

PLXMon98 User's Manual

Release 1.0, initial publishing February 2, 1998.

Copyright © 1998, PLX Technology, Inc.. All rights reserved.

This document contains proprietary and confidential information of PLX Technology Inc. (PLX). The contents of this document may not be copied nor duplicated in any form, in whole or in part, without prior written consent from PLX.

PLX provides the information and data included in this document for your benefit, but it is not possible for us to entirely verify and test all of this information in all circumstances, particularly information relating to non-PLX manufactured products. PLX makes no warranties or representations relating to the quality, content or adequacy of this information. Every effort has been made to ensure the accuracy of this manual, however, PLX assumes no responsibility for any errors or omissions in this document. PLX shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or the examples herein. PLX assumes no responsibility for any damage or loss resulting from the use of this manual; for any loss or claims by third parties which may arise through the use of this software; for any loss or claims by third parties which may arise through the use of this software; and for any damage or loss caused by deletion of data as a result of malfunction or repair. The information in this document is subject to change without notice.

Product and Company names are trademarks or registered trademarks of their respective owners.

Document number: plxmon98.doc

Table of Contents

1. GENERAL INFORMATION	1-1
1.1 About This Manual.....	1-1
1.2 Conventions and Terminology.....	1-1
2. WHAT IS PLXMon98	2-1
2.1 Getting PLXMon98 Started.....	2-1
2.1.1 Starting PLXMon98.....	2-2
2.2 The PLXMon98 Interface.....	2-2
2.2.1 The PLXMon98 ToolBar.....	2-3
2.2.2 The Command Line Interface (CLI).....	2-3
2.2.3 Status Bar.....	2-5
2.3 Selecting Device.....	2-5
2.4 Launching PlxLdr97 Application.....	2-5
2.5 Reset Embedded CPU.....	2-6
2.6 Interactive DMA.....	2-6
3. ACCESSING PCI 9080 REGISTERS	3-1
3.1 The PCI 9080 Register Dialog Box.....	3-1
3.2 The Register Group Dialog Boxes.....	3-1
3.2.1 PCI Configuration Register Group Dialog Box.....	3-1
3.2.2 Local Configuration Register Group Dialog Box.....	3-2
3.2.2.1 The Mode/Arbitration Dialog Box.....	3-3
3.2.2.2 Endian Descriptor Dialog Box.....	3-3
3.2.2.3 The Space 0/Exp ROM Dialog Box.....	3-4
3.2.2.4 The DM PCI Remap Dialog Box.....	3-4
3.2.2.5 The DM Config IO Addr Dialog Box.....	3-5
3.2.2.6 The Space 1 Dialog Box.....	3-5
3.2.3 The Runtime Register Group Dialog Box.....	3-6
3.2.3.1 The Interrupt Control/Status Register Dialog Box.....	3-6
3.2.3.2 The EEPROM, PCI, User IO Dialog Box.....	3-7
3.2.4 The DMA Register Group Dialog Box.....	3-7
3.2.4.1 The DMA Mode Dialog Box.....	3-8
3.2.4.2 The Descriptor Pointer Dialog Box.....	3-8
3.2.4.3 The DMA Channels Threshold Dialog Box.....	3-9
3.2.5 The Messaging FIFO Register Group Dialog Box.....	3-9
3.2.5.1 The FIFO Status/Control Register Dialog Box.....	3-9
3.3 The Serial EEPROM Dialog box.....	3-10

4. PLXMon98 BASICS	4-1
4.1 Command Line	4-1
4.2 Scrolling and Editing Command Lines.....	4-2
4.3 User-Variables	4-2
4.4 Expressions.....	4-2
4.4.1 Monadic Operators.....	4-3
4.4.2 Dyadic Operators	4-3
4.5 Macros	4-3
4.6 Ranges	4-4
4.7 Repeating Command Lines (Looping)	4-4
4.8 On-line Help	4-4
4.9 PLXMon98 Batch Files	4-4
4.9.1 The read command.....	4-4
4.9.2 Documenting your file	4-5
4.10 PLXMon98 Command Set.....	4-5
4.10.1 Memory Manipulation	4-6
4.10.1.1 Display: d, db, dw, dl, dd	4-6
4.10.1.2 Enter: e, eb, ew, el, ed	4-7
4.10.1.3 Move: m, mb, mw, ml, md.....	4-8
4.10.1.4 Compare: c	4-8
4.10.1.5 Search: s	4-8
4.10.2 Input/Output.....	4-9
4.10.2.1 Input: i, ib, iw, il, id.....	4-9
4.10.2.2 Output: ob, ow, ol, od.....	4-10
4.10.3 PCI Commands	4-10
4.10.3.1 Show or Select a Device: dev.....	4-10
4.10.4 PLX Device Commands.....	4-11
4.10.4.1 PCI Configuration Registers: pcr	4-11
4.10.4.2 Local Configuration Registers: lcr	4-11
4.10.4.3 Runtime Registers: rtr	4-11
4.10.4.4 Local DMA Registers: ldr	4-12
4.10.4.5 Messaging Unit Registers: mqr	4-12
4.10.4.6 Interactive DMA programming: idma.....	4-12
4.10.4.7 DMA Programming: dma.....	4-13
4.10.4.8 Read Serial EEPROM: re.....	4-13
4.10.4.9 Write Serial EEPROM: we.....	4-14
4.10.5 PLXMon98 Internal Commands	4-15
4.10.5.1 Show Variables: vars.....	4-15
4.10.5.2 Expressions: expr	4-15

4.10.5.3	Define a Macro: define.....	4-15
4.10.5.4	Show Macros: macs	4-15
4.10.5.5	Repeat: r	4-16
4.10.5.6	echo	4-16
4.10.5.7	wait.....	4-16
4.10.5.8	base.....	4-17
4.10.5.9	save.....	4-17
4.10.5.10	read.....	4-17
4.10.5.11	range	4-18
4.10.5.12	cmd_edit.....	4-18
4.10.5.13	help, h, ?	4-18

5. CUSTOMER SUPPORT

5-1

List of Figures

Figure 1 The PLXMon98 GUI.....	2-3
Figure 2 The PLXMon98 ToolBar	2-3
Figure 3 The Command Line Interface.....	2-4
Figure 4 Save Macro Dialog Box	2-4
Figure 5 The Select A PCI Device dialog box.....	2-5
Figure 6 The Download to Embedded dialog box	2-6
Figure 7 Reset Embedded CPU response box	2-6
Figure 8 Interactive DMA dialog box.....	2-7
Figure 9 PCI Configuration Registers dialog box.....	3-1
Figure 10 Local Configuration Registers dialog box.....	3-2
Figure 11 Mode/Arbitration dialog box.....	3-3
Figure 12 Endian Descriptor dialog box.....	3-3
Figure 13 Space 0/Exp ROM dialog box.....	3-4
Figure 14 The DM PCI Remap Details dialog box.....	3-4
Figure 15 DM Config IO Addr dialog box	3-5
Figure 16 Space 1 dialog box	3-5
Figure 17 Runtime Registers dialog box	3-6
Figure 18 Interrupt Ctrl Details dialog box	3-6
Figure 19 EEPROM, PCI, User IO Details dialog box.....	3-7
Figure 20 DMA Registers dialog box.....	3-7
Figure 21 DMA Mode dialog box	3-8
Figure 22 Descriptor Pointer dialog box.....	3-8
Figure 23 Threshold dialog box.....	3-9
Figure 24 Messaging Unit Registers dialog box.....	3-9
Figure 25 FIFO Status/Control Register dialog box	3-10
Figure 26 Configuration Serial EEPROM dialog box	3-10

List of Tables

Table 1 Monadic Operations (where N refers to a number)	4-3
Table 2 Dyadic Operators (where N and M refer to numbers and variables).....	4-3
Table 3 Index of Commands.....	4-6



PLX SOFTWARE LICENSE AGREEMENT

THIS SOFTWARE DESIGN KIT INCLUDES PLX SOFTWARE THAT IS LICENSED TO YOU UNDER SPECIFIC TERMS AND CONDITIONS. CAREFULLY READ THE TERMS AND CONDITIONS PRIOR TO USING THIS DESIGN KIT. BY OPENING THIS PACKAGE OR INITIAL USE OF THIS SOFTWARE DESIGN KIT INDICATES YOUR ACCEPTANCE OF THE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD RETURN THE ENTIRE SOFTWARE DESIGN KIT TO PLX.

