schematic.



Places a RS-422 or RS-485 interface onto the schematic.



Places a RS-232 interface onto the schematic.

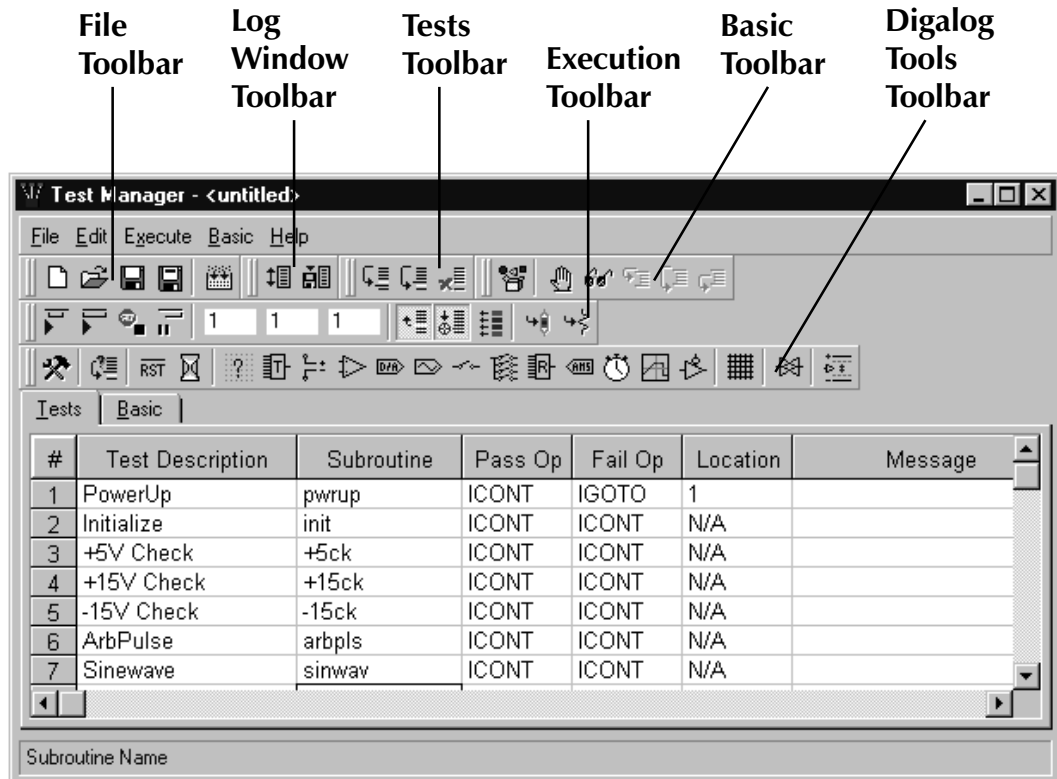


Places a CAN interface onto the schematic.

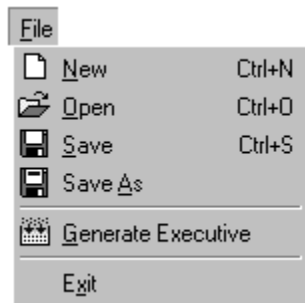
Help Menu - This menu contains entries to display **H**elp files for the Schematic Capture program. It also has an About Box selection to pop-up the title screen with the current version number of the program.

TEST MANAGER

The Test Manager is a program for creating and modifying tests and test sequences. It is the primary tool for developing test programs with Cyclops. With it, a set of tests can be created, debugged, modified, and organized. Test Manager edits and debugs code which is compatible with Microsoft® Visual Basic® Development System, version 5.0. The illustration below displays the main window for Test Manager.








File Menu



Toolbar



-  **N**ew - Creates a new test sequence file. It clears all the test sequence entries.
-  **O**pen - Opens an existing test sequence.
-  **S**ave - Saves the current test sequence.
-  **S**ave **A**s - Saves the current test sequence under a different name.
-  **G**enerate Executive - Generates an executive from Visual Basic code.

Generate Executive

When a CYX Executive is generated (clicking the Generate Executive option), several things take place:

- A default form is copied from the Digalog\Include\cyx\cyx.frm file into the current Cyclops project \cyclops\ subdirectory with the same name as the test sequence file with a .frm extension. If the file already exists, it is not replaced.
- A default cyresult.bas file is copied from the Digalog\Include\cyx\agresult.bas file into the current Cyclops project \cyclops\ subdirectory. If the file already exists, it is not replaced.
- A root.cls file and a cyclops.cls file are both created in the current Cyclops project \cyclops\ subdirectory. These files link the CYX Executive API to the test sequence list class.
- A Visual Basic project file is created with the same name as the test sequence with a .frm extension. If the file already exists, it is not replaced.

Once this procedure takes place, changes can be made to the form, cyresult.bas, or the .vbp file. The test sequence list file (.tcl) should only be

modified from within the Test Manager. Two subroutines in the .tcl file that are managed by the Test Manager that should also be left alone are:

- Public Sub Initialize Tests()
- Public Function Sequence() As Integer

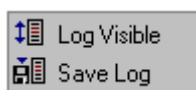
These functions interface to the CYX Executive API.

Log Window

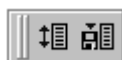
The Log Window is used to display status information for the various functions of the Test Manager. It displays information when loading and saving the test sequence list and it displays Pass and/or Fail information when executing tests. The Log Window and Log Window toolbar are shown below.

```
Finding Class_Initialize
Removing InitializeTests() and Sequence()
Removing InitializeTests()
Creating InitializeTests() and Sequence()
File Saved
```

Log Window Menu



Toolbar



 Save Log Window -

 Toggle Log Visible -

TEST MANAGER TAB

There are two main windows in the Test Manager:

- The Test Sequence List
- The Basic Code Window

Test Sequence List

The test sequence list manages information for each test being created. It displays this information on a data grid display as shown below. It allows setup and execution control on each test sequence entry as well as data storage of various parameters and message displays for the CYX executive created.

#	Test Description	Subroutine	Pass Op	Fail Op	Location	Message
1	PowerUp	pwrap	ICONT	IGOTO	1	
2	Initialize	init	ICONT	ICONT	N/A	
3	+5V Check	+5ck	ICONT	ICONT	N/A	
4	+15V Check	+15ck	ICONT	ICONT	N/A	
5	-15V Check	-15ck	ICONT	ICONT	N/A	
6	ArbPulse	arbpls	ICONT	ICONT	N/A	
7	Sinewave	sinwav	ICONT	ICONT	N/A	

Each test sequence entry in the test sequence list contains information for display, logging, and execution of a test. Test subroutines are created in the Basic Code Window when a subroutine name is given in a test sequence entry being modified. When the subroutine is executed, a data object is passed to the subroutine containing the information filled in from the test sequence entry called CyclopsData.

Sequence List Columns

- The sequence number is used primarily when using the GOTO operation in the PASS OP and FAIL OP columns. The number starts from 1 and goes to the last entry in the sequence.

Test Description - This is the description of the current test sequence entry. It is stored in the .Description property of the CyclopsData object that gets passed to the subroutine.

Subroutine - This is the subroutine to be called when the test sequence entry is executed.

Pass Op - The operation to be executed on a pass Condition.

Fail Op - The operation to be executed on a fail Condition.

Location - The sequence entry number to jump to if either the Pass Op or Fail Op is a IGOTO. It's N/A (not available) for many of the operations.

Message - A user definable message placed in the .Message property of the CyclopsData object passed to the subroutine.

Pass Op and Fail Op options

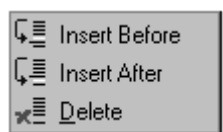
ICONT - Continue to the next sequence if Condition true.

IGOTO - Go to the sequence entry in the location column if Condition true.

ISTOP - Stop all sequence execution if Condition true.

IWAIT - Execute this sequence entry until Condition false.

Tests Menu



Toolbar



Insert Before - This tool inserts a test before the current test. If no tests exist, it creates one.

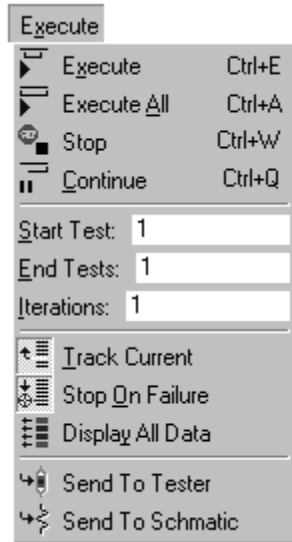


Insert After - This tool inserts a test after the current test. If no tests exist, it creates one.



Delete - This tool deletes the current test. It is disabled if no tests exist.

Execution Menu



Toolbar



Execute - This option executes the range of sequence entries specified by the **Start Test** textbox and the **End Test** textbox for the number of iterations specified in the **Iterations** textbox. It will stop at the next breakpoint if one exists.











Execute All - This option executes all of the tests from sequence entry #1 to the last entry in the sequence. It will set the **Start Test** textbox to 1 and the **End Test** textbox to the last entry in the sequence for the number of iterations specified in the **Iterations** textbox. It will stop at the next breakpoint if one exists.



Stop - This option stops execution on the currently executing test in the sequence.



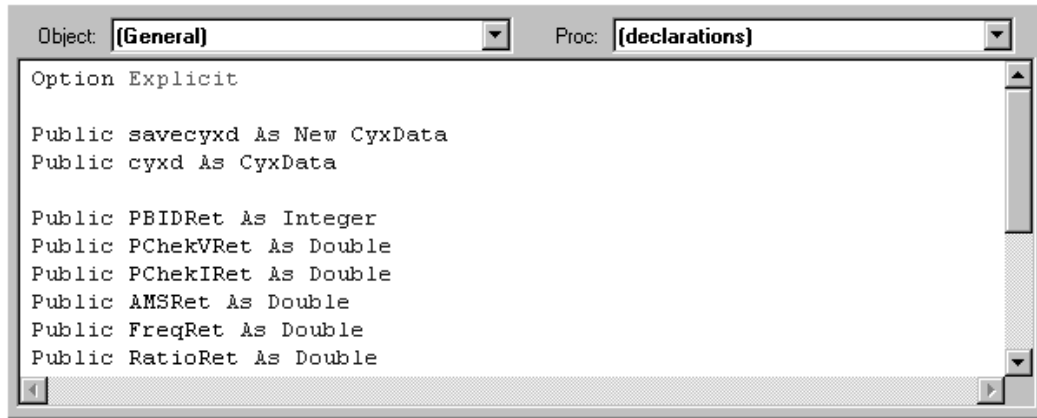
Continue - This option resumes execution at the current test or the current breakpoint in the Basic Code Window to the sequence entry in the **End Test** textbox or the next breakpoint if one exists.

-  **Start Test** - This is the first test in the sequence when the **Execute** or **Execute All** option is selected.
-  **End Test** - This is the last test in the sequence when the **Execute** or **Execute All** option is selected.
-  **Iterations** - The number of times to execute the sequence of tests when the **Execute** or **Execute All** option is selected.
-  **Track Current** - When this option is “checked” and a sequence is selected by clicking the mouse anywhere on a row of a sequence entry, the **Start Test** and **End Test** textboxes are updated with the selected row.
-  **Stop On Failure** - When this option is “checked” during **Execute** or **Execute All**, the first **FAIL** condition will stop execution.
-  **Display All Data** - When this option is “checked” during **Execute** or **Execute All**, both **PASS** and **FAIL** conditions are displayed. When this option is not “checked”, only **FAIL** data is displayed.
-  **Send To Tester** - When this option is checked and the Test Manager is running on a Series 2040 Test System, the sequence of tests and corresponding functional calls will be executed on the tester. When this option is not “checked”, the tests are not executed on the tester and the Test Manager is used solely for off-line development.
-  **Send To Schematic** - When this option is checked and the Schematic Capture program is running, the execution of functional calls will also update the parts placed on the schematic.

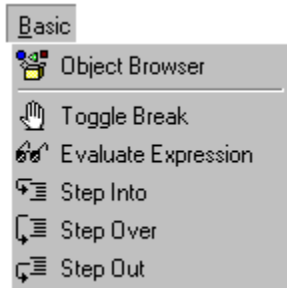
BASIC CODE WINDOW

The Basic Code Window contains all the code for each test as well as the code necessary to interface to the CYX Executive.

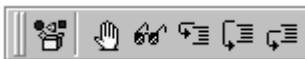
NOTE: Do not change the **PUBLIC SUB InitializeTests()** and the **Public Sub Sequence()**. These subroutines are always overwritten when the file is saved by the Test Manager.








Basic Code Menu



Toolbar



-  **Object Browser** - Opens the Object Browser window. For more information, consult the Visual Basic 5 programming documentation.
-  **Toggle Break** - Toggles the breakpoint at the current line in the Basic Editor. Execution will stop at that line when the subroutine is executed.
-  **Evaluate Expression** - Evaluate the highlighted expression. This option is used to display the value of a variable that the cursor is placed over.
-  **Step Into** - Step into the subroutine that the cursor is over.
-  **Step Over** - Step over the subroutine that the cursor is over.



Step Out - Step out of the current subroutine to the subroutine that called it.

Tools Toolbar



Edit Code - If the cursor is placed over a Digalog functional call, the tool dialog for the functional call will be displayed.



Add Condition Line - This button adds a line before the next End Sub statement that sets the .Condition property of the CyclopsData class.

The remainder of the tools are described in detail in the dialogs section of this manual.

CYX EXECUTIVE API

The CYX executive API was designed to enable the use of Cyclops and AutoGen generated code with a user defined executive. The interface is a list of functions, variables, and Visual Basic classes, some of which can be changed to suit the users needs. There are several types of code that the user must know before using the CYX executive API. There are several categories of CYX executive API code:



- Digalog - provided changeable code
- Digalog - provided unchangeable code
- Cyclops generated code
- AutoGen generated code

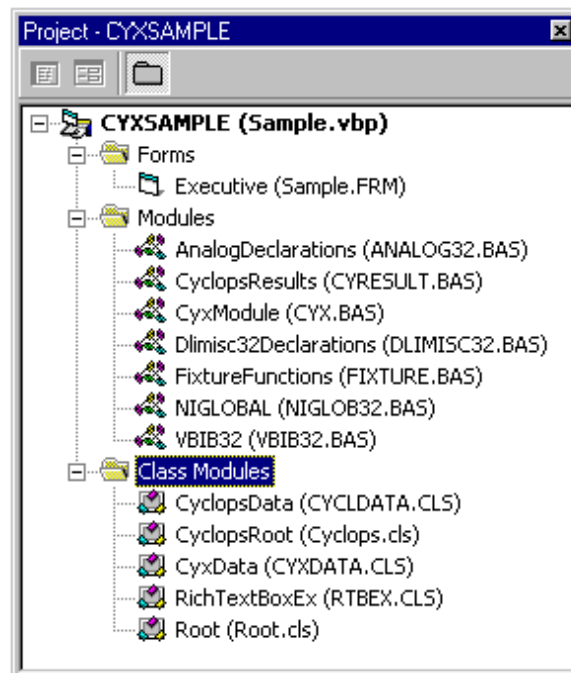
NOTE: The code generated by Cyclops and AutoGen is to be considered unchangeable.

Changeable Code

The CYX executive API contains changeable code provided by Digalog. This code is used as an example of how to interface to the CYX executive API. All of the necessary subroutines, variables, and class instantiations are implemented. The form sample.frm and cyresult.bas are changeable.

CYX Executive Form

Cyx.frm is the main form where the executive is started. There are two buttons on the form: **Start** and **Stop**. The **Start** button starts the test, disables itself and enables the **Stop** button. The **Stop** button stops the test, disables itself and enables the **Start** button. Three text boxes are used for logging of data: one for screen, one for printer, and one for log file. The



only textbox displayed during runtime is the screen. The other two are used only for data storage. There is also a pull down File menu to configure the results and to exit the program. The results for screen, printer and log file can be set to none, fail data or all data. The selections are stored in an .ini file so that each time the program is started, it remembers the settings.



Provided Unchangeable Code

The provided unchangeable code includes four Visual Basic class modules that store information about various tests. These two class modules are:

cyxdata.cls
cyldata.cls

Each of these are used for cyx tests and cyclops tests.

cyxdata.cls

The CyxData class is defined in cyxdata.cls and contains the following properties:

Name As String
Test As New Collection
PassOp As Integer
FailOp As Integer
Location As Integer
Condition As Integer

During execution of cyx in the ExecuteCYX() function, this object gets initialized by the values from InitializeCYX(). A copy of the object is made so that each execution of ExecuteCYX() will start off exactly the same way. This is used in root.cls, cyclops.cls, cyresult.bas, and the project generated code in [DIGALOG]\Projects\Sample\Cyclops\cyx.

cycldata.cls

The CyclData class is defined in cycldata.cls and contains the following properties:

- Description As String
- Name As String
- ResultVar As String
- PassOp As Integer
- FailOp As Integer
- Location As Integer
- Message As String
- Condition As Integer
- Test As String
- Value As Double

During execution of cyx in the ExecuteCYX() function, this object gets initialized by the values from InitializeCYX(). A copy of the object is made so that each execution of ExecuteCYX() will start off exactly the same way. This is used in cyresult.bas and the project generated code in [DIGALOG]\Projects\Sample\Cyclops\cyx.

cyx.bas

Cyx.bas is the unchangeable code that interfaces to the code generated by Cyclops or Autogen. This code contains the subroutines that interface to the CYX executive API. There are routines used to initialize the CYX executive API, execute the test, and an engineering notation format function. This code should not be changed because Digalog may modify it to include other functions.

The CYX executive API module contains the definition of InitializeCYX() and various constants: ICONT, IGOTO, ISTOP, IWAIT, IPASS and IFAIL. The first four are used for execution control and the last two are used to designate if a test is passed (IPASS) or failed (IFAIL).

Interface to CYX executive

The interface to the CYX executive API requires three parts: Initialization, Execution and Data Logging. Initialization is needed to setup the necessary classes with their respective data. Execution calls the subroutines located in the classes necessary to execute a set of tests. Data Logging is done for each

test in the class. The interfaces to these classes are wrapped by Visual Basic subroutines located in the `cyx.bas` module (described later).

CYX executive API initialization

In order to initialize all of the necessary data that the CYX executive API requires, some variables and subroutines must be declared, and called. The first is `InitializeCYX()`. This routine must be called at program startup or where the rest of the executive is initialized. This sets up the CYX executive API and initializes the classes that are generated from Cyclops and AutoGen. This function is declared in `cyx.bas` (described later). Once this is done, all of the AutoGen and Cyclops objects are initialized and the CYX executive can be executed.

CYX executive API execution

Once the CYX executive API initializes all of the Cyclops and Autogen classes, the CYX executive can be started by calling `ExecuteCYX()` like:

```
Dim pass As Integer
    ...
    ...
    pass = ExecuteCYX()
```

After `ExecuteCYX()` returns, `pass` contains one of two values: `IPASS` or `IFAIL`. These are defined in `Cyx.bas`. This notifies the calling routine if the CYX executive passed or failed. During the execution, there are hook subroutines that get called during execution. These routines are written by the user and are used for logging of data.

CYX executive API data logging

There are subroutines that need to be written before the CYX executive can be executed:

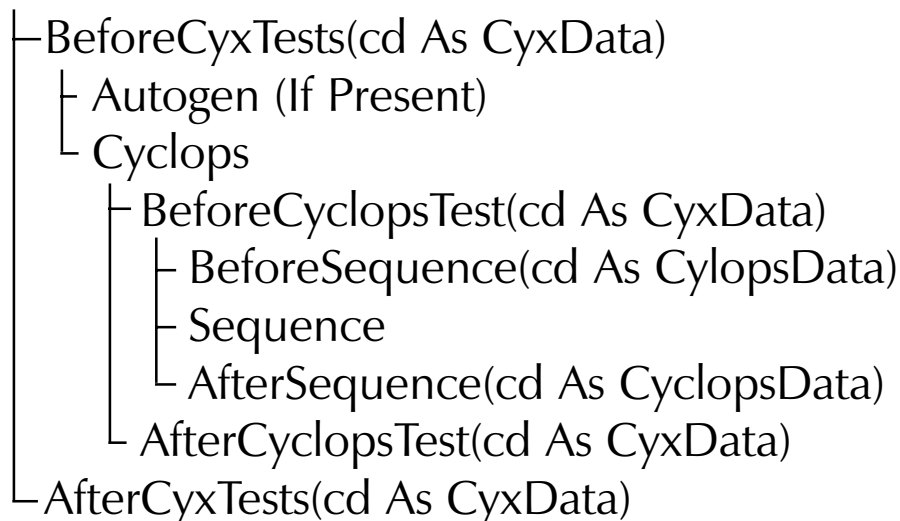
```
BeforeCyxTests(cd As CyxData)
AfterCyxTests(cd As CyxData)
BeforeCyclopsTest(cd As CyxData)
AfterCyclopsTest(cd As CyxData)
BeforeSequence(cd As CyclopsData)
AfterSequence(cd As CyclopsData)
```

They should be located in a user specific .bas module (for example, cyresult.bas) and included in the Visual Basic project. CyclopsData parameters are class objects that get passed to the routines after each test executed. They contain all of the information necessary to log information about a specific test. These will be described later. There are various properties within CyclopsData that the user can use to format and determine result of each test.

EXECUTIVE FLOW CHART

ExecuteCYX()

└ Root



The function BeforeCyxTests(cd As CyxData) is called before Cyclops or Autogen testing begins. This function can include global testing requirements such as Bar Code Reader initialization for tracking purposes.

The function BeforeCyclopsTests(cd As CyxData) gets called before any sub tests are executed. This allows the user to modify the execution flow. The cd.Location parameter is the first test to start executing. Before using this parameter, a knowledge of how many tests are in the current test object is necessary and can be obtained using cd.test.count().

The function BeforeSequence(cd As CyclopsData) sets conditions for the sequence function. Once the sequence function has been executed, the

AfterSequence(cd As CyclopsData) function is called. The pass/fail data is determined from the AfterSequence(cd As CyclopsData), and the test number is incremented (unless an execution control constant such as IGOTO, ISTOP, or IWAIT is set). The BeforeSequence(cdCyclopsData) function is again called, and the next test in the sequence is executed. This cycle is repeated for every test in the sequence. User code may also be added to the before and after functions as required.

The function AfterCyclopsTests(cyxd As CyxData) gets called after the sub test is executed. The PassOp and FailOp parameters are set from the sub test object. This is done so that the test can be interrupted if necessary. The Condition parameters are used to determine if the test passed (IPASS) or failed (IFAIL).

Next, the AfterCyxTests(cd As CyxData) function is called and the Executive logs the Pass/Fail data in whatever fashion is dictated in the cyresult.bas and agresult.bas modules.

fixture.bas

This interface contains unchangeable code to initialize and use the vacuum controller. Initialize Vacuum() initializes the vacuum to a known default state: all fixtures up. Fixture (side As Integer, position As Integer) allows the control of the fixture. Side can be DUTL or DUTR which are constants. Position can be DUTRaise or DUTLower which are also constants.

niglobal.bas, vbib32.bas

This unchangeable code is the interface used by fixture.bas.

analog32.bas

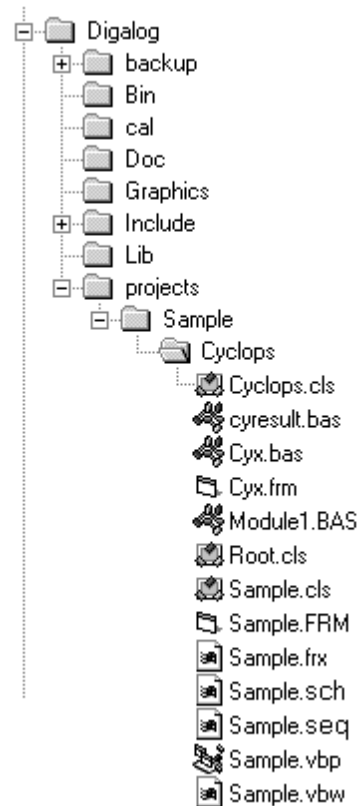
This unchangeable code contains the declarations for the analog functions used by Cyclops.

dlimisc32.bas

This unchangeable code contains the declarations for some miscellaneous functions used by Cyclops.

Directory Structure

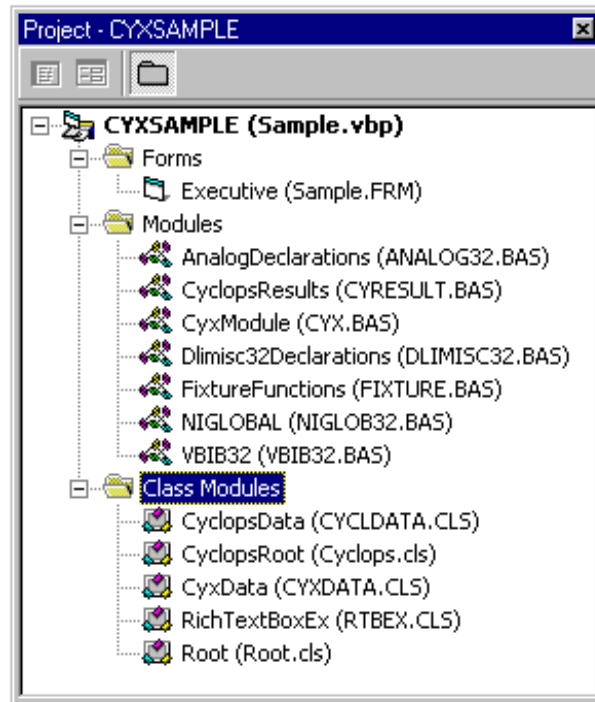
An example of the directory structure for the Sample project is shown to the right. Cyclops creates the Sample.sch and Sample.seq files in the Cyclops subdirectory of the Sample project. When the **G**enerate Executive option from the Cyclops **F**ile menu is selected, the CYX executive code is generated in the file structure shown to the right.



1. The cyresult.bas file is copied from the DIGALOG\Include\Cyx directory into the DIGALOG\Projects\Sample\Cyclops directory.
2. The Cyx.frm form from the DIGALOG/Include directory is copied and renamed Sample.frm in the DIGALOG\Projects\Sample\Cyclops directory.
3. The Visual BASIC project file Sample.vbp is created in the DIGALOG\Projects\Sample\Cyclops directory.
4. The Cyclops.cls, Root.cls, and Sample.cls class modules are created in the DIGALOG\projects\Sample\Cyclops directory.
5. The Cyclops.bas, Cyx.bas, Cyxdata.cls, Fixture.bas, GPIB.bas, Analog.bas, CyclData.bas, and Textboxe.cls modules from the DIGALOG\Include directory are linked to the Sample.vbp project file.


















When the Sample.vbp file is opened from Visual BASIC, the files mentioned in steps 1 to 5 above, appear in the project window as shown on the following page. The standard CYX Executive (as shown on page 4-11) may be run directly without further modification. The form, however, may be modified to suit the user's application. If so, the project may be saved with the desired modifications to the form. It will **NOT** be overwritten if the tests are modified under Cyclops. The only files that will be overwritten are the three generated files under the DIGALOG\Projects\Sample\Cyclops directory.

The code in Cyresult.bas may also be modified and saved within the project. Any additional user routines can be included in the appropriate spots within this file as indicated in the Executive Flow Chart shown on page 4-14.



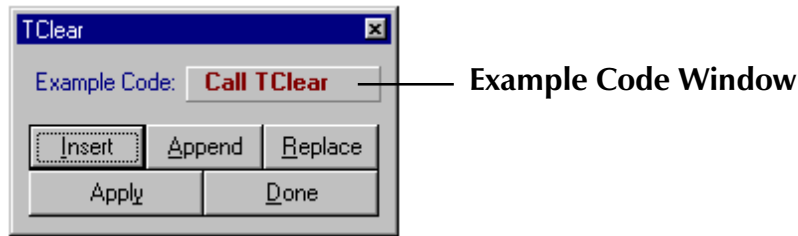
TOOL DIALOGS

The Test Manager is the central control for all of the tool dialogs. These dialogs are control panels for all of the parts and/or functions that control the Series 2040 Test System. Each dialog has information specific to the functional calls of the 2040 tester. In addition, each dialog has a common interface that is used to add or replace code in the Test Manager. The dialogs are as follows:

	Adjustable Digital I/O		Amplitude Measurement Sys.
	Arbitrary Waveform Generator		Auxiliary Relay
	D/A		Digital I/O
	Idle		Isolation Amplifiers
	Matrix Relays		Measurement Display Elect.
	Open Collector I/O		Patchboard ID
	Programmable Power Supplies		Relay Multiplexer
	Selftest Multiplexer		TClear
	Time Measurement System		Trigger Matrix

COMMON INTERFACE

The Common Interface for the Test Manager dialogs is used to modify the current sequence with the current dialog's example code. It is a simple interface for editing the current Test Manager sequence.



Example Code Window - All dialogs have an example code window that displays the functional call with the proper parameters selected from the dialog. Each time a parameter anywhere on the dialog changes, the example code is updated to reflect the change. This example code is the Visual Basic compatible code necessary to perform the functional call specified. It is used to actually execute the function as well as update the Schematic Capture part attributes that are specific to the dialog.

Insert Button - The **I**nsert button takes the current dialog's example code and inserts it into the current test subroutine. The code is inserted in the line before whatever step is highlighted.

Append Button - The **A**ppend button takes the current dialog's example code and appends it at the end of the current test subroutine. It does not matter where the highlight is on the current test subroutine.

Replace Button - The **R**eplace button takes the current dialog's example code and replaces the highlighted step in the current test subroutine.

Reset Button - If a function has a reset or default state, the Reset button will program it. Not all tools require this button.

Apply Button - There are two buttons on the main Test Manager button bar: Send To Tester and Send To Schematic. If the Send To Tester button is down, the functional call that is in the example code window is executed on the hardware. Any returned values will be displayed on the dialog. If the Send To Schematic button is down, the functional call that is in the example code window is sent to the Schematic Capture program to update the part attributes if the part is on the schematic. If present, the part turns yellow and shows the changes.

Done Button - The **D**one button just closes the current dialog.



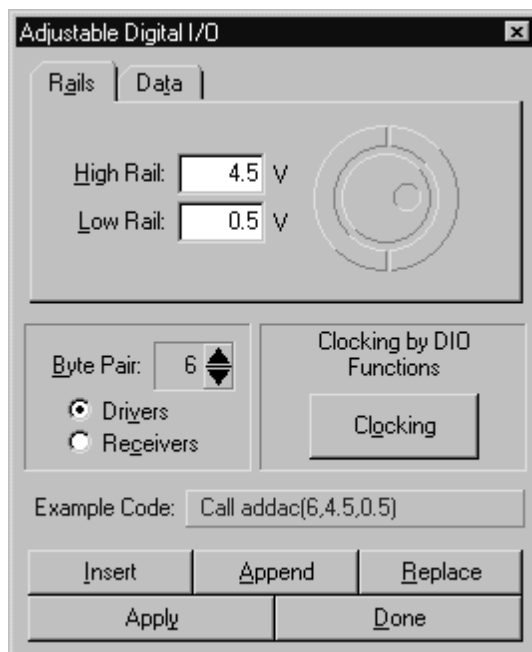
Adjustable Digital I/O

The Adjustable Digital Input/Output (ADIO) board supplies 32 channels of three-state drivers along with 32 programmable level receivers. The drivers and receivers are used for general purpose discrete (Boolean) testing at voltage levels from -15V to +15V. The functions of the DIO board are similar to the functions of the DIO board, except that the DIO boards are limited to TTL level signals where the ADIO boards are programmable from -15 to +15 volts. The clocking for the drivers and receivers for both boards is identical, and the same functional calls are used. A master for the DIO and ADIO is determined from the same functional call. Since these functional calls are identical for both boards and are covered in the DIO section (Page 44), only the ADIO specific functional calls are discussed in this section.

The dialog shown to the right is used to set the high and low rail voltages for the drivers and receivers. The ADDAC functional call is used to set rail voltages for each individual pair of Drivers. The byte pairs to be programmed are determined by using the spin control on the left side of the dialog.

The voltage rails are programmed by clicking the text tool in the desired textbox and using the jog/shuttle control or entering a value manually. If the programmer clicks the option button adjacent to the Receiver label, the receivers can be programmed in the same manner using the ARDAC functional call.

If the Data tab is selected, the dialog changes as shown on the next page. The ADData functional call is used to set the outputs of the ADIO drivers to either the high output level, low output level, or a three-state condition. The byte to be programmed can be selected using the spin control as before.



The output data is entered by clicking the text tool on the **O**utput Data textbox and using the jog/shuttle control or entering the data manually. The **T**hree-State Data, used to enable or disable each bit in the selected byte, can be entered in the same manner. If the **R**ecivers option button is selected, the ARData functional call is active and the receiver bytes can be programmed. The spin control is used to select the byte to read, and the received data is returned in the “Received Data” textbox. If the returned data is between the rails set by the ARDAC functional call, a corresponding byte (bit pattern) is returned in the “NML Data” textbox.



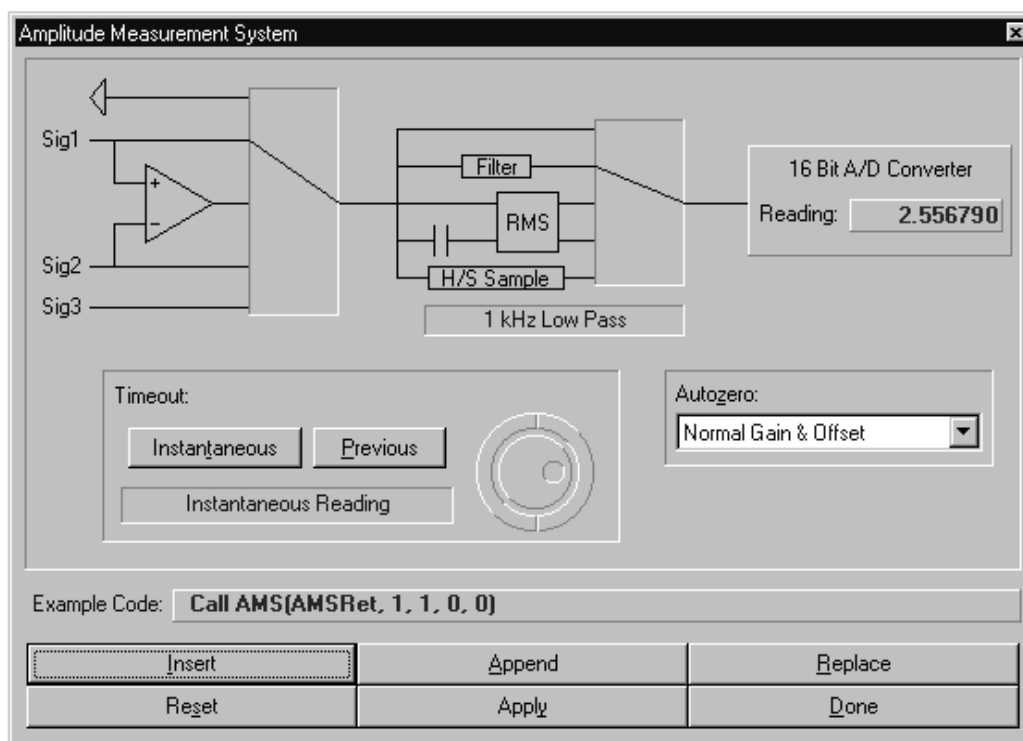
Consult the Series 2040 Windows Programming Manual for more information.



Amplitude Measurement System

The Amplitude Measurement System (AMS) uses an analog to digital converter with 16-bit resolution to measure voltages. It has a number of different modes that allow the programmer to capture and condition the voltage being measured. These are straight DC, filtered DC, DC and AC coupled RMS, and a high speed capture using a low acquisition time track and hold amplifier. The AMS has the capability to start conversions based on a trigger supplied by the MDE module. Voltages from one to four input channels can be scanned, converted, and stored in the computer's memory.

The programmer can select the signal to be measured by clicking the mouse cursor on a rectangular box adjacent to the graphic in the upper left of the dialog. In a similar manner, the rectangular box to the right of the mode graphic in the upper center of the dialog can be used to select the mode.



Modes 0, 1, & 3

The programmer also has the option of taking an instantaneous reading or a previously triggered reading. If a previously triggered reading is desired, the jog/shuttle control can be used to select the time to wait in milliseconds.

Modes 2 & 4

With the RMS modes, the user has the option of the jog/shuttle control to select the integration time (length) for the reading.

Autozero

The Autozero parameter is programmed using the listbox on the right side of the dialog. The reading returned by the AMS call is displayed in the textbox in the upper right corner of the dialog.

**Consult the Series 2040 Windows Programming Manual
for more information.**



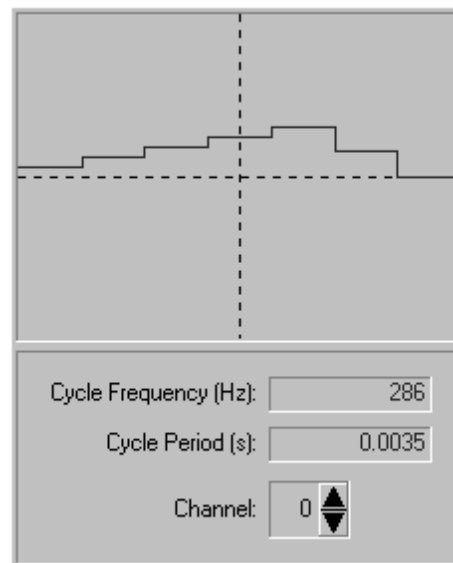
Arbitrary Waveform Generators

There are two Arbitrary Waveform Generators on each Analog Source board. These ARBs provide most of the waveforms for the system. The Arbitrary Waveform Generator dialog covers five different functional calls. When the dialog is opened, three of the functional appear on folder tabs to the lower left of the dialog. These calls are ARB, ARBFreq, and ARBSin.