LICENSE Copyright (c) 1998 PLX Technology, Inc.

This PLX Software License agreement is a legal agreement between you and PLX Technology, Inc. for the PLX Software Design Kit ("SOFTWARE PRODUCT") which is provided on the enclosed PLX diskettes, or may be recorded on other media included in this Software Design Kit. PLX Technology owns this SOFTWARE PRODUCT. The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties, and is licensed, not sold. If you are a rightful possessor of the Software Design Kit, PLX grants you a license to use the SOFTWARE PRODUCT as part of or in conjunction with a PLX chip on a per project basis. PLX grants this permission provided that the above copyright notice appears in all copies and derivatives of the SOFTWARE PRODUCT. Use of any supplied runtime object modules or derivatives from the included source code in any product without a PLX Technology, Inc. chip is strictly prohibited. You obtain no rights other than those granted to you under this license. You may copy the SOFTWARE PRODUCT for backup or archival purposes. You are not authorized to use, merge, copy, display, adapt, modify, execute, distribute or transfer, reverse assemble, reverse compile, decode, or translate the SOFTWARE PRODUCT except to the extent permitted by law.

GENERAL

If you do not agree to the terms and conditions of this PLX Software License Agreement, do not install or use the Software Design Kit and promptly return the entire unused SOFTWARE PRODUCT to PLX Technology, Inc. You may terminate your license at any time. PLX Technology may terminate your license if you fail to comply with the terms and conditions of this License Agreement. In either event, you must destroy all your copies of this SOFTWARE PRODUCT. Any attempt to sub-license, rent, lease, assign or to transfer the Software Design Kit except as expressly provided by this license, is hereby rendered null and void.

WARRANTY

PLX Technology, Inc. provides this SOFTWARE PRODUCT AS IS, WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. PLX makes no guarantee or representations regarding the use of, or the results based on the use of the software and documentation in terms of correctness, or otherwise; and that you rely on the software, documentation, and results solely at your own risk. In no event shall PLX be liable for any loss of use, loss of business, loss of profits, incidental, special or, consequential damages of any kind. In no event shall PLX's total liability exceed the sum paid to PLX for the product licensed hereunder.

1. General Information

PLXMon98 is a powerful Windows application designed to easily monitor PCI 9080 devices.

PLXMon98 works well with:

- The IBM 401 PPC RDK (PCI 9080RDK-401);
- The Intel i960 RDK (PCI 9080RDK-960);
- The Motorola MPC860 RDK (PCI 9080RDK-860); or,
- Any device that uses the PCI 9080 IC.

Although PLXMon98 can interact with any PCI device, it is specifically designed to work best with the PLX PCI 9080 IC. The operation of PLXMon98 with other PCI bridge chips can be unpredictable, therefore, due caution should be taken when using PLXMon98 with other PCI bridge chips.

PLXMon98 has been tested to ensure compatibility with both Windows NT and Windows 95.

1.1 About This Manual

This manual provides information about the functionality of PLXMon98. The following is a brief summary of the chapters to help guide your reading of this manual:

Chapter 2, What is PLXMon98, describes the functionality of PLXMon98.

Chapter 5, Customer Support, provides customer support contact information.

1.2 Conventions and Terminology

All references to Windows NT assume Windows NT 4.0 or higher and will be shown as WinNT. Similarly, references to Windows 95 will be shown as Win95.

All references to IOP (I/O Platform) throughout this manual refer to the embedded hardware and all references to IOP software refer to the embedded software.

All values used in the manual are hexadecimal numbers, with the exception for memory sizes (used in Local Configuration Register screen). The prefix '0x' has been omitted from all hexadecimal numbers and is not required when entering values for PLXMon98 fields.

Note: PLXMon98 is designed to run best with a high-resolution display, such as 1024x768 with 256 colors or better. Users choosing to run the software at lower resolutions will find it visually undesirable.

2. What is PLXMon98

PLXMon98 is a powerful Windows-based program designed for engineers working with the PCI bus, and the PLX family of PCI devices.

Some of the commands that are available to PLX devices and non-PLX devices include:

- The ability to select any PCI device on any PCI bus;
- To examine and modify that device's PCI Configuration Registers; and,
- To display, modify, copy, fill, compare and search memory using the programmed PCI Base Address Registers of the PCI device (if available) using either memory or I/O cycles on the PCI bus.

Other commands available to PLX devices include the ability to:

- Examine and modify the Local Configuration Registers;
- Examine and modify the Runtime Registers;
- Examine and modify the Local DMA Registers;
- Examine and modify the Messaging Unit Registers;
- Program and control chained and non-chained DMA transfers; and,
- Read from and write to the Serial EEPROM connected to the PCI 9080.

The PCI devices can be accessed in PLXMon98 either by using the various Graphical User Interface (GUI) windows (see Figure 1 to Figure 26) or by using the Command Line Interface (CLI). The CLI is similar to the one used with PLXMon97 and supports many of the same commands. The PLXMon98 commands available are generally short and can be chained together on a single command line. The settings can also be edited directly using the various GUI (Graphical User Interface) screens.

User-defined macros, variables and an extensive set of expression evaluation operators are supported through the CLI. With the repeat command, a command can be run infinitely or for a specific number of iterations. Sections of a command line can also be repeated within a command line including sections within sections.

2.1 Getting PLXMon98 Started

PLXMon98 can be started by:

- Typing PLXMon98 at the command prompt and press Enter.
- Clicking on the icon in the PCI SDK 1.2 folder in the Start Menu.
- Click on the PLX icon located on the Windows taskbar.

See section 2.1.1 for startup information.



2.1.1 Starting PLXMon98

When PLXMon98 starts, it:

1. Initializes PLX API, which provides a handle to the device driver. When the PLX API fails to connect to the device driver, a message box pops up notifying of the cause the failure, and terminates PLXMon98.
2. Locates all PCI devices on all PCI buses. When no PCI devices are found a message box is displayed.
3. Selects a PLX PCI device by searching the registered devices, and selecting the first one found. PLXMon98 sets the user-variables `s0`, `s1`, `s2`, and `s3` to the values found in the selected device's PCI Configuration Registers `0x18`, `0x1c`, `0x20`, and `0x24` respectively. For PLX devices, these variables refer to PCI base addresses for the device's local address spaces.
4. Allocates a 64KB buffer for data transfers, and sets a variable called `hbuf` that refers to it.
5. Sets a variable called `regbase`, which corresponds to the memory-mapped address of the Local Configuration Registers on the IOP bus. This value should be changed to correspond to the appropriate IOP bus address of the Local Configuration Registers for other non-PLX boards using the PCI 9080.
6. Sets a user-variable called `membase`, which corresponds to an address in the IOP memory available to load chained DMA descriptors. This value should be changed to correspond to the appropriate address for DMA descriptors in IOP memory for other non-PLX boards using the PCI 9080.
7. Displays the PCI Configuration screen for the first PLX PCI device found. When no PLX device is present in the system, a message window is prompted stating that no PLX devices are present and the PCI Configuration screen is not displayed.

2.2 The PLXMon98 Interface

The PLXMon98's main interface, shown in Figure 1, contains:

- (A) A drop-down menu bar, with the four main drop-down menus;
- (B) An optional toolbar (for a full view, see Figure 2);
- (C) The `load macro` and `save macro` buttons to load and save macros;
- (D) The CLI where PLXMon97 commands can be entered;
- (E) The optional status bar;
- (F) A pull down menu that displays the CLI command history; and,
- (G) A toggle button to toggle the CLI window between the top of the command window or the bottom of the command window.