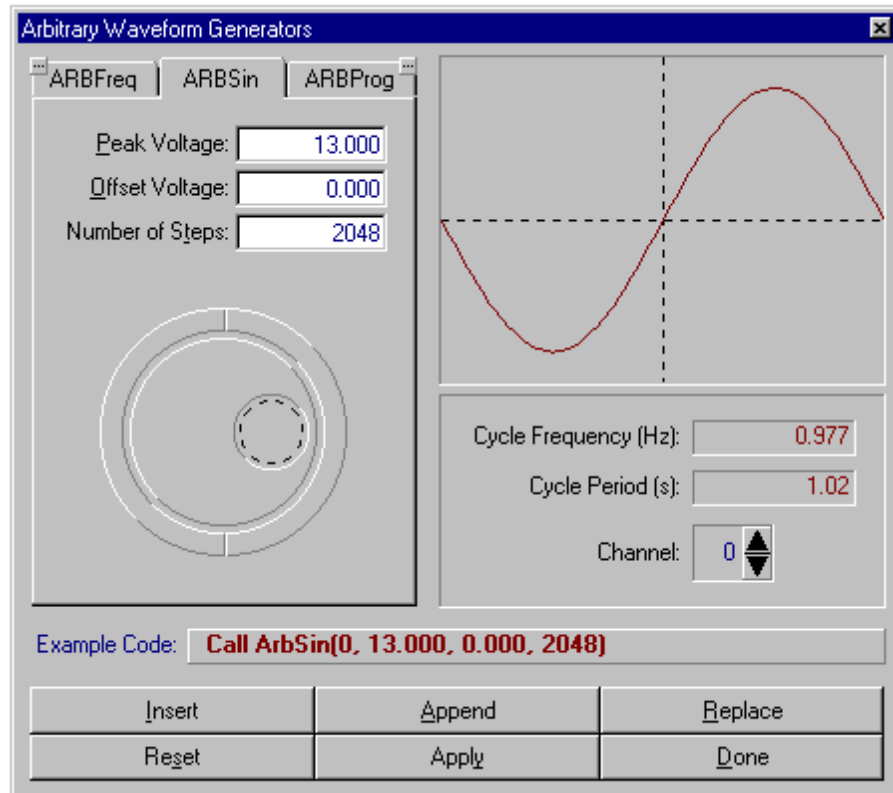
To view two additional calls, ARBProg and ARBPulse, click the mouse pointer on the small tab in the corner of the ARBSin tab, and the functional calls will rotate from right to left. If the programmer clicks on the small tab once, the three middle functional calls will be displayed. Note that the small tabs appear on both the ARBFreq and ARBProg functional calls allowing the programmer to “page” either way. When the ARBPulse call appears on the right tab, the small tab disappears from the right folder.

The upper portion of the right side of the dialog contains an oscilloscope window. This area displays a “thumbnail” representation of the waveform generated by the programmed device, using the calls generated by this dialog, as shown to the right. Labels for the frequency and period of the waveform (if one has been generated) are also displayed.



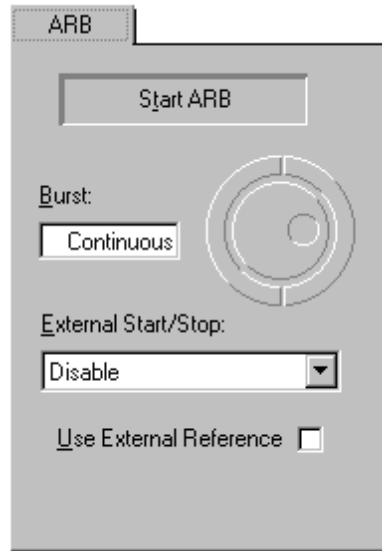
The spin control is used to select the desired ARB channel. The example shown on the previous page is the product of an ARBProg and ARBFreq call. The specific parameters are shown in the ARBProg and ARBFreq sections of this dialog.

The functional call currently being programmed will have its tabbed folder brought to the front, indicating it has focus. In the graphic below, the ARBSin functional call is active and the Example code textbox shows an example of the ARBSin functional call. As the programmer switches from call to call, the controls on the dialog will appear or disappear to match the functional call selected. Each of the five variations of this dialog will be briefly discussed.



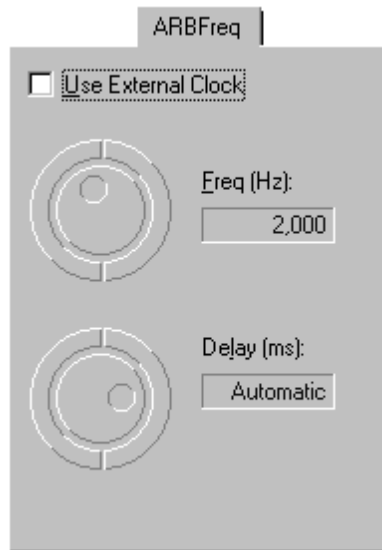
ARB

The Start ARB command button on the top of the ARB tab is used to start or stop the ARB. The **B**urst parameter can be programmed using the jog/shuttle wheel or by clicking the text tool on the Burst textbox and entering a value manually with the keyboard. The External Start/Stop parameter has a listbox for making a selection. The **U**se External Reference parameter uses a checkbox to select or deselect the external reference. The ARB channel may be selected using the spin control adjacent to the “Channel” label on the right of the dialog.



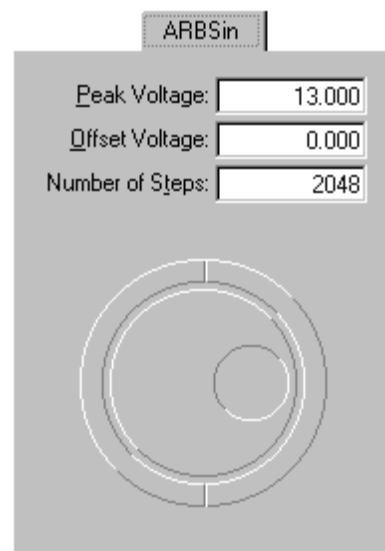
ARBFreq

The ARBFreq tab utilizes two jog/shuttle control for the clock **F**req and start **D**elay parameters, as shown to the right. The frequency is adjusted using the upper jog/shuttle control. If the **U**se External Clock checkbox is enabled, the Freq textbox displays “External.” The Delay parameter may be determined using the lower jog/shuttle control.

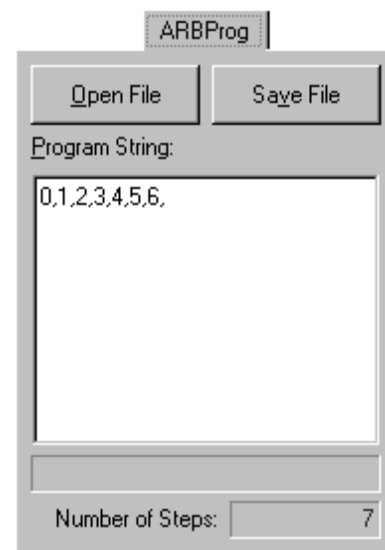


ARBSin

When the ARBSin functional call is being programmed, three textboxes for **Peak Voltage**, **Offset Voltage**, and **Number of Steps** are displayed. These values can be entered by clicking the text tool in the desired textbox and using the jog/shuttle control or manually entering a value with the keyboard. The channel may be selected using the spin control on the lower right of the dialog. When a waveform is determined, it is displayed in the oscilloscope window in the upper right of the dialog.

**ARBProg**

When this functional call is selected, two command buttons appear to **Open** an existing string or **Save** the string that appears in the large textbox below the controls. This call also allows the programmer to enter a new string of values in the large textbox or modify the existing string that was opened. When a string is saved, it may be opened at any time and modified, or it may be used on a different ARB channel. The spin control on the lower right of the dialog is used to select an ARB channel.

**ARBPulse**

This functional call has the same options as the ARBProg functional call except that the string can contain only zeros or ones.

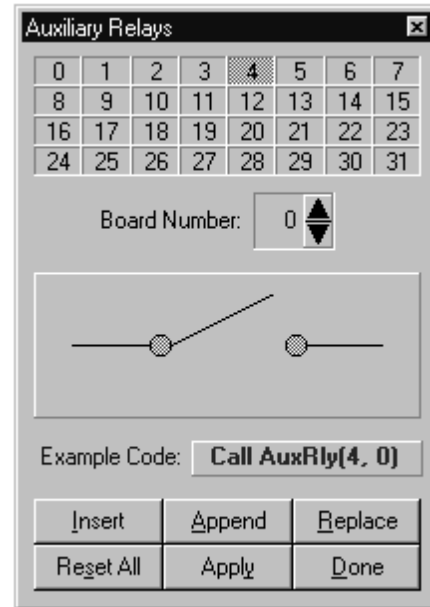
**Consult the Series 2040 Windows Programming Manual
for more information.**



Auxiliary Relays

Auxiliary Relays have both sides of a Form A contact relay connected to the Patchboard. Auxiliary FET boards use opto-isolated, bi-polar, VMOS FETs for power switching. High Current Relays and High Current FETs are useful for applications requiring up to 10 Amps of switching current. There are 32 switches per board. All Auxiliary Relay boards in the system respond to the same functional calls.

NOTE: AUXFETs have 1500pF open circuit capacitance, and are not useful for digital switching. AUXRelays can be damaged by the high in-rush currents of capacitive loads.



The spin control selects which relay (or FET) bank is being programmed. The mouse can be clicked on any channel to toggle it open or closed. If the Reset All button is pressed, the AUXRly Reset functional call is displayed, which will open all of the AUXRlys/AUXFETs/High Current Relays/High Current FETs in the system.

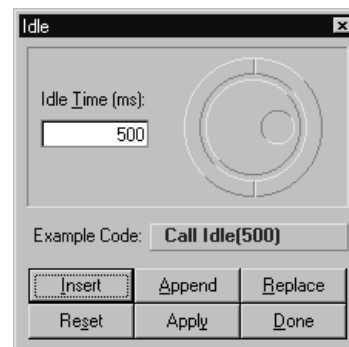
Consult the Series 2040 Windows Programming Manual for more information.



Idle

The Idle dialog is used whenever a programmed hardware delay is required within a test program. The idle time can be selected using the jog/shuttle wheel, or by clicking the text tool inside the idle textbox and entering the time with the keyboard.

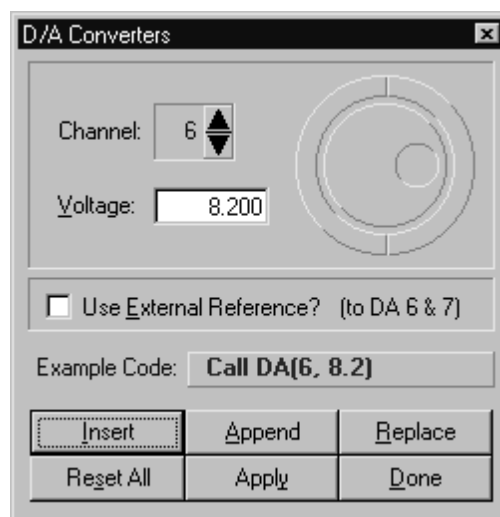
Consult the Series 2040 Windows Programming Manual for more information.





D/A Converters

There are 12 D/A converters on each Analog Source board. D/A pairs 4 & 5, 6 & 7, 8 & 9, and 10 & 11 have external reference inputs tied to each pair and brought out to the Patchboard. Each channel is automatically calibrated using the TMUX (Selftest Multiplexer), which is calibrated to the TDAC (Testhead D/A Converter) during a Selftest calibration. A channel can be selected using the spin control located adjacent to the "Channel" label. The voltage can be determined using the jog/shuttle control, or by clicking the text tool inside the Voltage textbox and manually entering a voltage with the keyboard. If the External Reference checkbox is checked using the mouse (or hotkey), the channel parameter in the Example Code window becomes negative indicating the selected channel and its adjacent channel (as described above) share an external reference.



**Consult the Series 2040 Windows Programming Manual
for more information.**



Digital I/O

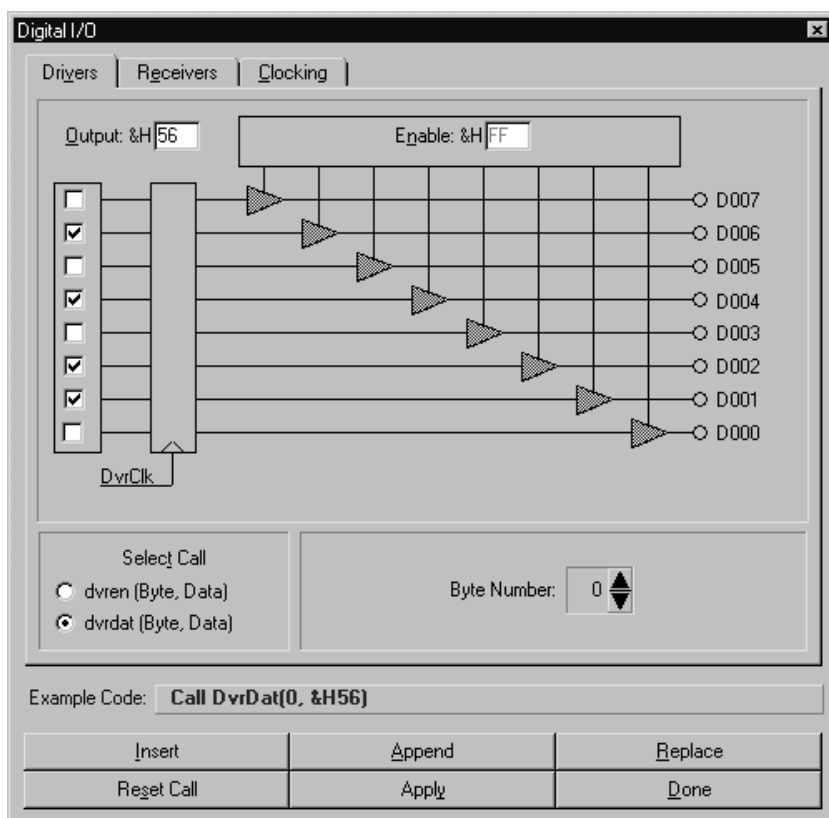
The Digital Input/Output (DIO) board provides the programmer the capability to drive and read back TTL signals to the Unit Under Test (UUT). Each board has 32 drivers and 32 receivers. The functional calls associated with the DIO divides the drivers and receivers into eight bit bytes for ease of programming. Driver outputs can be placed into a high impedance condition on an individual basis and are protected by series resistors. Receiver inputs are protected from overvoltage conditions using diodes.

Data out to the Patchboard pins can be either clocked with an external signal or clocked from the tester's computer. Data can be strobed into the DIO using a computer command, an external signal, or can be derived from the output clock. The receiver strobe can be delayed by using the on-board delay.

When using multiple DIO/ADIO cards in a system, all clocks and strobes come

from the board designated as the “master” card. All other cards in the system will get their clock and strobe signals from the digital T-Bus motherboard, which is driven by the master. There is only one master in the system at any time. There must always be a master designated, even if there is only one DIO/ADIO board in the system.

The Digital I/O dialog covers seven different functional calls for the DIO boards. These seven calls are divided into three categories, **Drivers**, **Receivers**, and **Clocking**.

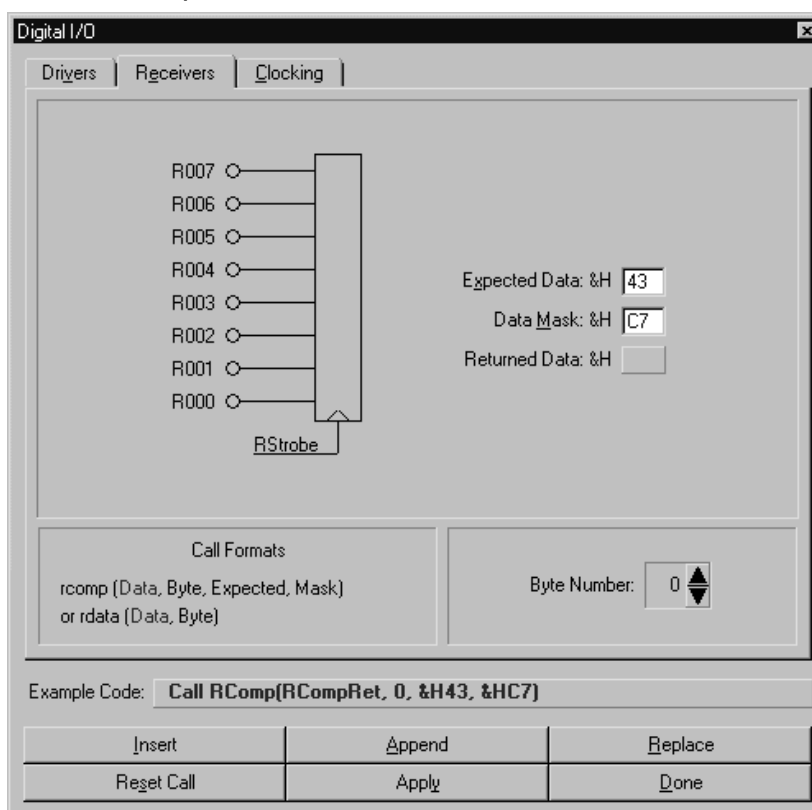


Drivers

When the Drivers tab is selected, the screen on the next page is displayed. When the option button adjacent to the `dvrdat` label is enabled, the individual output bits for the drivers can be programmed by clicking the checkboxes on the upper left of the dialog. The output value in hexadecimal is displayed in the textbox in the upper left corner of the dialog. The hex value may also be entered directly by

clicking the text tool in the textbox and manually entering the value. The byte number can be incremented using the spin control in the right-center of the dialog. Note, the designation for the output data bits changes to correspond to the selection of a different byte. In this manner, all of the bytes on all of the boards in the system can be programmed with individual bit patterns. However, a separate call must be used for each byte.

When the option button adjacent to the “dvren” label is enabled, the output drivers can be enabled. The spin control can be used to select the proper byte. The output bits to be enabled for the individual bytes can be determined by clicking the text tool in the textbox adjacent to the “Enable” label and manually entering a hex value corresponding to the desired bit pattern

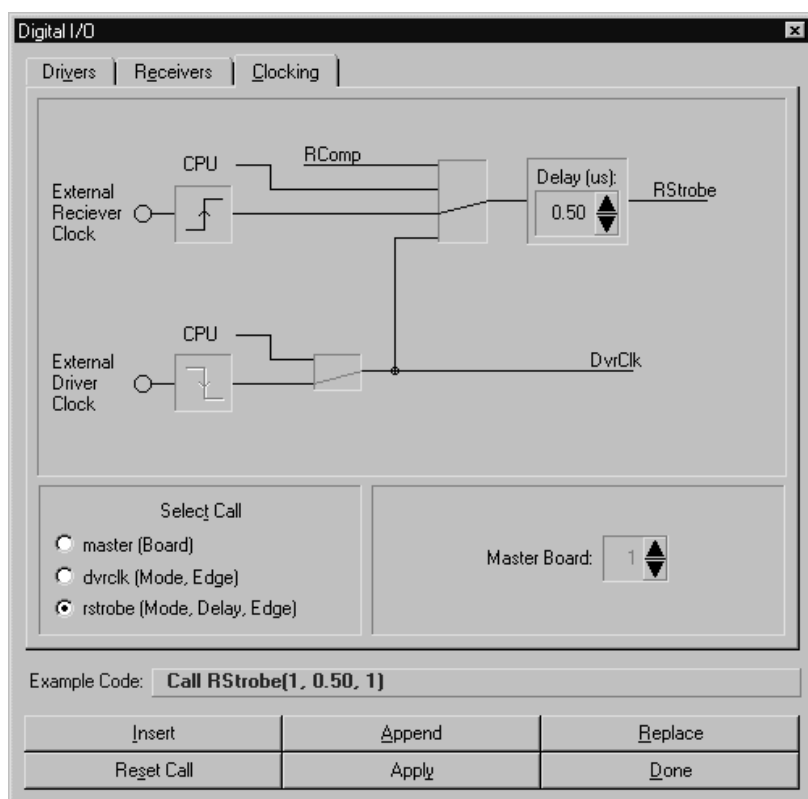


Receivers

When this tab is selected, the dialog changes to display controls for the receivers. Note, the rdata functional call first appears in the “example

Code” textbox. The spin control in the center of the dialog is used to select the desired byte. The bit designations for the receiver bits change to correspond with the selected byte. The return variable will contain data from the byte being read.

If the rcomp functional call is desired, the data in the textboxes for “Expected Data:” and “Data Mask:” must be entered in hexadecimal. Also note, the functional call changes to the rcomp call and the parameters entered in the textboxes appear in the call.



Clocking

When this tab is selected, the dialog changes to display controls for the driver clock, receiver strobe, and allows the programmer to designate a “master” DIO/ADIO Board. When the optionbutton adjacent to the “Master” label is selected, the programmer can use the spin control to designate which DIO/ADIO board in the system will be master. If only one DIO/ADIO board is present in the Testhead, the selection in the spin control will be “1”, and it will appear ghosted.

If the optionbutton adjacent to the “dvrclk” label is selected, the programmer may select a CPU clock or External Driver Clock by clicking on the rectangular box where both paths meet in the dialog. If the External Driver Clock is selected, the rectangular box adjacent to the “External Driver Clock” label becomes active, and the programmer can select the desired slope by clicking on this box.

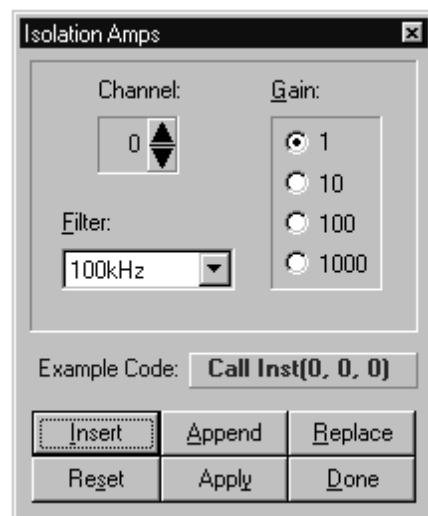
If the optionbutton adjacent to the “rstrobe” label is selected, the rectangular box adjacent to the Delay frame becomes active, and the programmer can select a source for the receiver strobe by clicking on this box. If an External Receiver Strobe is selected, the desired slope can be selected by clicking on the box adjacent to the “External Receiver Strobe” label. A delay for the strobe in 0.01 uS increments can be determined using the spin control in the Delay frame.

Consult the Series 2040 Windows Programming Manual for more information.



Isolation Amplifiers

There are four differential isolation amplifiers in the system. The differential inputs are brought in from the Patchboard. The single-ended outputs, with an associated ground, are also brought out to the Patchboard. Each amplifier has programmable gain stages and programmable filters, and can be read back with the Selftest Multiplexer (TMUX). The board containing these amplifiers must always occupy slot #1 since the board will also contain the TMUX for the Selftest utility.



The isolation amplifier channel is selected using the spin control in the upper left of the dialog. The gain is selected by clicking on the appropriate optionbutton in the “Gain” frame. The value for the programmable filter is determined using the drop-down listbox under the “Filter” label.

Consult the Series 2040 Windows Programming Manual for more information.

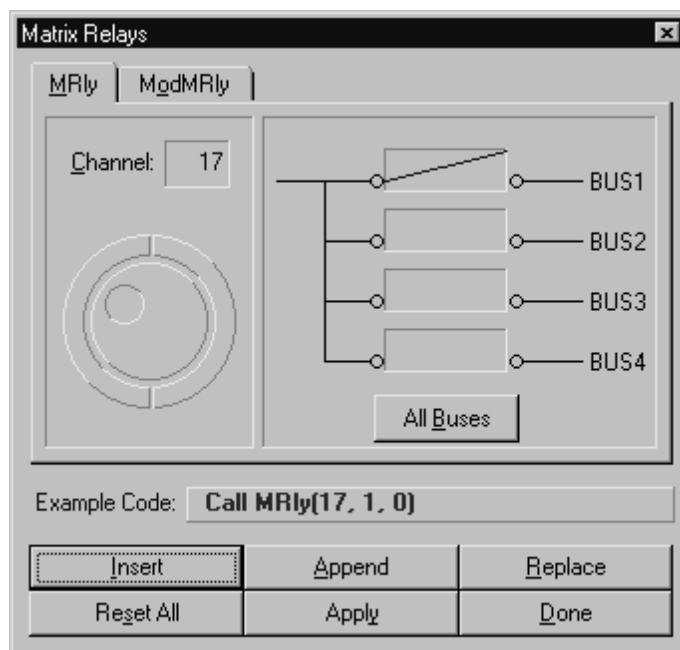


Matrix Relays

The Matrix Relay boards contain 256 relays organized in a 64 channel by 4 bus matrix. Up to 16 Matrix Relay boards may be placed in a Testhead at a time. The Matrix Relay boards may operate in any one of three possible modes set by the MODMRly functional call. Mode 0 (normal) is the most flexible mode of operation. It allows any channel to be connected to any bus. With Mode 1, a “Break-Before-Make” feature can be enabled or disabled. The duration of the break is programmable. Mode 2 (RMUX Emulation) emulates the Relay Multiplexer assembly.

The relays used on this board are instrument grade reed relays. They can be individually switched on or off using the MRly functional call. The power-on and reset state of all relays is open, with the break before make feature disabled.

Each slot of the Digalog Testhead has 68 connections to the Patchboard receiver. Of the 68, 64 of these connections are used for the pin side of the matrix. The remaining four receiver connections are used for the bus side of the matrix. This allows the user to connect any number of 64 points to any of four “buses.” This also means that any of the 64 points can be connected to any other of the remaining 63.



A typical application of this board is to replace the auxiliary relay board when switching to a common bus is desired. The Matrix Relay board allows the programmer a higher relay density per analog Testhead slot used. Such an application might be in pulling up (or down) high voltage UUT inputs (or outputs). Another application for this board might be a multiplexer for external equipment used in highly specialized applications.

MRly

When this tab is selected on the dialog, the individual channels may be programmed. The jog/shuttle control can be used to select the desired **C**hannel. By clicking on one of the rectangular boxes adjacent to the bus numbers, the box becomes active and can be used to make or break a connection to the bus. If a channel needs to be connected or disconnected from all of the buses, the All **B**uses command button can be used.



MODMRly

This functional call is used to determine a mode for the selected board. The spin control can be used to select the proper board. The Mode can be selected from the drop-down listbox with the mouse. If any mode is selected except normal, a Break **T**ime may be set in milliseconds using the jog/shuttle control.

MRlyReset

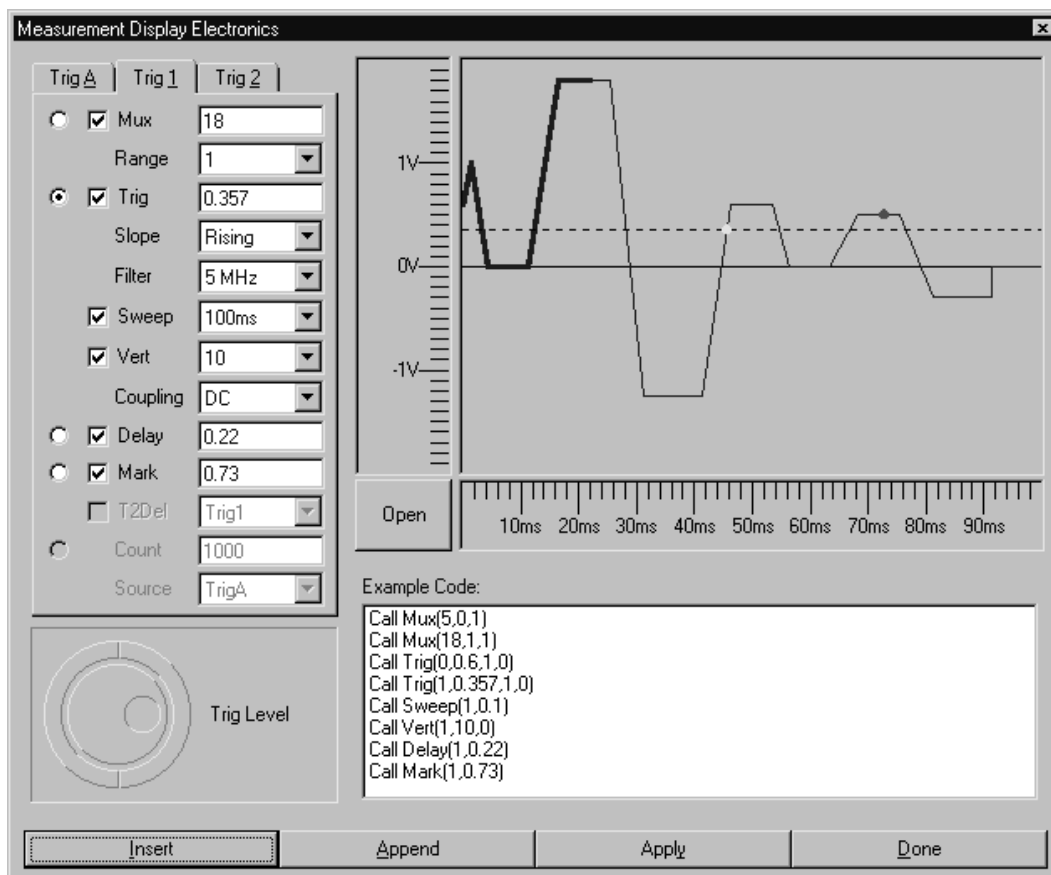
If it becomes necessary to reset all of the MRly boards in the system, use the Reset All command button to display the MRly Reset functional call.

Consult the Series 2040 Windows Programming Manual for more info.



Measurement Display Electronics

The Measurement Display Electronics (MDE) is integrated into the measurement system to provide waveform measurement capability. The MDE provides a “picture” of the waveform to be measured, and allows the test engineer to position measurement marks and delays.



The MDE is an alternate trace oscilloscope which is triggered by TrigA and displays Sig1 on Trace1 and Sig2 on Trace2. The “Z” axis is modulated with intensified Trig1 and Trig2 marks (TRIG Call), and a voltage measurement mark (MARK call). The “Z” axis is also modulated from the start of each trace with trigger inhibiting, intensified analog delay bands: Delay1 on Trace1 inhibits Trig1 and Delay2 on Trace2 inhibits Trig2. Both are positioned by the DELAY functional call. The sweep on each trace is set by the SWEEP call in seconds for total sweep time. The vertical amplitude is set on each trace with the VERT call. The T2DEL call sets the trigger mode for Trace2. Additional MDE waveform files may be created and loaded into the dialog display using the

“Open” command button. It may be beneficial to open an existing waveform to further understand the effects of the functional calls on a more representative waveform.

Since the MDE requires several functional calls to measure and/or display a waveform, the dialog is presented differently, and is based on the incoming triggers rather than the individual calls. The three triggers will be discussed individually. Each one of these three triggers appears on a tab on the middle left of the dialog. When the individual tabs are selected, only those functional calls pertaining to the programming of the selected trigger are active, and the remaining calls appear ghosted.

TrigA

When this tab is selected, only the MUX and TRIG functional calls are live. When the optionbutton adjacent to the “Mux” label is clicked, the Mux inputs become active and the channel may be selected with the jog/shuttle control on the lower left corner of the dialog, or a value can be manually entered using the keyboard. The Mux range is entered using the adjacent listbox. In a similar manner, the Trig inputs become active and an incoming signal can be selected for TrigA using the jog/shuttle control or the keyboard. The slope and filter for TrigA can also be selected using the adjacent listboxes. When the checkbox adjacent to a functional call is checked using the mouse, the functional call with the selected parameters is entered in the Example Code textbox on the lower right of the dialog as shown in the graphic on the previous page.

Trig1

When the Trig1 tab is selected, the functional calls for SWEEP, VERT, DELAY, and MARK become active and can be programmed in the same manner as the MUX and TRIG functional calls. As before, sample code with the selected parameters is added to the Example Code textbox when the checkbox adjacent to the functional call is checked.

Trig2

When this tab is selected, all of the functional calls for the dialog become active, and can be programmed in the same manner as before. The waveform displayed in the oscilloscope portion of the dialog will vary with the parameters of the MDE functional calls.



Open Collector I/O

The Open Collector I/O board (OCIO) is designed to drive and receive high voltage logic or “digital” signals. The board provides 64 open-drain outputs, arranged into byte-sized segments, each capable of handling up to 50 volts. Each output pin may also serve as an input, either to read the state of the output driver, or just as a receiver. The high voltage and current ratings of the OCIO board allow it to serve as a relay driver board. When this option is selected, the dialog shown to the right is displayed with option buttons for each of the OCIO functional calls. Each of the functional calls will be briefly discussed.

OCEn

This functional call is used to enable/disable the output from individual channels of the OCIO board. The spin control is used to select a byte, and the textbox adjacent to the functional call is used to enter the eight bit number specifying the bits (channels) to be enabled in each byte.

OCRail

The OCRail functional call is used to switch between the internal and external rail supply voltages. There are 64 channels per board, arranged into two banks of 32 each. The bank for this call is selected using the spin control, and the internal 5V or external rail is selected using the option buttons adjacent to the functional call on the dialog.

OCData

This functional call is used to latch the OCIO output registers with a byte of data. The byte to latch is selected using the spin control, and the 8-bit number (byte) specifying the logical high bits is entered in the textbox adjacent to the functional call on the dialog.

OCRead

The OCRead functional call is used to strobe the receiver latches and retrieve data from them. The byte to strobe (read) is selected using the spin control, and the latched data is returned in the textbox adjacent to the functional call on the dialog.

OCPut

This functional call is used to write a byte of data to output latches without clocking the output drivers. The data may be clocked later using the OCData or OCClk functional calls. The byte to write to is selected using the spin control, and the 8 bit number (byte) specifying the logical high bits is entered in the textbox adjacent to the functional call on the dialog.

OCGet

The OCGet functional call is used to retrieve a byte of data from a receiver latch previously clocked. The byte to read is selected using the spin control, and the data is returned in the textbox adjacent to the functional call on the dialog.

OCClk

This functional call is used to simultaneously latch the output drivers of all of the OCIO boards present in the system. No parameters are required for this call.

OCStrobe

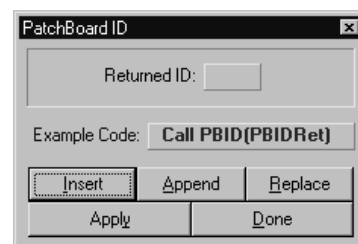
This functional call is used to simultaneously latch the input registers of all of the OCIO boards present in the system. No Parameters are required for this call.

**Consult the Series 2040 Windows Programming Manual
for more information.**

**Patchboard ID**

Is possible to assign an 8-bit Patchboard identification code to any fixture assembly.

The code is hard-wired into each fixture assembly. Identification codes fall into the range of 0 - 255 (00 - FF hex) and are coded by



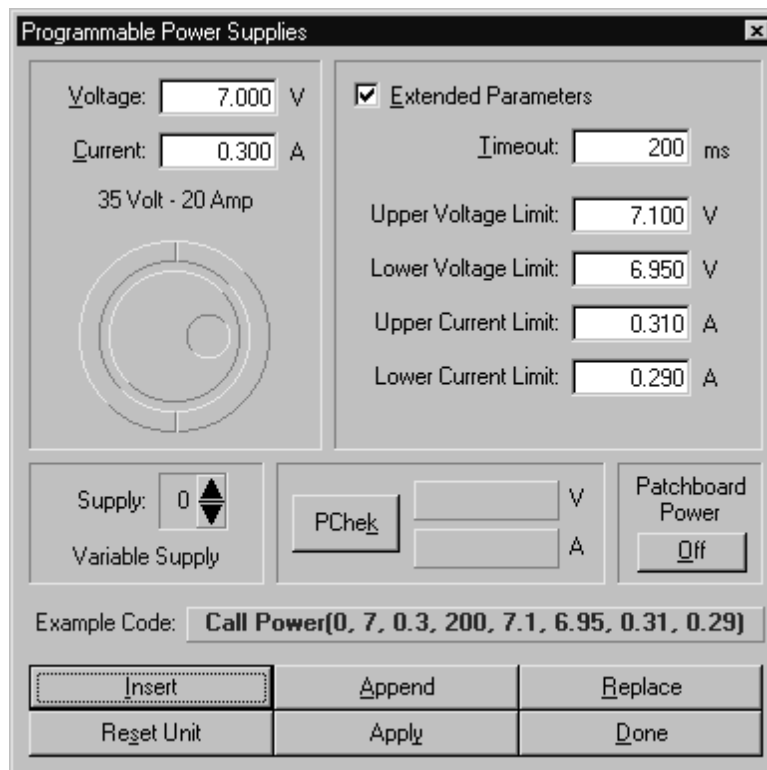
grounding pins (seen as logic highs) that correspond to the bits set in the desired identification code. An ID GROUND pin is provided adjacent to the identification pins for this purpose. The PBID functional call is used to read a fixture's identification code when it is installed on the Patchboard.