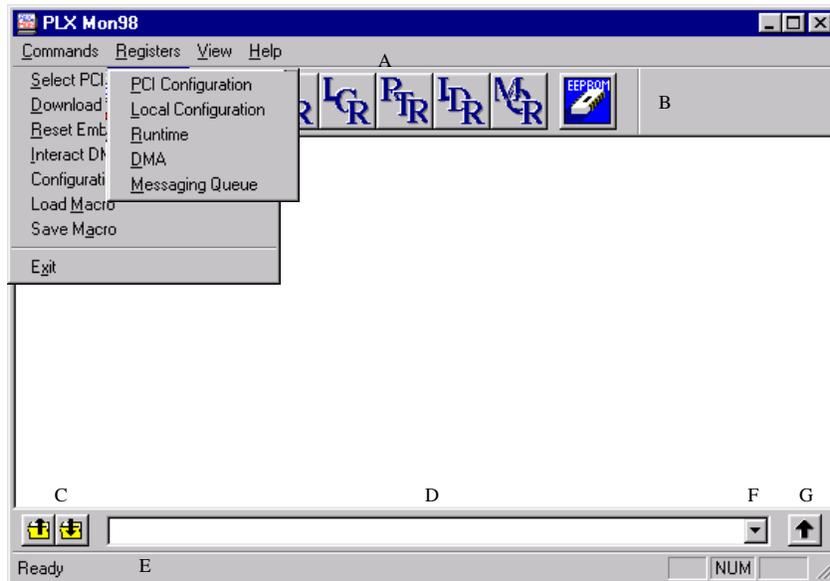


Figure 1 The PLXMon98 GUI

2.2.1 The PLXMon98 Toolbar

The PLXMon98 toolbar, shown in Figure 2, serves as an optional shortcut to the drop-down menu commands.



Figure 2 The PLXMon98 Toolbar

The first block of commands correspond to the Select PCI Device, Download to Embedded, Reset Embedded CPU, and Interactive DMA commands on the Commands drop-down menu. The next block of commands correspond to the Registers drop-down menu: PCI Configuration Registers, Local Configuration Registers, Runtime Registers, DMA Registers, and Messaging Queue Registers. The last button on the PLXMon98 toolbar reflects the Configuration EEPROM command window.

2.2.2 The Command Line Interface (CLI)

The CLI is where PLXMon97 commands are entered (see Figure 3 The Command Line Interface). To input commands to CLI; hit the ESC key, the Tab key, the Enter key, or click on the CLI with the mouse pointer with no dialog windows open. There will be a cursor in the CLI when it is ready to receive commands.



Figure 3 The Command Line Interface

Any command can be entered in the CLI and the results appear on main screen of PLXMon98. For more details on commands refer to the section 1.

A history of the commands are saved for the CLI. These commands can be accessed using up and down cursor arrow keys, or by clicking on the down arrow button (see Figure 1 for location of the down arrow button (F)) with the mouse pointer and selecting the command in the list.

The position of the CLI is on the bottom of PLXMon98's main screen . The CLI can be moved to the top of the main screen or can be floated as a pinup. Clicking on large up-arrow moves the CLI to the top of the main screen, changing the arrow to a large down-arrow. To return the CLI to the bottom, click on large down arrow. To float the CLI, press and hold the left mouse button anywhere on the CLI and drag to where you want to place it. To prevent moving the CLI by mistake, floating the CLI is only available after first moving the CLI to the top.

The Load Macro and Save Macro buttons (see (C) on Figure 1) allow storage and retrieval of existing command macros. When the Save Macro button is pressed, a Save As dialog box appears requesting the destination location of the macro file. Choose the desired file name and save location for the macro within this dialog box and press the Save button when complete. Pressing Load Macro results in the opening of an Open dialog box. Browse through the directories for the desired macro to be opened and push the Open button.

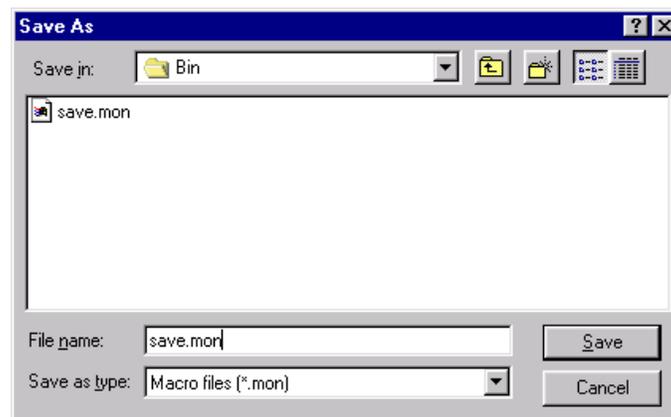


Figure 4 Save Macro Dialog Box

The default macro file name is save . mon and any new macro file names will be forced to have the . mon extension. A file can be selected by either clicking on file icon or by typing the file name in the File name text field.

See section 4.10.5.3 for more information on creating macros.

2.2.3 Status Bar

The status bar provides simple and useful tips. When the mouse is pointing on an object, a button or the CLI, in the main window of PLXMon98 the status bar displays some information on it. Tool-tips are also displayed when mouse pointer is held over a object for a short period.

2.3 Selecting Device

The `Select A PCI Device` menu item, shown in Figure 5, is used to select a PCI device to access. An asterisk before a PCI device in the device list denotes the currently selected device.

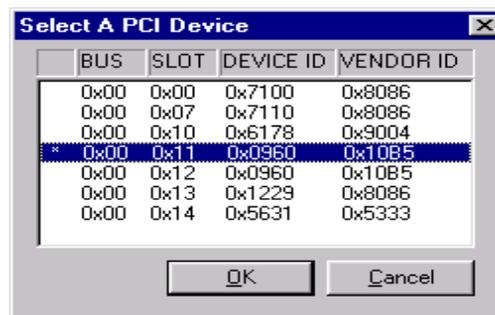


Figure 5 The Select A PCI Device dialog box

To select a new device, move the highlight to the desired PCI device by using either the mouse or the cursor arrow keys, and click the OK button or press the Enter key. The asterisk will not move to the new selection until OK button is pressed.

2.4 Launching PlxLdr97 Application

The `Download to Embedded` command serves as a shortcut to launch PlxLdr97 (See Figure 6). The `Reset IOP` button resets the I/O Platform to the power-on defaults. The `Download IOP` button initiates the download of the selected IOP file to the PCI device.

PLXMon98 looks at the Win32 registry for information on the location of `PlxLdr97.exe`. If `PlxLdr97.exe` is not in the directory defined within the Win32 registry, PLXMon98 will not be able to start it and a message box will appear to explain the failure.

Refer to the PCI SDK User's Manual for more information on using PlxLdr97.

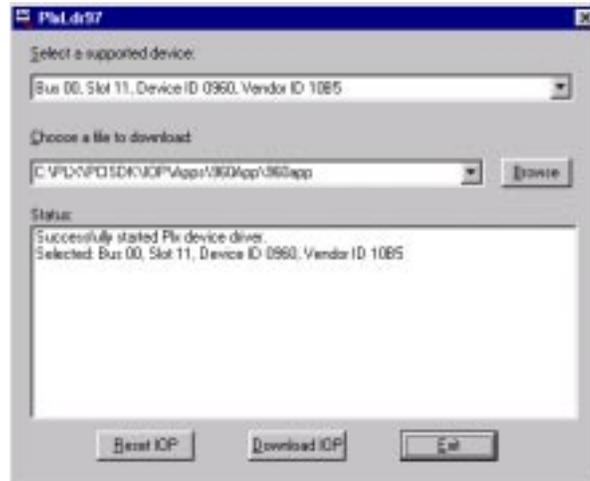


Figure 6 The Download to Embedded dialog box

2.5 Reset Embedded CPU

The `Reset Embedded CPU` command resets the currently selected PLX RDK device and displays the notice, shown in Figure 7, upon successful completion.



Figure 7 Reset Embedded CPU response box

The status of the reset can be monitored using Hyperterminal. The PLX icon on the Windows taskbar contains shortcuts to the currently supported PLX RDKs, those being the PCI 9080RDK-401, the PCI 9080RDK-860 and the PCI 9080RDK-401. Refer to PCI SDK User's manual for more information on the Hyperterminal shortcuts provided with the PCI SDK.

2.6 Interactive DMA

The `Interactive DMA` window, shown in Figure 8, helps the process of programming a DMA transfer. (This window will pop up as well by typing `idma` from the CLI).