**Consult the Series 2040 Windows Programming Manual
for more information.**



Programmable Power Supplies

The programmable power supply system is composed of two major components: the power supply and the controller. The controller can be made to work with any programmable power supply as long as the programming cable and configuration card are available for that supply. The controller is universal while power supply, configuration card, and programming cable are a matched set. Commands are given to the controller via an opto-isolated current loop. The output of the supply goes to the controller, where it is switched by a mercury-wetted relay to the Testhead. The cable between the controller and the Testhead, besides providing a path for



the output, has a fault loop. This loop, if broken, will cause the controller to shut down the power supply and report an error to the computer the next time the computer tries to give it a command. The fault loop is broken by the controller itself anytime that the controller detects an error. This signals to the other controllers that they too should shut down. Programming of the power supplies is accomplished with the Power, PowerReset, and PowerUUT functional calls. The PowerUUT, PowerReset, and PChek functional calls are also compatible with GPIB and HPIB controlled power supplies.

The voltage and current values can be selected by clicking the text tool in the appropriate textbox and manually entering a value with the keyboard or using the jog/shuttle control. If the **E**xtended Parameters checkbox is checked, the functional call switches from a PowerUUT call to a Power call and the upper and lower voltage and current limit textboxes become live. The **T**imeout textbox also becomes active at this time. Values may also be entered by clicking the text tool in the appropriate textbox and manually entering a value with the keyboard or using the jog/shuttle control.

The programmable power supply to be programmed can be selected using the spin control located directly below the jog/shuttle control. Information about the type and range of the power supply will be displayed on the form. If the outputs are to be monitored, click on the **P**Chek command button and the voltage and current for the selected supply will be displayed in the textboxes adjacent to the button. The PChek functional call will also be shown. The command button to the right of the PChek textboxes is used to turn the fixed Patchboard Power Supplies **O**n and **O**ff, and the PowerPB functional call will be displayed. The Reset Unit command button will display and invoke the PowerReset functional call.

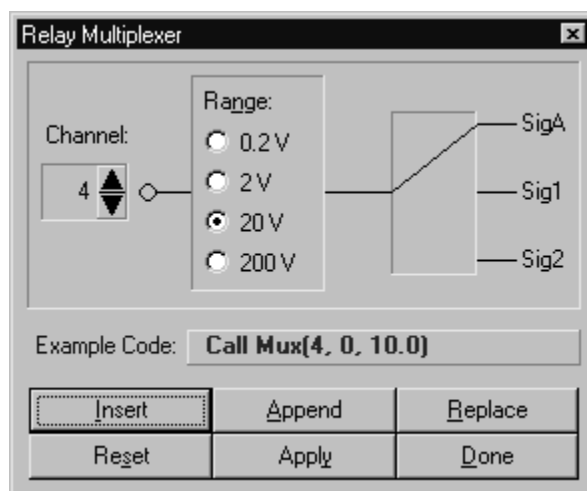
**Consult the Series 2040 Windows Programming Manual
for more information.**



Relay Multiplexer

The test system voltage measurement capability is contained on two circuit boards, the Relay Multiplexer board and the Amplitude Measurement System board. Since all input channels are part of a group of 16:1 multiplexers, only one input channel of each group may be used at one time. To multiplex input channels to Sig1, Sig2, and SigA requires three Mux calls. A single input channel may source Sig1, Sig2, and SigA, but it still

requires 3 individual calls.



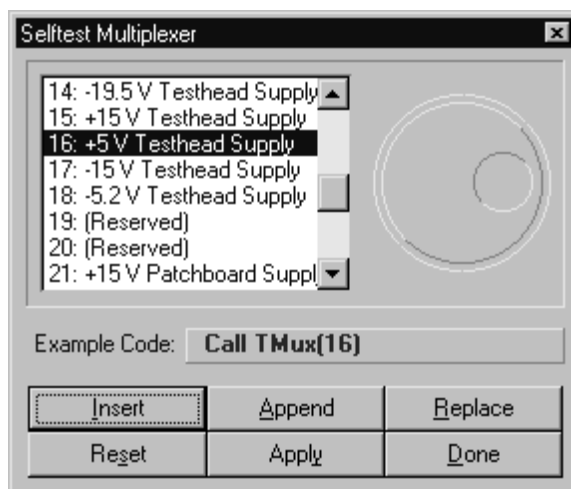
The desired channel can be set with the spin control on the left of the dialog. The Range parameter can be determined by selecting one of the option buttons in the Range frame. By clicking the rectangular box to the right of the “Range” frame, the programmer can toggle the graphic and the functional call between the three signals on the right side of the dialog.

Consult the Series 2040 Windows Programming Manual for more information.



Selftest Multiplexer

The Selftest Multiplexer (TMUX) can be located on any one of four boards, the Instrumentation Amplifier board, Isolation Amplifier board, TMUX utility board, or the Multiple Serial Protocol board. The board containing the TMUX, however, must occupy slot 1. The TMUX provides readback of system signals via Sig3, which is returned to the Amplitude Measurement System board using the analog Motherboard. It is used in calibrating the D/As, ARBs, and AMS using the



TDAC in the Selftest Assembly as a reference. The TDAC is calibrated to a secondary source during the Digalog Certification procedure. The Selftest Multiplexer is available to the programmer and may be used to read back Isolation/Instrumentation Amplifier outputs.

The vertical scroll bar or the jog/shuttle control can be used to scan the list of inputs to be sampled, while the desired channel can be selected with the mouse.

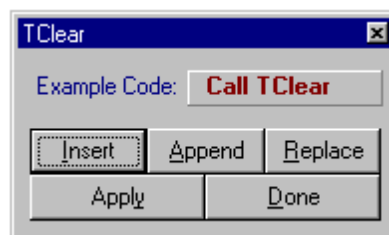
**Consult the Series 2040 Windows Programming Manual
for more information.**



TClear

The TClear dialog is used to reset the Testhead to its original power-up state. It performs the following:

1. Faults the power supply system. This causes all UUT Product Power Supplies to shut down (i.e. their voltage and current outputs are programmed to zero and their relays disconnect them from the Patchboard). In addition, the Patchboard power supplies will turn off and disconnect.
2. Selects the lowest Relay Multiplexer channel in each group multiplexer and sets all to the 200 volt range.
3. Resets all D/As and ARBs to zero volts.
4. Three-states all DIO and ADIO drivers.
5. Opens all AUXRly/AUXFET/High Current Relay/High Current FET channels.
6. Resets the AMS, MDE, and TMS boards.
7. Resets other hardware as needed.



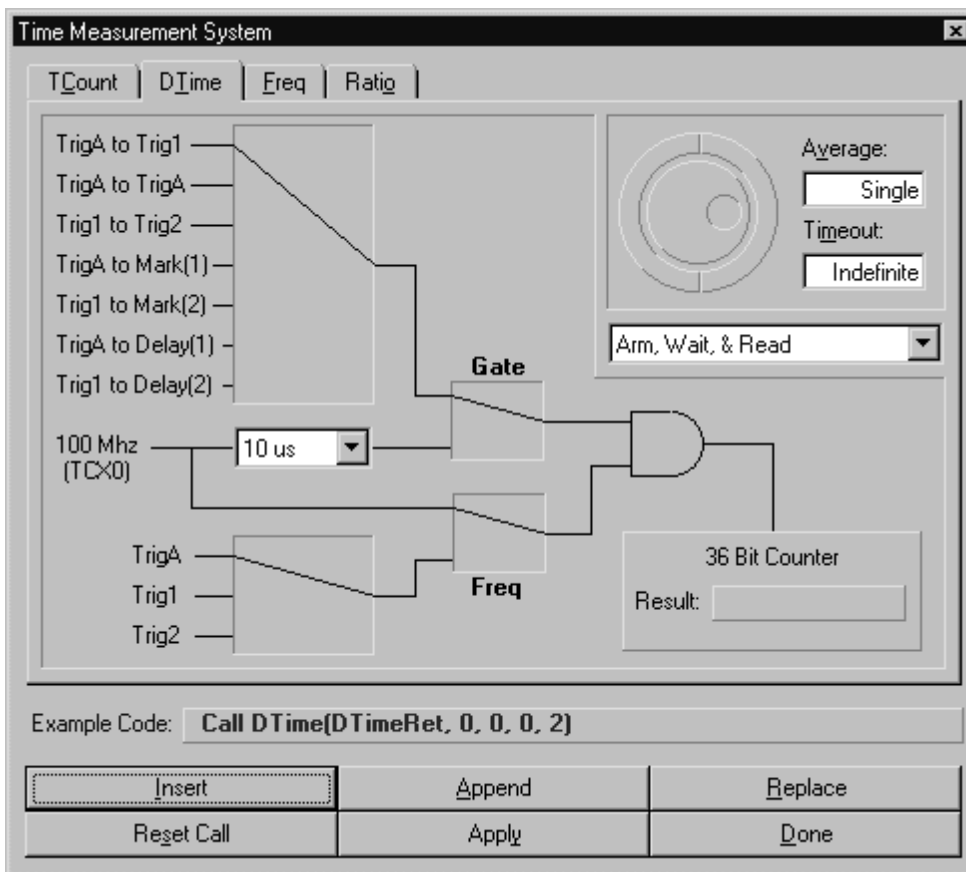
Consult the Series 2040 Windows Programming Manual for more info.



Time Measurement System

The TMS is the system timer/counter. It has the capability of making time (period) and frequency measurements. In addition, the TMS has the ability to count non-periodic events. The TMS gets its input signals from the MDE board. Signal gating and multiplexers on the MDE allow the programmer to select the period to measure, the frequency to measure, or both in the case of the RATIO functional call. Programming of the multiplexers is handled by the DTime and FREQ functional calls. Counting of events can also be accomplished using the TMS event counter and the TCount functional call. Events to count are determined by trigger circuitry on the AMS board. Multiplexers on the MDE select which trigger output to count.

The Time Measurement System dialog covers four different functional calls. When the dialog is opened, tabs for all four functional calls are displayed. The functional call being programmed will have a dotted box around it indicating



that it has focus and is active. In the graphic shown below, The DTime call is active (dotted box around it), and the Example Code textbox shows an example of the DTime functional call. As the programmer switches from call to call, the controls on the dialog will change (appear and/or disappear) to match the functional call selected. Each of the four variations of this dialog will be briefly discussed.

TCount

This functional call is armed by the initial TCount call, and begins counting the selected signal. When the second TCount call is made, the return variable displays the number of occurrences of the selected signal since the previous TCount call, etc. The input signal can be selected from an internal trigger (TrigA, Trig1, or Trig2) or an external trigger by clicking the mouse on the rectangular box adjacent to the trigger textboxes. The return count will be displayed in the grey textbox adjacent to the "Count" label.

DTime

The DTime functional call measures the time between the gates selected by the gate multiplexer in the upper left corner of the dialog. The gate can be selected by clicking the mouse on the rectangular box adjacent to the gate trigger labels to toggle the inputs. Note, the 100 MHz internal clock gives the return variable a resolution of 10 nS. The average and timeout parameters can be selected by clicking the text tool in the desired textbox and using the jog/shuttle control or entering a value manually using the keyboard. The mode parameter uses a listbox for selection. The return time will be displayed in the textbox adjacent to the "Result" label. Also note, the listbox adjacent to the 100 MHz clock input and the trigger inputs in the lower left corner of the dialog will appear ghosted since they are not used in this call.

Freq

The Freq functional call counts an input signal for the timebase selected and returns the frequency in hertz. Note that the gate multiplexer appears ghosted since it is not used in this call. Also note, the gate and freq paths have also changed to match the call. To select the number of readings to average, click the text tool on the textbox adjacent to the "Average" label and use the jog/shuttle control or manually enter a value. The timebase parameter uses a listbox for selection. To select a

trigger, click on the rectangular box adjacent to the trigger labels in the lower left of the dialog.

Ratio

The **Ratio** functional call returns the ratio of the frequency of a waveform to a selected gate time. The gate and signal parameters are selected by clicking the mouse on their adjacent rectangular boxes as before. The average and timeout parameters can be set by clicking the text tool on the desired textbox and using the jog/shuttle control or entering a value manually with the keyboard. The mode listbox appears ghosted since it is not used for this call. Note, if the mouse is clicked on either of the small gate or freq boxes, the signal paths and functional calls change accordingly. The return ratio (dimensionless number) is returned in the grey textbox adjacent to the “Result” label.

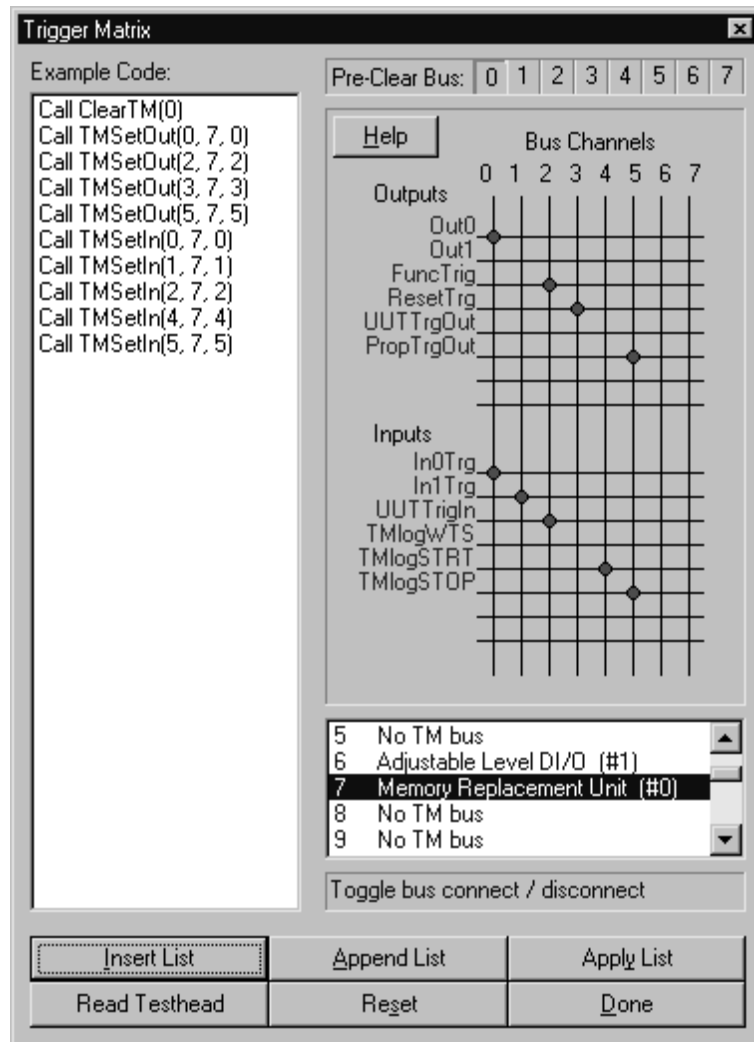
Consult the Series 2040 Windows Programming Manual for more information.



Trigger Matrix

This dialog is used to control the Trigger Matrix circuitry on any board in the Testhead that contains Trigger Matrix resources. When this dialog is opened, the program reads the resource table and displays which board slots in the Testhead contain Trigger Matrix circuitry in a listbox on the lower right corner of the dialog. When one of the slots is selected with the mouse, the appropriate Trigger Matrix functional calls for the board in that slot appear on the dialog as shown on the next page for a MRU board in slot #8.

To “Pre-Clear” the bus, any or all of the boxes in the upper right corner of the dialog can be enabled to clear the individual TMBus lines. At the same time, an appropriate Clear TM functional call for the enabled boxes appears in the Example Code textbox. To program the Input and Output signals, merely click the mouse pointer on the intersection of one of the bus channels and one of the signal channels. A red dot will appear at the junction indicating that the signal was assigned to a bus line. In addition, the appropriate functional call for the signal assignment will be added to the Example Code window. As additional signals are assigned by the dialog, the corresponding functional calls will be added to the code window from the lowest output channel to the highest output channel. The Clear TM functional calls will also be added to the Example Code window from the lowest channel to the highest BEFORE



any of the signal assignment calls are made.

When the “Read Testhead” command button is selected, the program reads the Trigger Matrix configuration from the Testhead and displays the TMBus assignments in the matrix to the right side of the screen. Since this is the current status of the TMBus, no code is displayed in the Example Code window. However, any changes to the current configuration will result in the addition of new code.

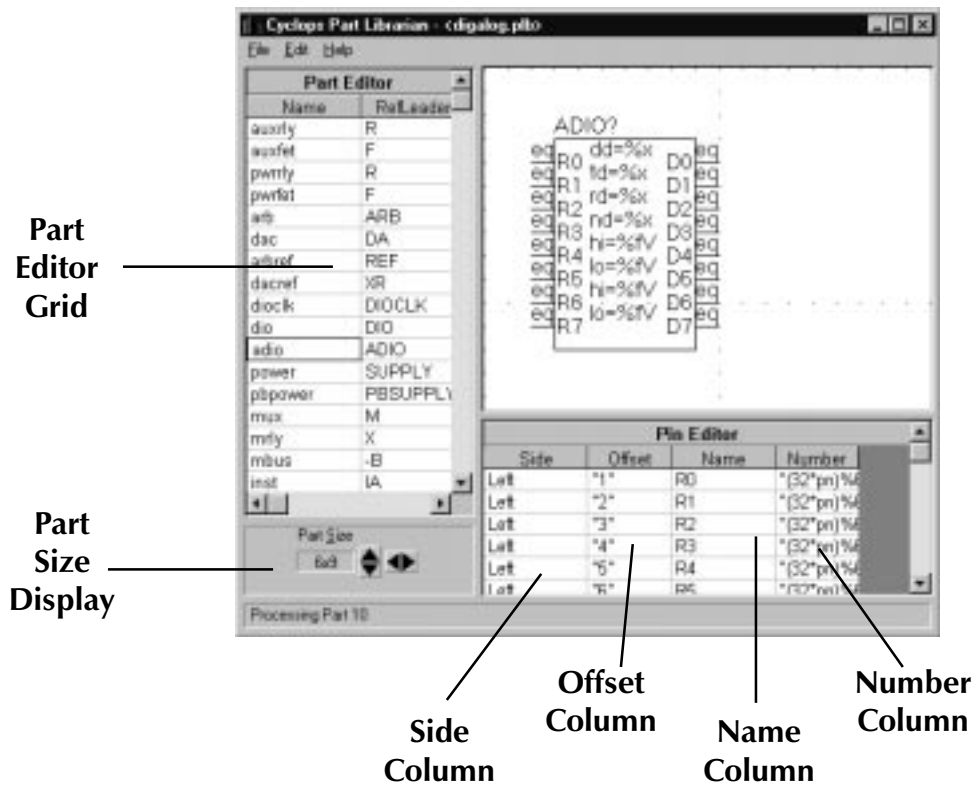
**Consult the Series 2040 Windows Programming Manual
for more information.**

PART EDITOR

The Cyclops Part Editor is used to create and modify part libraries used in the Schematic Capture program specific to the current project. Currently it supports only block parts. Every part library created and stored in the project\PROJECT\Cyclops directory is read into the schematic capture program. Part libraries are normally created for components not in the lib\part.plb file and are specific to the project.

Part Editor Grid

The Part Editor Grid is responsible for maintaining the parts in the library. This is where new parts are created. Parts can be deleted here as well. There are two columns in the grid that display the part name and the part reference leader. The name is the part name displayed in the Schematic Capture part list box. The part reference is displayed and used on the actual schematic. The part reference can have up to 5 alpha characters and cannot have any numbers. Numbers are automatically assigned from the Schematic Capture program. Parts are added and deleted with the Edit Menu.



Pin Editor Grid

The Pin Editor Grid displays information about the pins of the current part. There are four columns: Side, Offset, Name, and Number. Pins are added and deleted with the Edit Menu.

Side Column - The Side Column is used to select the side where the pin is located. Currently only left or right can be selected from the combo box.

Offset Column - The Offset Column is the number of pins spaces from the top of the part. The offset can only be from 0 to the height of the part displayed in the Part Size display.

Name Column - The Name Column is used to display a meaningful name to the part pin. Examples are: D0, D1, RESET. The name is displayed inside the part body adjacent to the pin it references.

Number Column - The Number Column is the pin number. This number must be unique for the current part. For example, there cannot be two pin 5's. This number is displayed just above the pin.

Part Size Display

The Size Display has two spin controls: vertical and horizontal. The vertical control changes the vertical size and the horizontal control changes the horizontal size. The minimum size of a part is 2x2.

Part Display

The Part Display is used to show what the part would look like in the Schematic Capture program. It also displays the part reference with a question mark signifying where the part reference leader and its part reference number is located.

Menu Bar

File Menu

This is a standard Windows File Menu with options for New parts, Open existing parts, Save existing parts, Save As a different name or

path, and Exit.

Edit Menu

The Edit menu contains options for adding or deleting parts and pins.

New Part - The New Part option adds a part after the current part selected in the part editor.

Delete Part - The Delete Part option deletes the current part selected in the part editor.

New Pin - The New Pin option adds a new pin after the current pin selected in the pin editor.

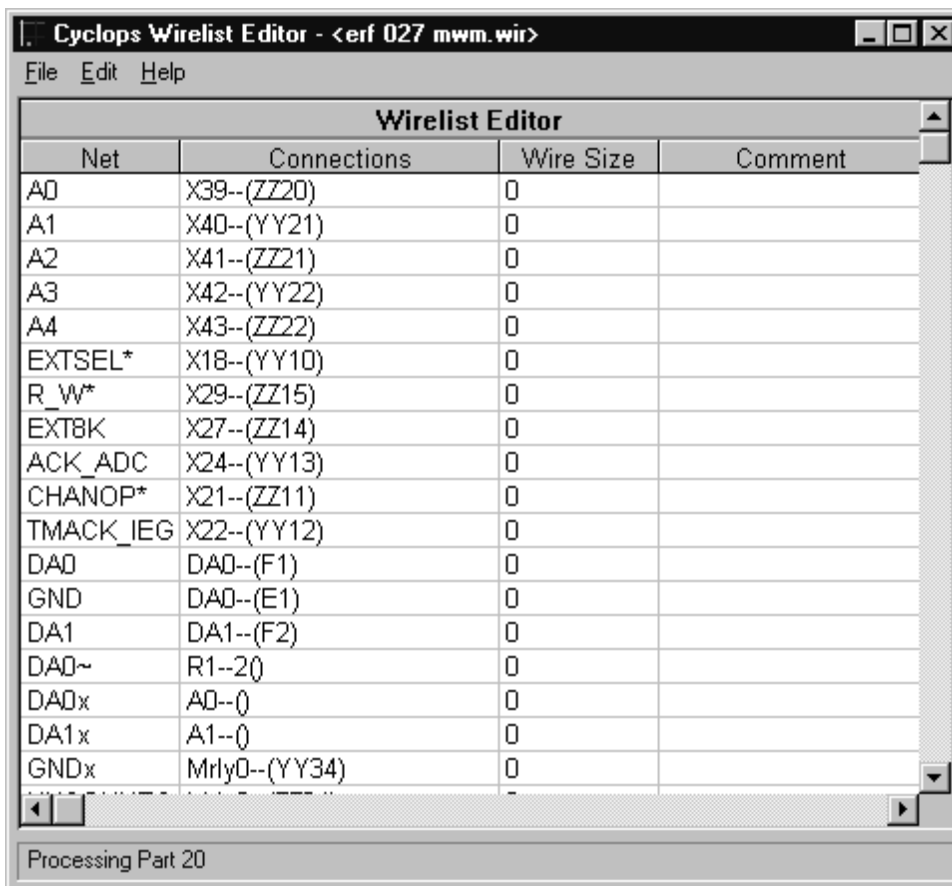
Delete Pin - The Delete Pin option deletes the current pin selected in the pin editor.

Help Menu

The Help menu has entries to start the Help system for the Part Editor. There is also an About screen to display the title and current version of the program.

WIRELIST EDITOR

The Cyclops Wirelist Editor is used to add wire sizes to the wirelist generated from the Schematic Capture program. It contains four columns: Net, Connections, Wire Size, and Comment. Only Wire Size and Comment are changeable. The Net and Connections columns are used for browsing information stored in the wirelist.



Wire Editor Grid

Net Column

The Net Column displays the current Net. This is assigned in the Schematic Capture program automatically or manually. For a manual name, a label must be placed on a wire for that net.

Connections Column

The Connections Column displays a connection to the current net.

Wire Size Column

This column is editable. Any wire size can be entered.

Comments

The Comments Column is used to enter comments for a particular net, usually a description of why the wire size is what it is.

Menu Bar

File Menu

This is a standard File menu with options to Open an existing wirelist, Save a wirelist, Save As another filename or path, and Exit.

Edit Menu

The only option under the Edit menu is Edit Wiresizes. A small dialog will be displayed allowing the programmer to globally change one wire size to another.



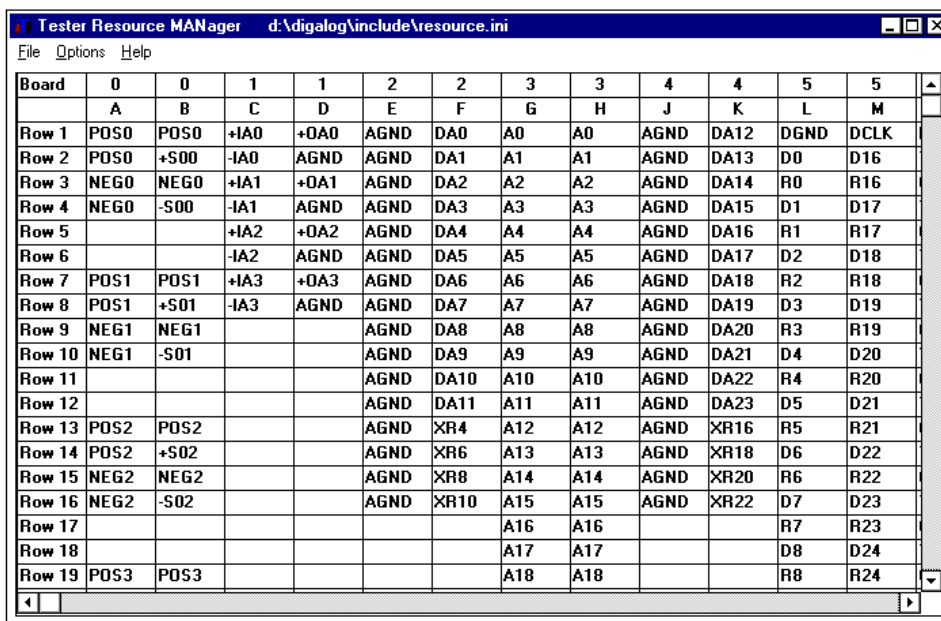
Help Menu

The Help menu has entries to start the Help system for the Wirelist Editor. There is also an About screen to display the title and current version of the program.

TRMAN (Tester Resource Manager)

The Tester Resource Manager is used to track and manage the tester resources including all the boards in the Testhead and the UUT power supplies. Information about these resources can be automatically generated or manually defined, and is used to generate a Patchboard Interface Map and define the pin locations of these resources at the Patchboard. This information can be saved to a project specific file called resource.ini and can be used by other Digalog System's applications.

Specifically, the software is capable of printing out a "Patchboard Map" containing Patchboard pin mnemonics by either automatically interrogating the tester for its resources, or by asking the programmer to define the tester's resources. When the configuration is performed manually, it allows the programmer to configure additional resources beyond what the tester physically contains. In this manner, a programmer has the additional resources to generate programs and fixtures for any tester.

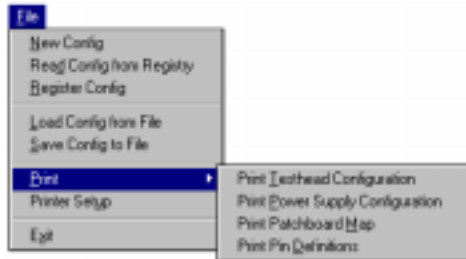


The screenshot shows the 'Tester Resource Manager' application window with the title bar 'd:\digalog\include\resource.ini'. The window contains a table with columns labeled 'Board', '0', '0', '1', '1', '2', '2', '3', '3', '4', '4', '5', '5' and rows labeled 'Row 1' through 'Row 19'. The table contains various resource identifiers and pin locations.

Board	0	0	1	1	2	2	3	3	4	4	5	5
	A	B	C	D	E	F	G	H	J	K	L	M
Row 1	POS0	POS0	+IA0	+OA0	AGND	DA0	A0	A0	AGND	DA12	DGND	DCLK
Row 2	POS0	+S00	-IA0	AGND	AGND	DA1	A1	A1	AGND	DA13	D0	D16
Row 3	NEG0	NEG0	+IA1	+OA1	AGND	DA2	A2	A2	AGND	DA14	R0	R16
Row 4	NEG0	-S00	-IA1	AGND	AGND	DA3	A3	A3	AGND	DA15	D1	D17
Row 5			+IA2	+OA2	AGND	DA4	A4	A4	AGND	DA16	R1	R17
Row 6			-IA2	AGND	AGND	DA5	A5	A5	AGND	DA17	D2	D18
Row 7	POS1	POS1	+IA3	+OA3	AGND	DA6	A6	A6	AGND	DA18	R2	R18
Row 8	POS1	+S01	-IA3	AGND	AGND	DA7	A7	A7	AGND	DA19	D3	D19
Row 9	NEG1	NEG1			AGND	DA8	A8	A8	AGND	DA20	R3	R19
Row 10	NEG1	-S01			AGND	DA9	A9	A9	AGND	DA21	D4	D20
Row 11					AGND	DA10	A10	A10	AGND	DA22	R4	R20
Row 12					AGND	DA11	A11	A11	AGND	DA23	D5	D21
Row 13	POS2	POS2			AGND	XR4	A12	A12	AGND	XR16	R5	R21
Row 14	POS2	+S02			AGND	XR6	A13	A13	AGND	XR18	D6	D22
Row 15	NEG2	NEG2			AGND	XR8	A14	A14	AGND	XR20	R6	R22
Row 16	NEG2	-S02			AGND	XR10	A15	A15	AGND	XR22	D7	D23
Row 17							A16	A16			R7	R23
Row 18							A17	A17			D8	D24
Row 19	POS3	POS3					A18	A18			R8	R24

Menu Bar

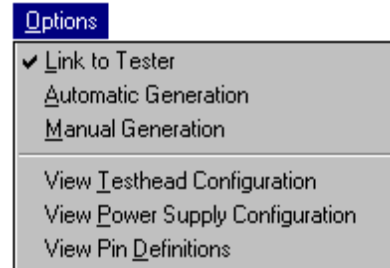
File Menu - The TRMAN **F**ile menu is similar to the standard Windows file



menu. It has selections for a **N**ew configuration, **L**oad an existing configuration, **S**ave a configuration, **S**ave an existing configuration As another file, and **E**xit. The **P**rint Utility as shown below will be discussed later in this section. For the purposes of this explanation, the file being used for illustrations is the standard resource.ini

file in the \Digalog\include\ directory.

Options Menu - When the Options menu is selected, a pull-down menu will be displayed prompting for a choice of the following: **A**utomatic generation, **L**ink to tester, **M**anual Generation, **V**iew pin **D**efinitions, and **V**iew Testhead configuration. Each of these options, along with the **P**rint utility from the **F**ile menu, will be briefly discussed.



Link to Tester - This item will toggle the link between the application and a tester. When checked, the application assumes a Testhead is present and enables Automatic generation. When using Tester Resource Manager on a development computer that is not connected to a Digalog tester, this item should never be checked as it could lead to corruption of certain memory locations.

Automatic Generation - If Automatic Generation is selected, the program scans the tester for its resources, and then updates the map of the Patchboard as shown on the next page. Tester resources required to generate the Patchboard map plus any other tester resources can also be saved to a resource file on the hard drive (resource.ini) stored in the Registry, or printed out.

All programmers writing code for the system should be familiar with this map, since it is the actual physical configuration of the Patchboard.

The screenshot shows the 'Tester Resource Manager' window with a menu bar (File, Options, Help) and a grid of resources. The grid has columns for Board (0-5), Slot (A-M), and various resource types (PDS, NEG, +V, -V, AGND, DA, A, B, C, D, E, F, G, H, J, K, L, M, R, DCLK, D16, R16, D17, R17, D18, R18, D19, R19, D20, R20, D21, R21, D22, R22, D23, R23).

Board	0	0	1	1	2	2	3	3	4	4	5	5
	A	B	C	D	E	F	G	H	J	K	L	M
Row 1	PDS0	PDS0	+V0	+V0	AGND	DA0	A0	A0	AGND	DA12	DGND	DCLK
Row 2	PDS0	+S00	AGND	AGND	AGND	DA1	A1	A1	AGND	DA13	D0	D16
Row 3	NEG0	NEG0	+V1	+V1	AGND	DA2	A2	A2	AGND	DA14	D0	R16
Row 4	NEG0	-S00	AGND	AGND	AGND	DA3	A3	A3	AGND	DA15	D1	D17
Row 5			+V2	+V2	AGND	DA4	A4	A4	AGND	DA16	R1	R17
Row 6			AGND	AGND	AGND	DA5	A5	A5	AGND	DA17	D2	D18
Row 7	PDS1	PDS1	+V3	+V3	AGND	DA6	A6	A6	AGND	DA18	R2	R18
Row 8	PDS1	+S01	AGND	AGND	AGND	DA7	A7	A7	AGND	DA19	D3	D19
Row 9	NEG1	NEG1			AGND	DA8	A8	A8	AGND	DA20	R3	R19
Row 10	NEG1	-S01			AGND	DA9	A9	A9	AGND	DA21	D4	D20
Row 11					AGND	DA10	A10	A10	AGND	DA22	R4	R20
Row 12					AGND	DA11	A11	A11	AGND	DA23	D5	D21
Row 13	PDS2	PDS2			AGND	DA14	A12	A12	AGND	DA16	R5	R21
Row 14	PDS2	+S02			AGND	DA6	A13	A13	AGND	DA18	D6	D22
Row 15	NEG2	NEG2			AGND	DA8	A14	A14	AGND	DA20	R6	R22
Row 16	NEG2	-S02			AGND	DA10	A15	A15	AGND	DA22	D7	D23

Manual Generation - If Manual generation is selected, the screen below will be displayed. Note that the Power Supply Distribution board is always in slot 0 and therefore does not show in the Testhead

The 'Manual Configuration' dialog box is divided into three sections: Product Power Supply Resources, Testhead Resources, and Tester Information.

Product Power Supply Resources: Channel Number (0), Patchboard Connection (0), PPS Type (None, Variable, HPIB, GPIB), Voltage (Hi limit: 55, Lo limit: 0), Current (Hi limit: 10, Lo limit: 0), Serial Loop Position (0-4).

Testhead Resources: Slot Number (Slot 0), Description (Testhead P/S Cont. with Trigger Matrix), Board Number.

	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot
Board Code	1A4	D07	6						
Board Switches	150	4	4						

Tester Information: Tester Serial Number (None), Tester Description (DLINT2).

Buttons: OK, Cancel, Clear All.

configuration table. From this utility, any system configuration may be generated. It is the users responsibility to make sure the generated configuration is valid.

The Programmable variable power supplies (maximum of five) may be defined in terms of maximum voltage and current once a Programmable Power Supply is selected (up/down button). With Volts or Amps selected, the jog shuttle located to the right modifies the selected value. The option buttons under the jog shuttle control determine what position in the serial loop that the supply being defined occupies. However, the variable supplies must be filled in a contiguous manner from #0 to #4. The spin control on the upper left corner of the dialog is used to determine which programming channel is being configured.

If a GPIB or HPIB power supply is being defined or added, the channel and type can be selected and the upper right of the dialog changes to allow the programmer to select a power supply type, a GPIB/HPIB Device number, and what Relay Control board will be used. The spin control on the middle left of the dialog (Patchboard Connection) is used to define which of the five Patchboard connections are connected to which supply, or if a GPIB/HPIB supply will use an external output (i.e. the supply's output does not physically appear on the Patchboard.)

The rest of the Testhead is displayed in table format by slot number. The Description and Board Number drop-down menus are directly linked to, and will modify the table. The Clear All button will clear all configuration items.

Board Codes are specific identifiers for the particular type of board selected for that slot. Board Numbers are used to define the resources of that particular board. For example, if there are two of the same type of board in the system, board numbers zero and one, the resources of board one will be numbered over and above those of board number zero. In other words, for a Relay Multiplexer Board containing sixty-four channels, board zero channels would be labeled zero through sixty-three and board one channels would be labeled sixty-four through one hundred and twenty-seven.

The table is directly linked to the slot drop-down menu, therefore any slot changes in either place will be reflected in the other.

View Testhead Configuration - This utility displays the entire Testhead configuration in an organized format. An example of a typical Testhead configuration is shown below.

Test Head Configuration				
SLOT	Code	Number	Board description	Description of pins
slot 0	104	00	TestHd P/S Conn.	
slot 1	B	00	Instrumentation Amp	Amps 0-3
slot 2	6	00	Analog Source Board	DA 0-11 and ARB 0-1
slot 3	C	00	Auxilliary Relay	Relay 0-31
slot 4	6	01	Analog Source Board	DA 12-23 and ARB 2-3
slot 5	7	00	Digital I/O	Bytes 0-3
slot 6	193	02	Adjustable Level DI/O	Bytes 8-11
slot 7	3	02	Auxilliary FET	FET 64-95
slot 8	C	01	Auxilliary Relay	Relay 32-63
slot 9				
slot 10	2	01	Relay Multiplexor	Mux channels 64-127
slot 11	2	00	Relay Multiplexor	Mux channels 0-63
slot 12	14	00	VI Source	Channel 0-1
slot 13				
slot 14	D08	01	Open Collector I/O	Bytes 8-15
slot 15				
slot 16				
slot 17				
slot 18				
slot 19				
slot 20	5	03	Matrix Relay	Mrlly chan 192-255
slot 21	5	02	Matrix Relay	Mrlly chan 128-191
slot 22	5	01	Matrix Relay	Mrlly chan 64-127
slot 23	5	00	Matrix Relay	Mrlly chan 0-63
slot 24	8	00	Amplitude Measurement	No pin definitions
slot 25	9	00	Time Measurement	No pin definitions
slot 26	A	00	Measurement display	No pin definitions

View Pin Definitions - This utility from the Options menu explains the mnemonics used in the Patchboard map as shown to the right.