The desired DMA channel and direction of transfer are set using the radio buttons. The `Direction` Button, marked with (A) in Figure 8, represents the direction of DMA transfer. In the spaces provided enter PCI and IOP addresses and the transfer size. To start the DMA transfer, push the `Direction` Button.

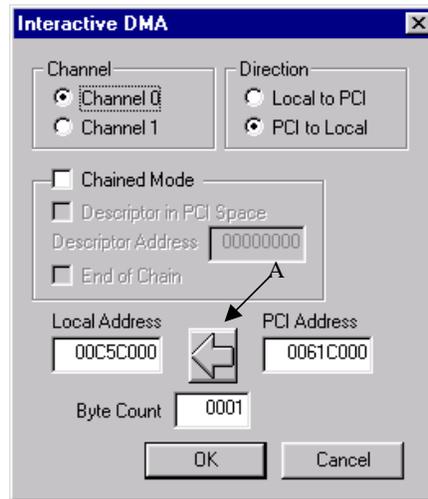


Figure 8 Interactive DMA dialog box

By default, IOP address is set to zero, and PCI address is set to Host PCI Buffer address.

Note: All the values are hexadecimal numbers and prefix, 0x, is omitted.

All information contained within the Interactive DMA dialog box will be saved when the OK button is pressed. This information is used by the DMA command (typed on the CLI, see section 4.10.4.7). The information will not be saved if the Cancel button is pressed.

Note: The PCI address is reset to the Host PCI Buffer every time the Interactive DMA dialog box is opened.

This version of PLXMon98 does not support DMA chaining.

3. Accessing PCI 9080 Registers

3.1 The PCI 9080 Register Dialog Box

Each of the PCI 9080's register groups has a distinct dialog box. Each dialog box has the register values, the register's PCI based addresses, and a description of the register. Some registers have check boxes and radio buttons to help in describing and setting the register values. When required additional dialog boxes are available for more complex registers.

Changes made in the register group dialog box are immediate. Changes made within a register's detailed dialog box will take effect only when the OK button is pressed.

3.2 The Register Group Dialog Boxes

The PLXMon98's toolbar contains five buttons for register accesses. They are for PCI Configuration Registers, Local Configuration Registers, Runtime Registers, DMA Registers, and Messaging Queue Registers.

3.2.1 PCI Configuration Register Group Dialog Box

The grayed text, pointed to by (A) in Figure 9, on the PCI Configuration Registers dialog box indicates that the values cannot be modified using this dialog box. The radio buttons and check boxes, pointed to by (B) in Figure 9, indicate the current settings of the register bit fields. To update the contents of the dialog box push the Refresh button.

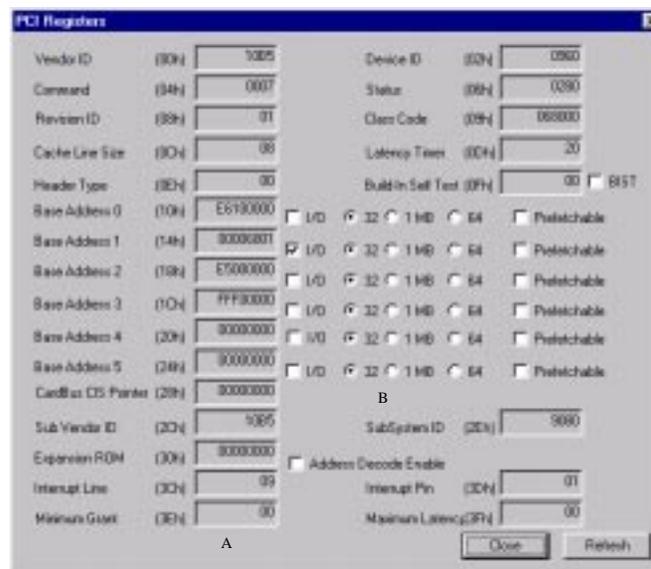


Figure 9 PCI Configuration Registers dialog box

3.2.2.1 The Mode/Arbitration Dialog Box

The Mode/Arbitration dialog box provides information on the current value Local/DMA Arbitration register and allows modification of that value (see Figure 11).

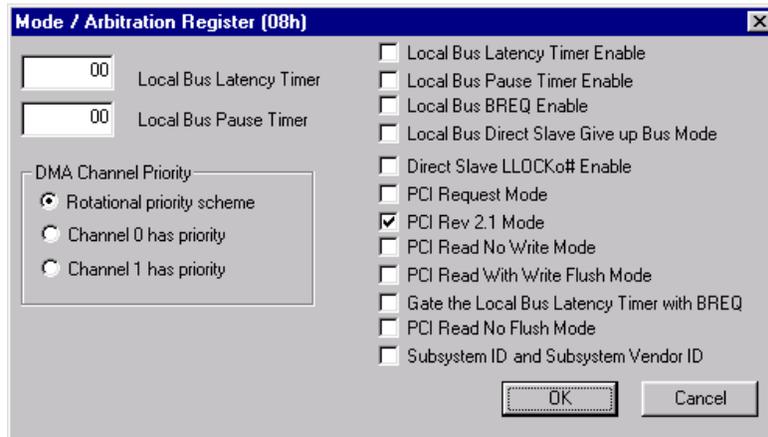


Figure 11 Mode/Arbitration dialog box

3.2.2.2 Endian Descriptor Dialog Box

The Endian Descriptor dialog box provides information on the current value of the Big/Little Endian Descriptor register and allows modification of that value (see Figure 12).

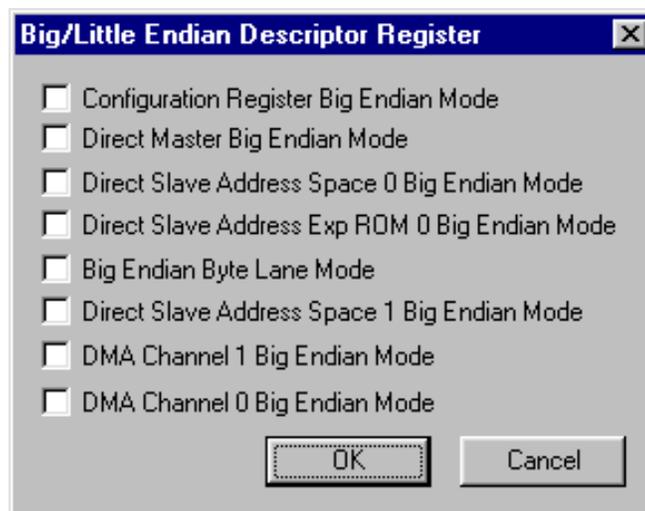


Figure 12 Endian Descriptor dialog box

3.2.2.3 The Space 0/Exp ROM Dialog Box

The Space 0/Exp ROM dialog box provides information on the current value of the Local Address Space 0/Expansion ROM Bus Region Descriptor register and allows modification of that value (see Figure 13).

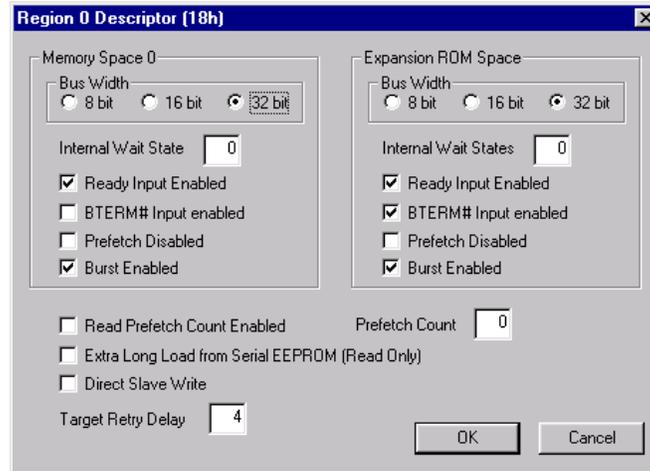


Figure 13 Space 0/Exp ROM dialog box

3.2.2.4 The DM PCI Remap Dialog Box

The DM PCI Remap dialog box provides information on the current value of the PCI Base Address (Remap) Register for Direct Master to PCI Memory and allows modification of that value (see Figure 14).