View Power Supply Configuration - If this option is selected, a grid is displayed showing all of the power supplies present, along with the type, name, device designation, and Patchboard connection for each supply as shown below.

Unit	Type	Name	Device	Patchboard Connection
PPS 0	Variable	55V - 10A	Serial 0	0
PPS 1	Variable	55V - 10A	Serial 1	1
PPS 2	HPIB	HP6038A (Channel 1)	DEV5	2
PPS 3				
PPS 4				
PPS 5				
PPS 6				
PPS 7				
PPS 8				
PPS 9				
PPS 10				
PPS 11				
PPS 12				
PPS 13				
PPS 14				

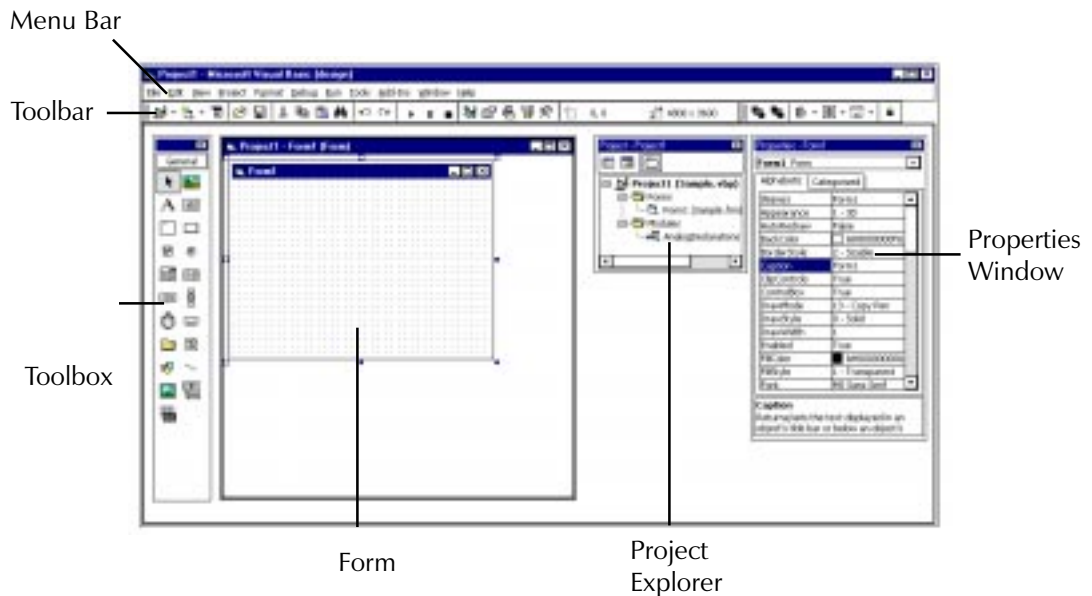
Print Utility - When this utility is selected from the File menu, a small inputbox is displayed prompting for a specific serial number for the printout to be generated. If a specific tester's configuration is to be generated, enter the serial number in the textbox. If not, merely select the OK command button, press <Enter> on the keyboard, or select Cancel. After this inputbox is satisfied, select an individual printout to be generated, or select all three options.

MICROSOFT® VISUAL BASIC®

Microsoft Visual Basic is used as a programming environment for the Series 2040 Test System. It gives the programmer complete flexibility over the test code and the user interface. By combining the simplicity of BASIC programming with graphical design tools, Visual Basic provides the programmer with a quick and easy way to develop and maintain test programs.

VISUAL BASIC FUNDAMENTALS

To open Visual Basic from the Start Button, click on the button, drag the mouse from Programs to the Visual Basic 5.0 Menu, and then to the Visual Basic 5.0 application. Release the mouse button, and the Visual Basic main window will be displayed as shown below.



This window displays all of the main elements of Visual Basic. Each of these will be briefly discussed in the following section. The program also comes with "Visual Basic Books On-Line" which is accessed through the **H**elp pull-down menu from the Menu bar. This option includes all of the normal Visual Basic documentation such as the Visual Basic Programmer's Guide and Language Reference books.

MENU BAR

File Menu

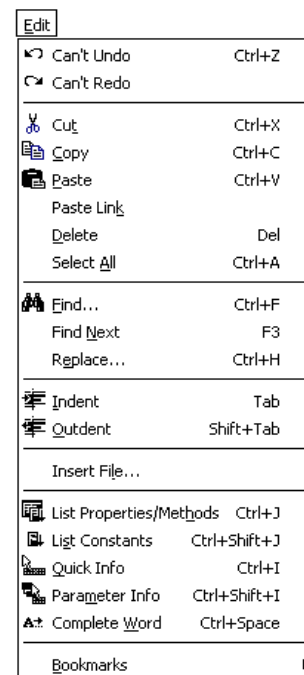
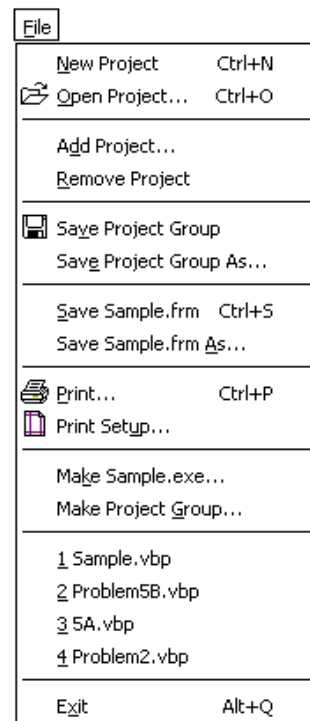
The **F**ile Menu from VB5.0 contains options to create a **N**ew Project, **O**pen an existing Project, **A**dd a Project to the existing Project Group, **R**emove a Project from the existing Project Group, **S**ave a Project Group, **S**ave a Project Group As another name, **S**ave a form, **S**ave a Form **A**s, and the standard Windows NT® Print options. It also contains options for compiling the project into an executable and creating a Project Group from the open projects.

Note: Both the **S**ave Sample.frm and Save Sample.frm **A**s options and the **M**ake Sample.exe option use the current Project or Project Group name, and these names will change as the Project or Project Group name changes.

Edit Menu

The **E**dit Menu from VB5.0 contains the usual **U**nDo, **R**eDo, **C**ut, **C**opy, **P**aste, **D**ele~~t~~e, and **S**elect **A**ll options. The **F**ind, **F**ind **N**ext, and **R**eplace options are used to search for text strings within the project. If the search is successful, the **F**ind dialog disappears and VB5.0 highlights the desired text.

The **I**ndent and **O**utdent options are used to shift a block of highlighted text to the next tab stop to the right (Indent) or the left (Outdent). The **I**nsert **F**ile option is used to insert a file (composed of text) at a location specified by clicking the text tool.



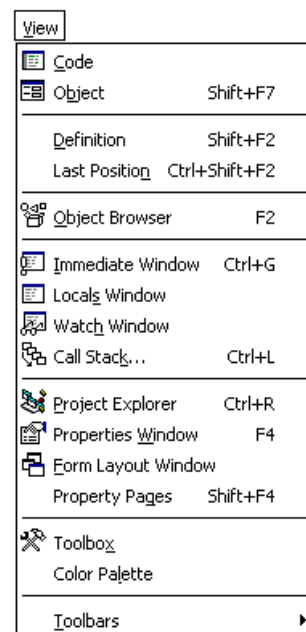
The List Properties/Methods option displays a list box containing the properties and methods available for the object that precedes the period(.). The List Constants option displays a list box containing the valid constants for a selected property. The Quick Info command displays the syntax for a variable, function, statement, method, or procedure which has been selected from the Code window. Parameter Info displays information about the parameters of the initial function or statement. The Complete Word option instructs Visual Basic to complete the remainder of a word once enough letters in the word have been entered for recognition. The last option from the Edit Menu is Bookmarks. This option allows the programmer to create, remove, or manipulate bookmarks in the code window.

View Menu

The View Menu contains options for viewing the various parts of the project. The Code and Object options will display the active object or code behind the active object for the current project. The Definition option displays the location of a selected variable or procedure in the Code window. Last Position allows the programmer to jump to a previous location in the Code window. The Object Browser displays a dialog showing all of the classes available to the project.

The Immediate Window option displays debugging information for troubleshooting purposes. The Locals Window displays the variables in the current stack and their individual values. The Watch window displays the current watch expressions for debugging purposes. The Stack option displays the currently running procedures in the application.

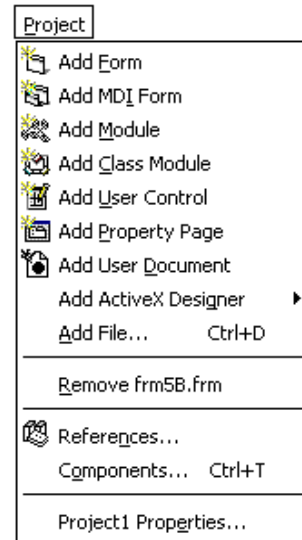
The Project Window, Properties Window, and Form Layout Window options force the display of each of these windows respectively. These windows will be discussed separately later in this section. The Properties Pages option displays a dialog containing a Property or



group of Properties as an alternative to the Properties **W**indow. The **T**oolbox and Color Pa**l**ette options are used to display their respective dialogs. The **T**oolbars option is used to display the various Toolbar groups or design a custom Toolbar.

Project Menu

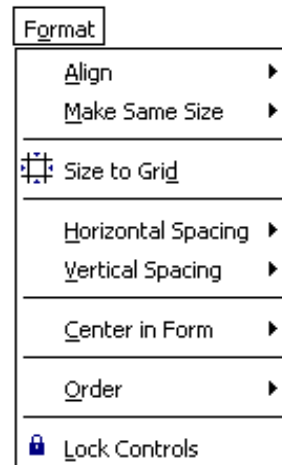
The **P**roject Menu is used to add or remove various options to/from the project. A new or existing **F**orm, **M**DI Form, **M**odule, **C**lass Module, **P**roperty Page, User **D**ocument, Active X **D**esigner, or File can be added. The **A**dd File option is very useful to add a list of *.bas files to the project. These files add declarations and subroutines to the project and eliminate needless code when using Dialog functional calls. The **R**emove Form option allows the programmer to remove the active form from the project.



The **R**eferences option allows the addition of specific libraries to the project while the **C**omponents option allows the addition of controls to the Toolbox. The **P**roject **P**roperties option displays a dialog with General information, Make file information, and Compiler information.

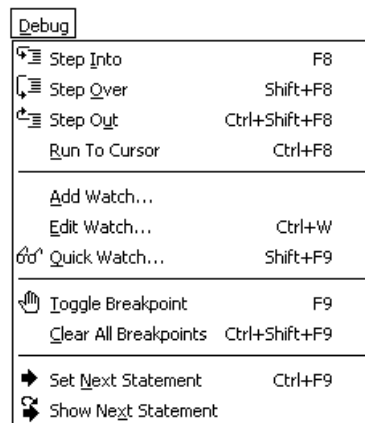
Format Menu

The **F**ormat menu contains options for aligning, sizing, and manipulating graphics, controls, etc. The **O**rder option even allows the programmer to send objects to the back or bring them to the front when overlapping objects are desired. When the objects are set in the desired locations, the **L**ock Controls option can be used to lock the objects in their current positions.



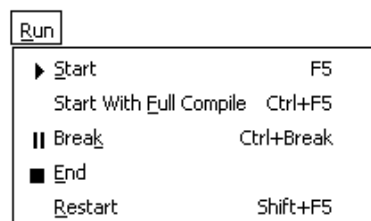
Debug Menu

The Debug Menu allows the programmer to Step **I**nto, Step **O**ver, or Step **O**ut of procedures, **A**dd or **E**dit Watches, and **T**oggle or **C**lear Breakpoints while debugging code. The Set **N**ext Statement can be used to skip or bypass sections of code during the Debug process. The Show **N**ext Statement displays the next statement to be executed.



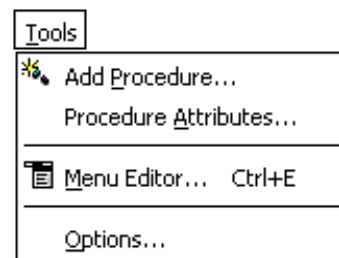
Run Menu

The **R**un menu allows the programmer to **S**tart a program, **S**tart the program after a **F**ull Compile, **B**reak a program, **E**nd a program, and **R**estart a program after it was interrupted for any reason. These operations are usually controlled using the VCR type controls located on the Toolbar.



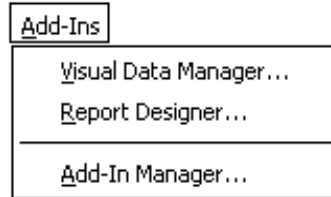
Tools Menu

The Tools menu contains the options to Add a **P**rocedure or display Procedure **A**tttributes. It also contains the **M**enu Editor to generate all of the Menu Bar options and submenu options. In addition, the **O**ptions selection displays a dialog for configuring the default settings for the Visual Basic Development Environment as shown to the right.



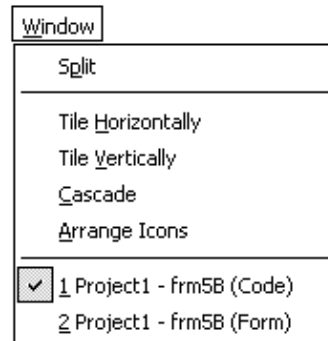
Add-Ins Menu

This menu contains options for invoking the **V**isual Data Manager, the **R**eport Designer, and the **A**dd-In Manager. The Data Manager allows easy manipulation of data in VB50. Report Designer uses Crystal Reports™ to generate reports. The Add-In Manager tool is used to include or delete specific add-ins for the project.



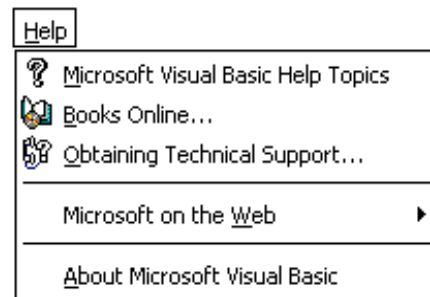
Window Menu

The Split option splits the code window in half horizontally when the window is active. The Tile Horizontally, Tile Vertically, and Cascade options are only available in the MDI mode. Arrange Icons arranges the icons of all minimized windows in the lower left corner of the window. The Window List displays a list of the open windows in the project.



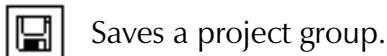
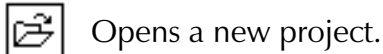
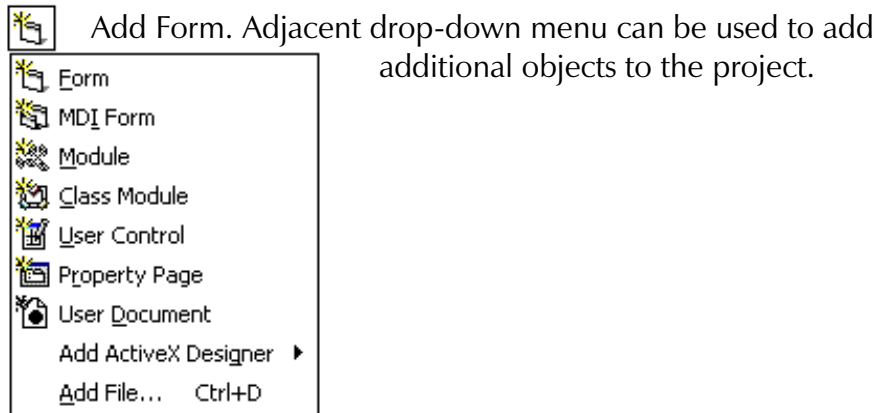
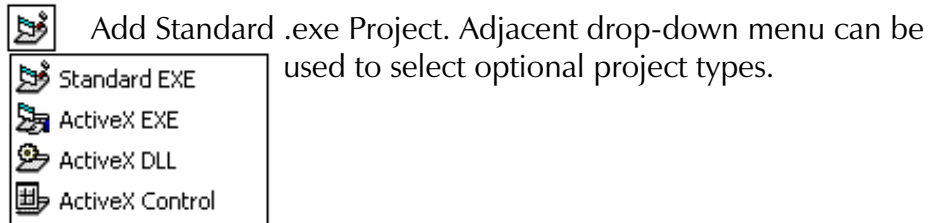
Help Menu


















This menu contains the options for opening the Microsoft Visual Basic Help Topics, Books On-Line, Obtaining Technical Support, Microsoft on the Web which links directly to the Web through the Internet Explorer (if installed), and the normal Windows About window. Books On-Line includes all of the normal documentation that is shipped with Visual Basic V5.0.







TOOLBAR

The buttons displayed on the toolbar correspond to frequently used commands in Visual BASIC, and are intended as a quick method of selecting these commands rather than use the drop-down menus. The commands corresponding to each of the buttons are shown below. Note, all of the tools of the optional toolbars from the View menu are shown.



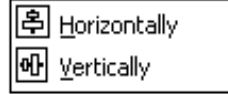
-  Find.
-  Undo.
-  Redo.
-  Start Project
-  Break.
-  End Project.
-  Display the Project Explorer.
-  Display the Properties Window.
-  Display the Form Layout Window.
-  Open the Object Browser.
-  Display the Toolbox.
-  Bring to Front.
-  Send to Back.
-  Align Left. Adjacent drop-down menu can be used to align graphics, controls, etc.
 -  Lefts
 -  Centers
 -  Rights

 -  Tops
 -  Middles
 -  Bottoms

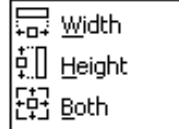
 -  to Grid



Align Horizontally. Adjacent drop-down menu can be used to Align Vertically as well.



Make Height Same Size. Adjacent drop-down menu can be used to Make Width Same Size or both Height and Width.



Lock Controls.



Toggle Breakpoint.



Step Into (Single Step).



Step Over (Procedure Step).



Step Out.



Displays the Locals Window.



Displays the Immediate Window.



Displays the Watch Window.



Quick Watch.



Call Stack.



List Properties/Methods.












List Constants.



Quick Info.