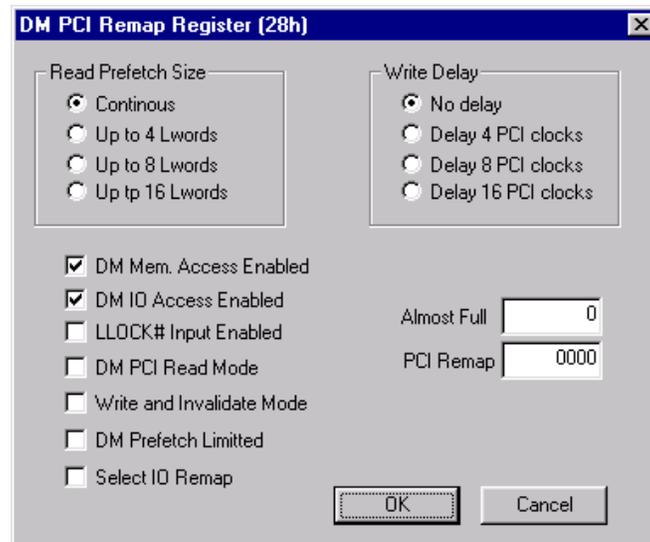


Figure 14 The DM PCI Remap Details dialog box

3.2.2.5 The DM Config IO Addr Dialog Box

The DM Config IO Addr dialog box provides information on the current value of the PCI configuration Address Register for Direct Master to PCI IO/CFG and allows modification of that value (see Figure 15).

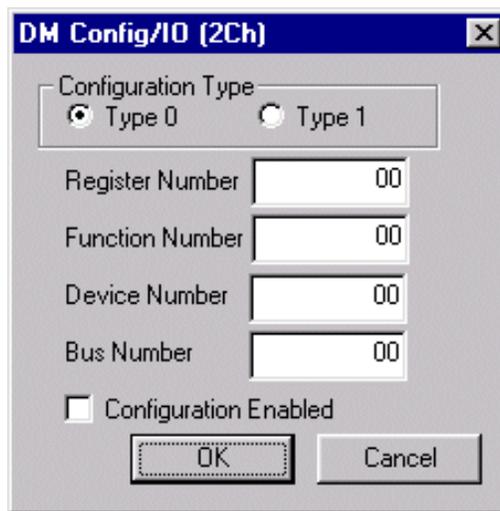


Figure 15 DM Config IO Addr dialog box

3.2.2.6 The Space 1 Dialog Box

The Space 1 dialog box provides information on the Local Address Space 1 Bus Region Descriptor register and allows modification of that value (see Figure 16).

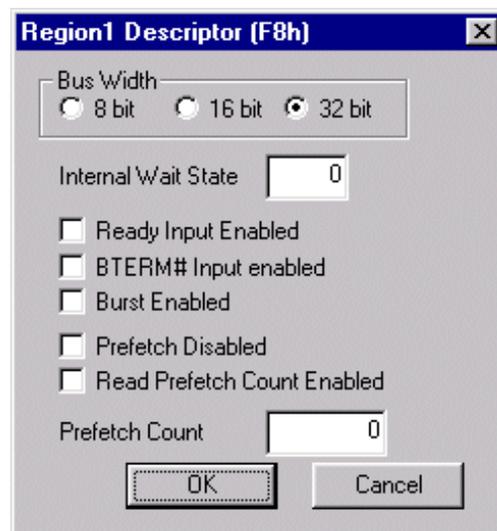


Figure 16 Space 1 dialog box

3.2.3 The Runtime Register Group Dialog Box

The Runtime Register Group dialog box displays the current register values. All register values can be modified by changing the contents of any register.

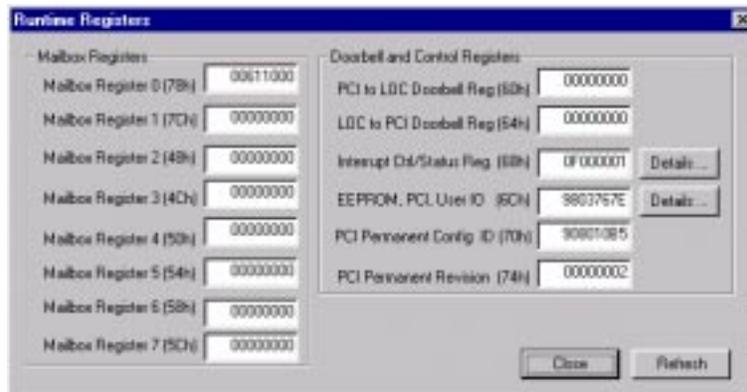


Figure 17 Runtime Registers dialog box

3.2.3.1 The Interrupt Control/Status Register Dialog Box

The Interrupt Control/Status Register Dialog Box provides information on the current value of the Interrupt Control/Status register.

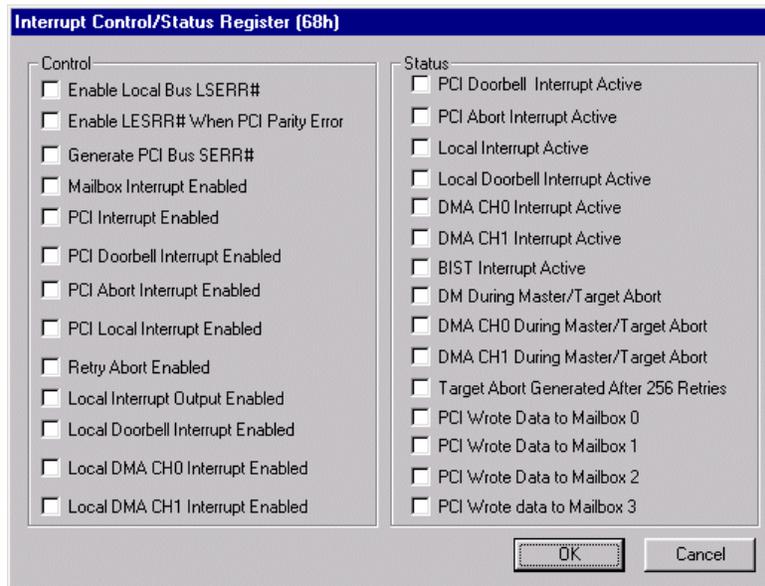


Figure 18 Interrupt Ctrl Details dialog box

The information contained in the dialog box is grouped into two categories, the Control bits and the Status bits. The control bits enable triggering of interrupts for certain events, such as DMA

events, doorbell events and others. The status bits cannot be modified directly. They show the current status of the various interrupt triggers.

3.2.3.2 The EEPROM, PCI, User IO Dialog Box

The EEPROM, PCI, User IO dialog box provides information on the current contents of the EEPROM Control, PCI Command Codes, User I/O Control, Init Control Register and allows modification of that value (see Figure 19). The Status section contained in this dialog box contain values that cannot be modified.

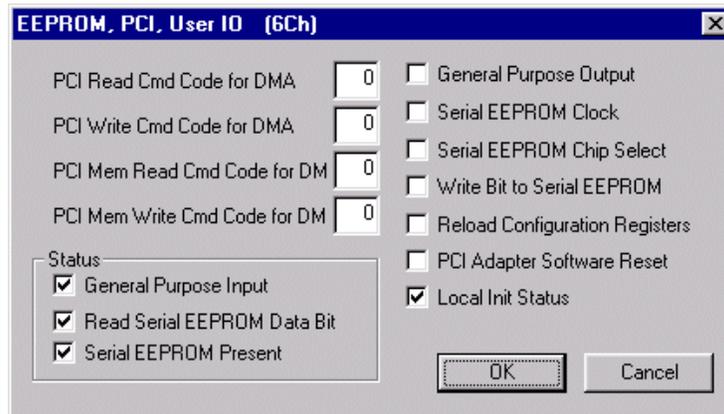


Figure 19 EEPROM, PCI, User IO Details dialog box

3.2.4 The DMA Register Group Dialog Box

The DMA Register Group dialog box contains the current values for the DMA registers for both DMA channels (see Figure 20).

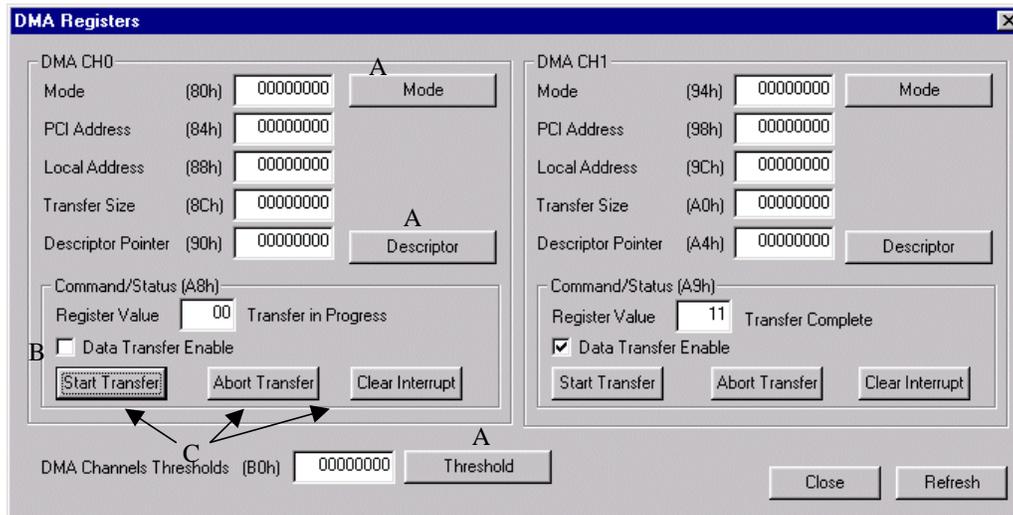


Figure 20 DMA Registers dialog box

The Start and Abort Transfer buttons, pointed by (C) in Figure 20, initiate and terminate the DMA transfer using the current information provided in the DMA registers for the given DMA channel. The Clear Interrupt button resets the interrupts to their default state. The Data Transfer Enable bit, pointed by (B) in Figure 20, enables DMA transfers and activates the Start, Abort, and Clear Interrupt buttons.

3.2.4.1 The DMA Mode Dialog Box

The DMA Mode dialog box provides information on the current value the DMA Channel's Mode Register and allows modification of that value (see Figure 21).

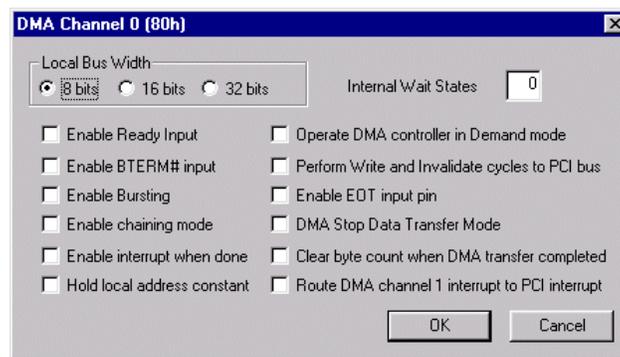


Figure 21 DMA Mode dialog box

3.2.4.2 The Descriptor Pointer Dialog Box

The Descriptor Pointer dialog box provides information on the current value of the DMA Channel's Descriptor Pointer Register and allows modification of that value (see Figure 22).



Figure 22 Descriptor Pointer dialog box

3.2.4.3 The DMA Channels Threshold Dialog Box

The DMA Channels Threshold dialog box provides information on the current value of the DMA Threshold Register and allows modification of that value (see Figure 23).

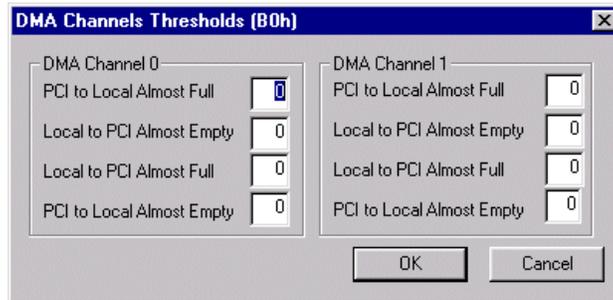


Figure 23 Threshold dialog box

3.2.5 The Messaging FIFO Register Group Dialog Box

The Messaging FIFO Register Group dialog box contains the current values for the Messaging FIFO Registers as shown in Figure 24.

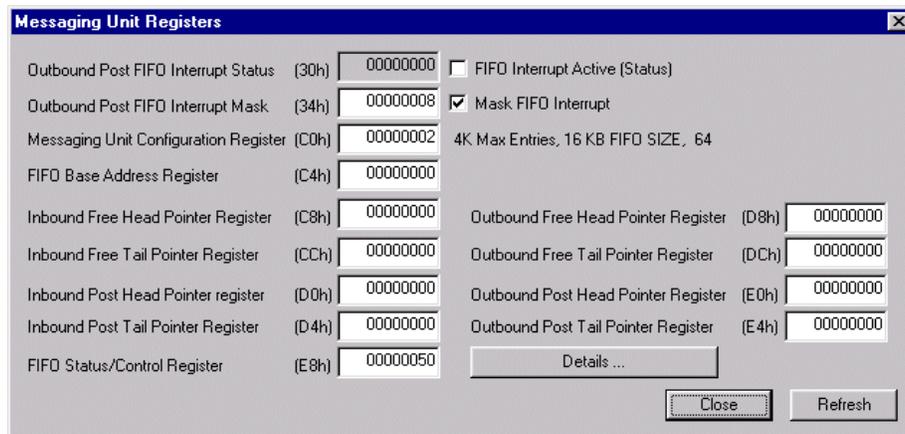


Figure 24 Messaging Unit Registers dialog box

All the register values can be modified with the exception of the Outbound Post FIFO Interrupt Status register. This register provides only the status of the Outbound Post FIFO interrupt and cannot be modified.

3.2.5.1 The FIFO Status/Control Register Dialog Box

The FIFO Status/Control Register dialog box provides information on the current value of the Queue Status/Control Register and allows modification of that value (see Figure 25).

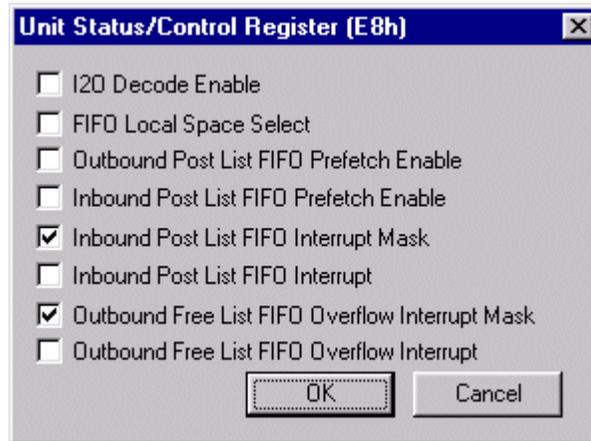


Figure 25 FIFO Status/Control Register dialog box

3.3 The Serial EEPROM Dialog box

The serial EEPROM initializes the PCI 9080 IC at power-up. The Serial EEPROM dialog box provides the current configuration data used to program the PCI 9080 and allows modification of that data.



Figure 26 Configuration Serial EEPROM dialog box

Fields in the Configuration EEPROM dialog box are divided into three groups according to the registers configured by the serial EEPROM.

Warning: Since programming serial EEPROM can be very critical to the system under use, all values are written to serial EEPROM only when Write button is pressed. To discard all changes and close the dialog box push the Close button.

To get the current values contained in the serial EEPROM push the Refresh button.

4. PLXMon98 Basics

4.1 Command Line

In the case of ambiguity between commands and parameters on the command line, a semicolon must be used between commands.

As hexadecimal notation includes the characters '0 through 9' and 'a through f', an ambiguity exists when a numeric parameter could be confused for a command, and vice-versa.

For example, to display a block of memory at 100, then input from port 200, you can enter:

```
db 100;i 200
```

The semicolon states that *i 200* is a new command and not a numeric parameter for the display command (See section 4.10.1.1). In this case, however, the semicolon is not needed because *i* cannot be interpreted as a hexadecimal number.

Therefore:

```
db 100 i 200
```

will be interpreted the same as the first example.

The semicolon is required in this example:

```
db 100; db 200
```

since the command could otherwise be interpreted as *db 2693332992*.