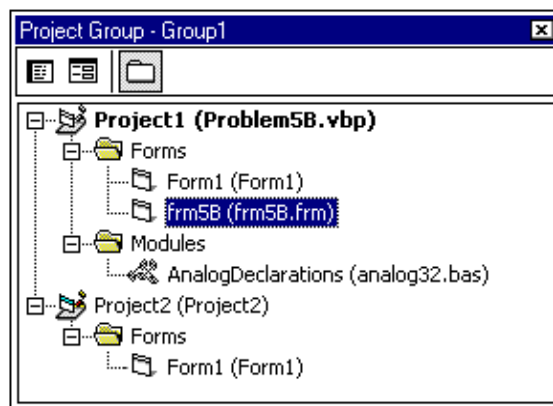
Parameter Info.

-  Complete Word.
-  Indent.
-  Outdent.
-  Comment Block.
-  Uncomment Block.
-  Toggle Bookmark.
-  Next Bookmark.
-  Previous Bookmark.
-  Clear All Bookmarks.

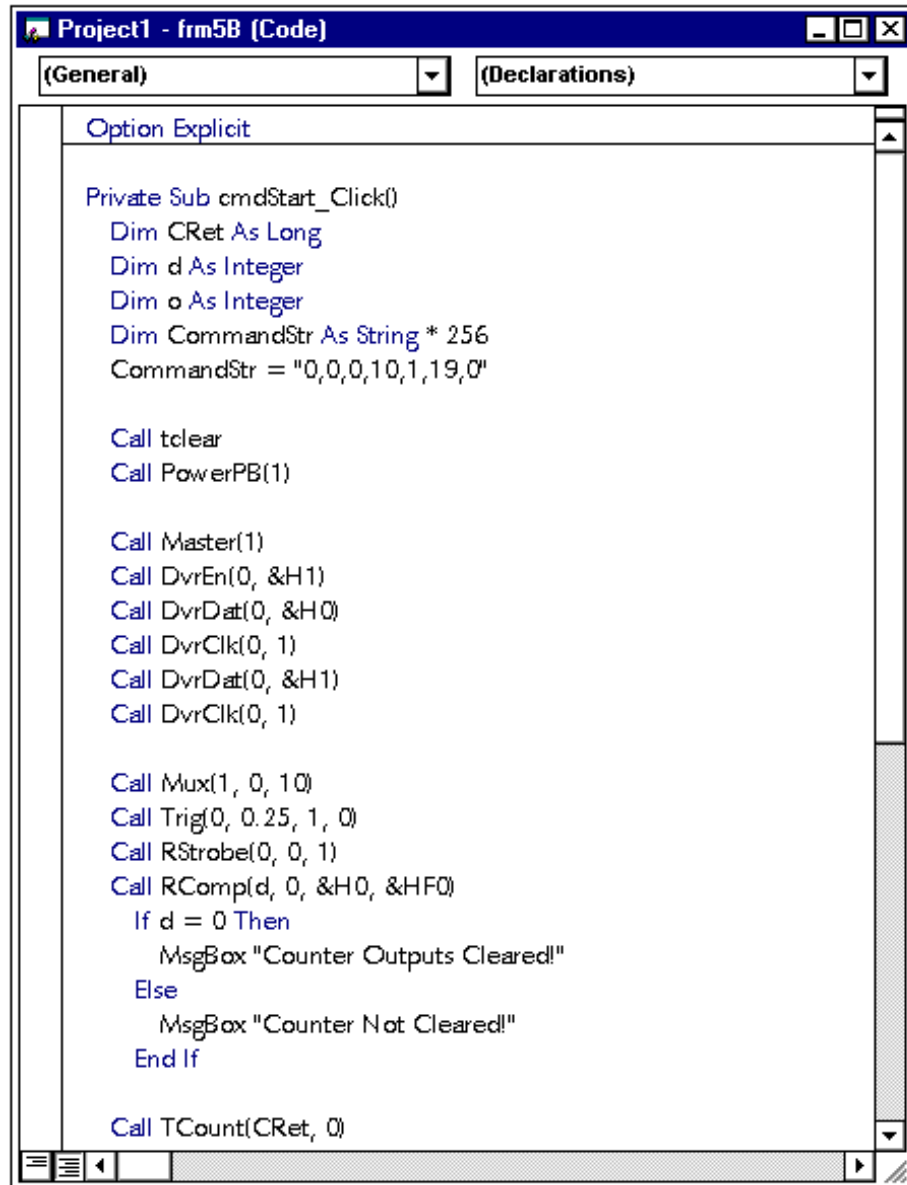
PROJECT WINDOW

Every time a new project is started, a project window for that project is created containing the form, code modules, and custom control files that comprise the new or current project or project group. When Visual Basic is first started, a default project window is generated as shown to the right.

The window also contains two buttons for displaying project related information. The “View Form” button allows you to display the form selected from the project window. If more than one form is associated with the project, use the mouse to highlight the desired form, and select the “View Form” button. The “View Code” button allows the programmer to view the actual programming code



for the selected file. The code appears in a different dialog box, which allows the programmer to add, delete, or modify the code for the selected file. Again, use the mouse to highlight the desired file, then select the "View Code" button. The graphic below shows the code behind frm5B as indicated on the project window.



```
Option Explicit

Private Sub cmdStart_Click()
    Dim CRet As Long
    Dim d As Integer
    Dim o As Integer
    Dim CommandStr As String * 256
    CommandStr = "0,0,0,10,1,19,0"

    Call tclear
    Call PowerPB(1)

    Call Master(1)
    Call DvrEn(0, &H1)
    Call DvrDat(0, &H0)
    Call DvrClk(0, 1)
    Call DvrDat(0, &H1)
    Call DvrClk(0, 1)

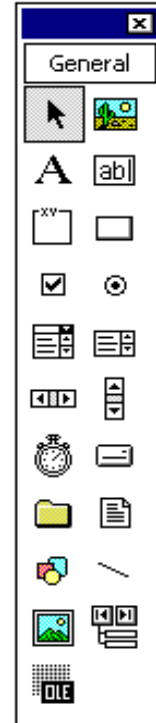
    Call Mux(1, 0, 10)
    Call Trig(0, 0.25, 1, 0)
    Call RStrobe(0, 0, 1)
    Call RComp(d, 0, &H0, &HF0)
    If d = 0 Then
        MsgBox "Counter Outputs Cleared!"
    Else
        MsgBox "Counter Not Cleared!"
    End If

    Call TCount(CRet, 0)
End Sub
```

TOOLBOX

When a new project is opened, Visual Basic creates a blank form which it calls Form 1. This form serves as a scratchpad area for graphics such as scroll bars, file lists, etc.. These graphic “objects” are created on the form using the tools from the Visual Basic Toolbox as shown to the right. For example, if a horizontal scroll bar is required by the application, click on the Horizontal Scroll Bar tool, and a horizontal scroll box appears on Form 1. The scroll bar may be sized by “dragging” the sizing handles on the corners and sides of the graphic using the mouse.

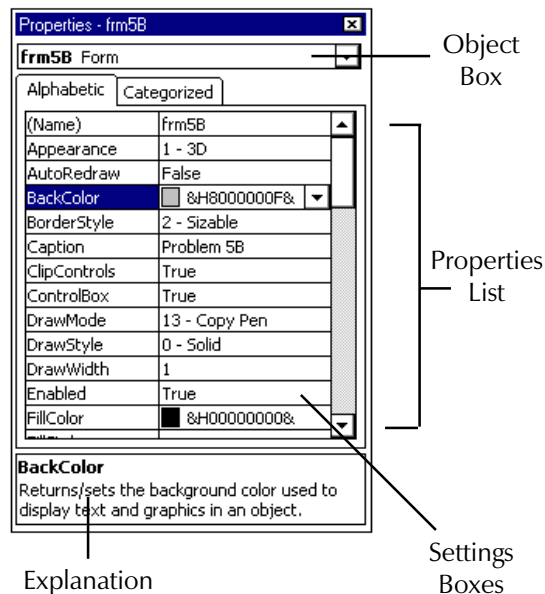
Any and all of the tools shown to the right work in basically the same manner. Each tool can be identified by placing the mouse pointer over the tool, and a small prompt will appear with the name of the tool. The blank space is left for the addition of another tool. The Toolbox may also be expanded to include additional tools from Visual Basic Professional Edition or a third party vendor using the **C**omponents option from the **P**roject menu.



PROPERTIES WINDOW

This window contains all of the properties of any of the objects in the form, or the properties of the form itself. To select an object, click on the down arrow next to the object designation box as shown to the right, and Visual Basic responds with a choice of Form 1 itself or the scroll bar generated in the previous example.

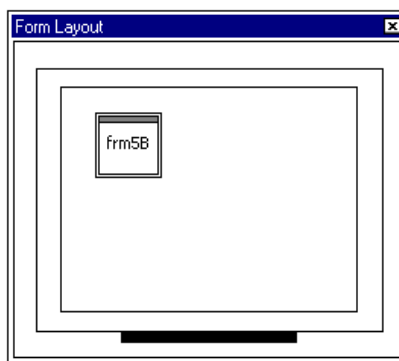
The properties may be arranged in alphabetical order or categorized by clicking on the appropriate folder. To change one of the properties, click on that property in the properties list and the current



selection will appear in the Settings box. When the down arrow next to the Settings box is selected, the available options are displayed in a pull-down window. If a new property is selected, it immediately becomes a property of the selected object, and any changes will be displayed on Form 1 (or whatever form or object was selected). If the property to be changed is text, the Settings box becomes a text entry field. The text box on the bottom of the Properties Window will contain a brief explanation of the property selected. If more information is required, consult the Visual Basic Programmer's Guide or the "Books On-Line" option from the Help menu.

FORM LAYOUT WINDOW

The Form Layout Window shows a representation of the size and location of the active form in the project at run-time, as shown to the right. The size may be changed by dragging the handles of the form itself. The location of the form may be changed by modifying the "StartPosition" property in the Properties Window.



FUNCTIONAL CALLS

In Visual Basic, the Functional Calls are subroutines that talk to the Testhead. Most of the Functional Call subroutines are written in C language. Functional Calls pass data back and forth between your test program and the electronic circuitry in the tester.

Each Functional Call consists of a Call statement, the name of the subroutine, and a list of variables, constants, or expressions called the parameter list. Most of the Functional Calls are used as commands to control the tester by processing and passing the parameters from your program to the Testhead. A few of the Functional Calls are used to take measurements by processing data from the Testhead and sending it back to your program in the parameter list.

To use Functional Calls, you need to know the names of the subroutines (procedures) and the parameters they require. These subroutines are located in libraries of procedures called dynamic-link libraries (DLLs). Since these

libraries are usually not always part of Visual Basic, the procedures must be declared before they can be called. In addition, any variables included in the procedure must be specified. For example, the DA functional call has the following Visual Basic declaration:

Public Sub DA(ByVal Chan As Integer, ByVal Volts As Integer)

The declaration sets the variables Chan and Volts as integers. This Functional Call provides a DC voltage output from the Analog Source Board. When you use this Functional Call in your program, you replace the parameter names Channel and Voltage with the constants, expressions, or variables of your choice to get the desired effect.

If the procedure is written to return a variable, it must be declared as a function as follows:

Declare Function VBPtrd (ErrorCount, TestDescription, DeviceNumber, TestNumber, Address, Expected, Actual) As Integer

Functional calls written as subroutines may also pass back values in their parameter list (i.e. AMS call). In Visual Basic, it is necessary to specify **all** of the parameters for the functional calls.

Since entering all of the necessary declarations can be error prone and time consuming, use the **Add File** option from the **Project** menu to add a file called "analog32.bas" to the Project Box. The file will automatically load whenever the project is opened. For example, whenever using the Digalog functional calls MUX or DA, add "analog32.bas" to the project box. The declarations for the Digalog calls will then be globally loaded into memory and the individual declarations are unnecessary.

cycldata.cls

```
VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
END
Attribute VB_Name = "CyclopsData"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'
' File: cycldata.cls
'
' $Revision: 1.2 $
'
Option Explicit

Private Const Header = "$Header: /u/v/2040/vbincl/cycldata.cls,v 1.2 1997/
10/09 20:05:35 eng12 Exp $"

Public Description As String 'description of test
Public Name As String       'name of current test data (i.e. file name)
Public ResultVar As String  'name of result variable
Public PassOp As Integer    'function to do on pass
Public FailOp As Integer    'function to do on fail
Public Location As Integer  'where to go on fail
Public Message As String    'user tests prompt
Public Condition As Integer 'pass or fail
Public Test As String       'test parameters / conditions
Public Value As Double      'read test value

Public Function Copy() As CyclopsData
    Dim ncd As New CyclopsData
    ncd.Description = Description
    ncd.Name = Name
    ncd.ResultVar = ResultVar
    ncd.PassOp = PassOp
    ncd.FailOp = FailOp
```

```
ncd.Location = Location  
ncd.Message = Message  
ncd.Condition = Condition  
ncd.Test = Test  
ncd.Value = Value  
Set Copy = ncd  
End Function
```

Cyx.bas

```
Attribute VB_Name = "CyxModule"
```

```
,
```

```
' File: cyx.bas
```

```
,
```

```
' $Revision: 1.6 $
```

```
,
```

```
Option Explicit
```

```
Const Header = "$Header: /u/v/2040/vbincl/cyx.bas,v 1.6 1997/07/23  
14:38:34 eng10 Exp $"
```

```
' Pass/Fail Constants
```

```
Public Const IFAIL = 0
```

```
Public Const IPASS = 1
```

```
' Instruction Type Constants
```

```
Public Const ICONT = 0
```

```
Public Const IGOTO = ICONT + 1
```

```
Public Const ISTOP = IGOTO + 1
```

```
Public Const IWAIT = ISTOP + 1
```

```
Public Const OPMASK = &HFFFE
```

```
Public RootObj As New Root
```

```
Private Sub CVSInfo()
```

```
    Dim s As String
```

```
    s = Header
```

```
End Sub
```

```
Public Sub InitializeCYX()
```

```
    RootObj.Initialized = True
```

```
End Sub
```

```
Public Function ExecuteCYX() As Integer
```

```
    ExecuteCYX = RootObj.sequence()
```

```
End Function
```

```
Public Function DoubleToEngNotation(d As Double) As String
  Const EngFollower = "yzafpnum kMGTPEZY"
  Dim x As Integer
  Dim z As Integer
  Dim a As Double
  Dim y As Double
  Dim s As String
  Dim f As String

  x = 0
  a = Abs(d)
  f = "##0.000"
  If a > 1E-24 Then
    x = Int(Log(a) / Log(10) + 0.00001) + 24
    y = d / (10 ^ ((x - (x Mod 3)) - 24))
    If Abs(y) < 0.001 Or x < 0 Or (x \ 3) > 16 Then
      s = "0.000"
    Else
      z = Int(Log(Abs(y)) / Log(10))
      If z < 0 Then z = 0
      s = Format(y, Mid(f, 3 - z, 5) & Mid(EngFollower, x \ 3 + 1, 1))
    End If
  Else
    s = "0.000"
  End If

  DoubleToEngNotation = s

End Function
```

Cyresult.bas

```
Attribute VB_Name = "CyclopsResults"
```

```
,
```

```
' File: cyresult.bas
```

```
,
```

```
' $Revision: 1.4 $
```

```
,
```

```
Option Explicit
```

```
Private Const Header = "$Header: /u/v/2040/vbincl/cyx/cyresult.bas,v 1.4  
1998/01/22 20:31:38 eng12 Exp $"
```

```
Private Sub CVSInfo()
```

```
    Dim s As String
```

```
    s = Header
```

```
End Sub
```

```
Public Sub Main()
```

```
    Executive.Show
```

```
End Sub
```

```
Public Sub BeforeCyxTests(cyxd As CyxData)
```

```
    Dim s As String
```

```
    s = App.EXENAME & " Date: " & Date & " Time: " & Time
```

```
    Executive.LogHeader s
```

```
End Sub
```

```
Public Sub AfterCyxTests(cyxd As CyxData)
```

```
End Sub
```

```
Public Sub BeforeCyclopsTests(cyxd As CyxData)
```

```
End Sub
```

```
Public Sub AfterCyclopsTests(cyxd As CyxData)

End Sub
Public Sub BeforeSequence(cd As CyclopsData)

End Sub

Public Sub AfterSequence(cd As CyclopsData)
  Dim s As String

  s = cd.Description & ": " & cd.Test & " Act=" & cd.Value & " " & _
    If(cd.Condition = IPASS, "Pass", "Fail")

  Executive.Log s, cd.Condition

  DoEvents
  If Executive.Interrupted = True Then
    cd.PassOp = ISTOP
    cd.FailOp = ISTOP
  End If

End Sub
```

Glossary Of Terms

.arb - File extension for an ARB waveform.

.arp - File extension for an ArbPulse waveform.

API (Application Programmer's Interface) - Collection of Dynamic Link Libraries (DLLs) which Windows applications use to display windows, graphics, manage memory, etc.

.bas File - Standard modules (.bas filename extension) contain public or module level declarations of types, constants, variables, external and public procedures.

Class - A class module is similar to a Form module but has no visible user interface. A Class module can be used to create objects, and can contain code for defining the object.

.cls - File extension for a class module.

Dialog - A custom form containing command buttons, option buttons, text boxes, etc. that allow the user to supply information to the application.

DLL (Dynamic Link Libraries) - Libraries of procedures that applications can link to at "run time" rather than be included in compiled code.

DUT - Device Under Test - The device or unit being tested. (Same as UUT)

Environment Variable - This statement in the Autoexec.bat file that sets the path to the Digalog directory; usually set to c:\Digalog.

Fixture - Hardware interface between the UUT (Unit Under Test) or DUT (Device Under Test) and the Patchboard.

Form Module - Text description of a form and its properties.

.frm - File extension for Visual Basic forms.

.frx - File extension for Visual Basic form "stash" file (binary).

Functional Call - Digalog subroutines written in the C programming language for the purpose of communicating with and controlling the Testhead circuitry.

GPIB - General Purpose Interface Bus - A communication bus using the IEEE-488 standard.

Iteration - A Digalog software technique used to determine the best test parameters to use for a specific part for the best standard deviation.

MDA - Manufacturing Defects Analyzers

.mde - File extension for a MDE waveform file.

Net - A junction of two or more components in a circuit. Some test schemes also designate a net at an input or output to/from a complete circuit for testing purposes.

.plb - The .plb file extension is used for the part libraries for the Schematic Capture program.

Patchboard - Digalog interface from the Testhead circuitry to the user's fixture.

.sch - The .sch file extension is used for schematic files for the Schematic Capture program.

.tcl - The .tcl file extension is associated with the test classes generated by the Test Manager containing all the code and information necessary to conduct tests and evaluate the results.

Testhead - A card cage containing all of the test circuitry in the tester including a direct hardware interface to the Patchboard. No power supplies are included in the Testhead.

UUT - Unit Under Test - The unit or device being tested. (Same as DUT)

.vbp - File extension for a Visual Basic project file.

.wir - An output file from the Schematic Capture program used as a wirelist of a generated schematic..