Parameters that are listed with an *expr* substring, such as *valexpr*, can be numbers, variables, or expressions. See sections 4.3 and 4.4 for more information about variables and expressions.

Angle brackets in command lines enclose required parameters. Text of this sort should not be typed verbatim, for example:

```
m <addrexp> [expr ...]
```

Square brackets in command lines enclose optional parameters. Text of this sort should not be typed verbatim, for example:

```
eb addrexp [INC] [expr ...]
```

The vertical bar, or pipe, indicates that either one parameter or the other must be provided, but not both. As in:

```
command xexpr | yexpr
```

Ellipses indicates that you can enter more than one value for the parameter. As in:

```
command expr [...]
```



4.2 Scrolling and Editing Command Lines

The program allows you to scroll through the command line history using the up and down arrow keys. You can edit any line by using the left and right arrow keys, backspace and escape.

Hitting Enter, Tab, or Esc key will bring the focus to Command Line Interface.

PLXMon98 can only be in *Insert* mode.

4.3 User-Variables

User-definable variables can be used as simple substitutes for obscure numbers or they can be combined with expressions, numbers and commands to create complex and powerful command sequences.

Create (or modify) a variable by typing a variable name, followed by the equals sign, and then a value. The value can be a legal combination of numbers, variables or expressions, as in the following:

```
x = 1
x = x+1
```

Eliminate a variable by leaving the right-hand-side empty, as in:

```
x =
```

Here is an example to show the usefulness of variables. The example outputs an incrementing value to a 16-bit port:

```
x = 0 [ow 100 x; x = x + 1; r 10]
```

You can use virtually any name of any length for your variables. Be careful to avoid using duplicate names; PLXMon98 scans the built-in command list before the variable list. If the variable name matches a built-in command, the command will be executed, and the variable will be ignored.

For more information on expressions, see section 4.4.

For more information on how to show variables, see section 4.10.5.1.

4.4 Expressions

Expressions are legal arrangements of numbers, variables and operators. (See section 4.3 for information on variables.) PLXMon98 lets you work with expressions 'on-the-fly' i.e. anywhere you enter numbers, you can enter expressions. Expressions can even be evaluated without a command right at the command line.

Expressions are evaluated left to right, but can be overridden with parenthesis. Expression operators include monadic and dyadic operators.

4.4.1 Monadic Operators

Monadic operators include one's complement, two's complement, etc. as listed in Table 2-1 Monadic Operators (where N refers to a number).

Syntax	Operation	Example (Hex)	Result
~N	Inverse of N	~1	Returns 0xFFFFFFFFE (One's complement)
-N	0 minus N	-1	Returns 0xFFFFFFFF (Two's complement)
+N	0 plus N	+1	Returns 0x00000001 (doesn't do anything!)
*N	Pointer N	*12345678	Returns value pointed to by 12345678

Table 1 Monadic Operations (where N refers to a number)

4.4.2 Dyadic Operators

Dyadic operators include addition, subtraction, etc. as listed in Table 2-2 Dyadic Operators (where N and M refer to numbers and variables).

Syntax	Operation	Example (Hex)	Result
N+M	N plus M	4+1	Returns 0x00000005
N-M	N minus M	4-1	Returns 0x00000003
N*M	N multiplied by M	7*3	Returns 0x00000015
N/M	N divided by M	7/3	Returns 0x00000002
N%M	N modulo M	7%3	Returns 0x00000001
N&M	N AND M	7&3	Returns 0x00000003
N M	N OR M	7 3	Returns 0x00000007
N^M	N XOR M	7^3	Returns 0x00000004

Table 2 Dyadic Operators (where N and M refer to numbers and variables)

- *Note:* Any division by zero will return 0xFFFFFFFF.

4.5 Macros

Macros allow you to create command sequences, and call them by a name of your choice. To execute the command sequence, use the macro name instead. Macros are convenient if you find yourself using a command sequence repeatedly.

For more information on creating and using macros, see section 4.10.5.3.



Both macros and variables can be saved to a file, to be restored in a future session. See sections 4.10.5.9 and 4.10.5.10 for information on the *save* and *read* commands.

4.6 Ranges

Several commands require a full or partial range to be specified. The range includes a starting address and a byte count. The byte count can be implied with an ending address or explicit; as in:

```
db 4000 4020
```

which implies a count of 20 (4020-4000).

```
db 4000 L 20
```

which specifies a count of 20.

```
db 4000 20
```

which also specifies a count of 20.

- **Note:** The count must be smaller than the starting address, otherwise, use the format of example two or three.

Commands that accept partial ranges use the starting address you type and substitute a default count. Some commands, such as the display commands, will substitute both the starting address and the count if you do not provide them.

4.7 Repeating Command Lines (Looping)

The *r* command allows you to repeat all or part of a command line infinitely, or for the number of iterations you specify. You can also nest repeat loops using nesting tokens '[' and ']'. See section 4.10.5.5 for more details.

4.8 On-line Help

PLXMon98 offers built-in help messages for every command. See section 4.10.5.13 for more information.

4.9 PLXMon98 Batch Files

You can create a text file full of command lines and pass the file to PLXMon98 for execution in one of two ways;

1. the *read* command, or
2. the *Load Macro* button from CLI.

4.9.1 The read command

Use *read* to read and execute the command file. Refer to section 4.10.5.10 for details.

4.9.2 Documenting your file

It is a good idea to add comments to your command file. This can be done using *echo* (See section 4.10.5.6).

4.10 PLXMon98 Command Set

Table 2-3 Index of Commands shows all PLXMon98 commands listed by category.

Command Set	Command Line	Reference
Memory Manipulation		
Display	d, db, dw, dl, dd	4.10.1.1
Enter	e, eb, ew, el, ed	4.10.1.2
Move	m, mb, mw, ml, md	4.10.1.3
Compare	c	4.10.1.4
Search	s	4.10.1.5
Input/Output		
Input	i, ib, iw, il, id	4.10.2.1
Output	Ob, ow, ol, od	4.10.2.2
PCI Commands		
Show or Select a Device	dev	4.10.3.1
PLX Device Commands		
PCI Configuration Registers	pcr	4.10.4.1
Local Configuration Registers	lcr	4.10.4.2
Runtime Registers	rtr	4.10.4.3
Local DMA Registers	ldr	4.10.4.4
Messaging Unit Registers	mqr	4.10.4.5
Interactive DMA Programming	idma	4.10.4.6
DMA Programming	dma	4.10.4.7
Read Serial EEPROM	re	4.10.4.8
Write Serial EEPROM	we	4.10.4.9
Internal Commands		
Show Variables	vars	4.10.5.1
Expressions	expr	4.10.5.2

Command Set	Command Line	Reference
Define a Macro	define	4.10.5.3
Show Macros	macs	4.10.5.4
Repeat	r	4.10.5.5
Echo	echo	4.10.5.6
Wait	wait	4.10.5.7
Base	base	4.10.5.8
Save	save	4.10.5.9
Read	read	4.10.5.10
Range	range	4.10.5.11
Command Edit	cmd_edit	4.10.5.12
Help	help, h, ?	4.10.5.13

Table 3 Index of Commands

4.10.1 Memory Manipulation

The memory manipulation commands allow you to display, modify, search and compare memory in several ways. From the user's perspective, all memory is referenced as a flat, non-segmented, 4 gigabyte space.

- **Note:** Exercise caution when manipulating memory as there are no restrictions on memory accesses and it is easy to crash your system.

Most of the memory manipulation commands accept range parameters. See section 4.6 for information on specifying ranges.

Display, enter, and move have variations for byte, word and long word sizes. PLXMon98 uses the appropriate byte, word or long word assembler instruction (hence the appropriate bus cycle) for each variation.

d and *e* adopt the size (byte, word or long word) of the most recently used display or enter command; that is, if you just used *el*, *d* will display long words.

4.10.1.1 Display: *d*, *db*, *dw*, *dl*, *dd*

```
d [range]
db [range]
dw [range]
dl [range]
dd [range]
```

range	memory address and count (See section 4.6)
-------	--

This group of commands displays a range of memory. You can display in byte, word, long word or double format. (Long word is the same as double.)

Display commands accept two, one or zero parameters:

two	range (section 4.6)	displays the specified memory range
one	address only	displays memory starting at the address specified for the default count (0x80 bytes)
zero		displays memory starting <u>after</u> the last address for the default count (0x80 bytes)

Both programs display the ASCII values of displayed memory in a separate column. This column displays in byte-order, regardless of the display size (byte, word, long word or double.)

4.10.1.2 Enter: e, eb, ew, el, ed

```
e [addrexpr] [INC incexpr] [valexr ...]
eb [addrexpr] [INC incexpr] [valexr ...]
ew [addrexpr] [INC incexpr] [valexr ...]
el [addrexpr] [INC incexpr] [valexr ...]
ed [addrexpr] [INC incexpr] [valexr ...]
```

addrexpr	memory address to start entering
incexpr	address increment
valexpr	value to enter

Use these commands to modify – or interactively examine and modify – memory. You can specify bytes, words, long words or doubles. (Long word is the same as double.)

Enter commands require a starting address. If *valexpr* is not specified, PLXMon98 interactively requests information. This mode is the *interactive-enter* mode. The interactive-enter mode displays the value at the starting address and allows you to type in a new value. You then have three choices:

1. type a new value and press the return or spacebar keys (The new value is actually written to memory when you press return or spacebar.)
2. press the spacebar, or
3. press the Enter key

Pressing the spacebar continues interactive-enter mode at the next address, while pressing Enter ends interactive enter mode.



Pass at least one *valexpr* on the command line to have values entered in a non-interactive fashion.

Use `INC incexpr` when you want to enter values at non-sequential addresses. You may, for example, want to enter a new value every 100 bytes starting at your starting address:

```
e 123400 INC 100 a b c d
```

The `INC` specifier must be capitalized, and *incexpr* must be supplied. This unique feature can be applied to both interactive and non-interactive modes.

4.10.1.3 Move: *m*, *mb*, *mw*, *ml*, *md*

```
m <range> <destexpr>
mb <range> <destexpr>
mw <range> <destexpr>
ml <range> <destexpr>
md <range> <destexpr>
```

range	source address and count (See section 4.6)
destexpr	Destination address

Use these commands to move blocks of memory from one location to another. You can specify byte, word, long word or double variations. (Long word is the same as double.)

The byte variations, *m* and *mb*, are appropriate for most needs; the other variations are provided because they perform 16 and 32 bit data transfers at the assembler level.

- **Note:** Unlike *d* and *e*, *m* always moves bytes. Moves do not use DMA so do not expect bursting on the PCI bus. As of this writing, moves are limited to 64KB.

4.10.1.4 Compare: *c*

```
c <range> <addrexpr>
```

range	left address and count (see section 4.6)
addrexpr	right address

This command is useful for checking the result of a move or DMA.

The command performs a byte-by-byte comparison of data starting at the left and right address locations. Both the addresses and data bytes of any invalid comparisons are shown.

4.10.1.5 Search: *s*

```
s <range> <expr> [...] | [text]
```

range	search address and count (see section 4.6)
-------	--

expr	first byte of search pattern
...	Subsequent bytes of search pattern
“text”	text pattern being searched

You can search for a text or data pattern that is one or more bytes long. The address of any matching pattern within the range is printed.

The text parameter is case-sensitive, and must be typed between double quote characters

4.10.2 Input/Output

The input and output commands offer unrestricted port I/O. Variations for byte, word or long word port I/O are provided. The assembler instruction used for the port transfer is appropriate for the I/O width (byte, word or long word) you specify.

4.10.2.1 Input: i, ib, iw, il, id

```

i [range]
ib [range]
iw [range]
il [range]
id [range]

```

range	port address and count (see section 4.6)
-------	--

These commands read and display from a range of port addresses. You can display in byte, word, long word or double format. (Long word is the same as double.)

Input commands accept two, one or zero parameters:

two	range (section 4.6)	the specified range of ports is read and displayed
one	address only	the port address specified is read and displayed
zero		the port address starting after the last port address read is read and displayed.

The most popular usage is to specify the port address (one parameter) as in:

```
iw 100
```

i adopts the size (byte, word or long word) of the most recently used input command; that is, if you just used *il*, *i* will read and display long words.



PLXMon98 displays the ASCII values of the port data in a separate column. These values are displayed in byte-order, regardless of the size specified (byte, word, long word or double.)

4.10.2.2 Output: ob, ow, ol, od

ob <exprport> <exprvalue>

ow <exprport> <exprvalue>

ol <exprport> <exprvalue>

od <exprport> <exprvalue>

exprport	port address
exprvalue	value to be written

Use an output command to write data to ports. You can write bytes, words, long words or doubles (Long word is the same as double.)

4.10.3 PCI Commands

PCI commands apply to all PCI devices – not just PLX PCI devices.

4.10.3.1 Show or Select a Device: dev

dev [devnum]

devnum	logical device number
--------	-----------------------

With no parameters, the command displays a list of all devices found on all buses. Each line shows the device's logical device number, vendor ID, device ID, and a few PCI configuration registers. The currently selected device is marked with an asterisk.

If *devnum* is given, and it is in the list of registered devices, the associated PCI device is selected. User-variables 's0', 's1', 's2', 's3', are set to the values found in the selected device's PCI configuration registers 0x18, 0x1c, 0x20, 0x24 respectively. These variables refer to PCI base addresses for the local address spaces of the device.

When new device is selected, its vendor ID and device ID is verified. If the new selected is not PXL device, register features from PLXMon98 are disabled. The disabled featured are including *Interactive DMA Transfer, PCI Configuration Register, Local Configuration Register, Runtime Register, Local DMA Register, Messaging Unit Register, and Configuration EEPROM*. Selecting PLX device will enable disabled features.

The user-variables 's0', 's1', 's2', 's3', combined with PLXMon98's '+' operator, makes it easy to access the device's memory. For example, to display a portion of the memory of the current device at offset 1000, you could use:

d s0+1000

- **Note:** You may have to adjust the device's re-map register to access a desired portion of the device's local memory.

4.10.4 PLX Device Commands

PLX device commands are designed for use with the PLX family of PCI devices.

4.10.4.1 PCI Configuration Registers: *pcr*

```
pcr [regnum] [regval]
```

regnum	Register number (long-word aligned)
regval	Register value (long-word)

With no parameters, this command displays all of the currently selected device's PCI configuration registers. When a legal *regnum* is specified, one register is displayed. Apply the *regval* parameter to set a PCI Configuration Register to a specific value.

- **Note:** *pcr* does not prohibit writing to unwritable registers. Exercise caution when using this command in order to avoid corrupting your configuration.

4.10.4.2 Local Configuration Registers: *lcr*

```
lcr [regnum] [regval]
```

regnum	register number (long-word aligned)
regval	register value (long-word)

With no parameters, this command displays all of the currently selected device's Local Configuration Registers. When a legal *regnum* is specified, one register is displayed. Apply the *regval* parameter to set a Local Configuration Register to a specific value.

- **Note:** *lcr* does not prohibit writing to unwritable registers. Exercise caution when using this command in order to avoid corrupting your configuration

4.10.4.3 Runtime Registers: *rtr*

```
rtr [regnum] [regval]
```

regnum	register number (long-word aligned)
regval	register value (long-word)

With no parameters, this command displays all of the currently selected device's Runtime Registers. When a legal *regnum* is specified, one register is displayed. Apply the *regval* parameter to set a Runtime Register to a specific value.



- **Note:** *rtr* does not prohibit writing to unwritable registers. Exercise caution when using this command in order to avoid corrupting your registers.

4.10.4.4 Local DMA Registers: *ldr*

`ldr [regnum] [regval]`

regnum	register number (long-word aligned)
regval	register value (long-word)

With no parameters, this command displays all of the currently selected device's Local DMA Registers. When a legal *regnum* is specified, one register is displayed. Apply the *regval* parameter to set a Local DMA Register to a specific value.

- **Note:** *ldr* does not prohibit writing to unwritable registers. Exercise caution when using this command in order to avoid corrupting your registers.

4.10.4.5 Messaging Unit Registers: *mqr*

`mqr [regnum] [regval]`

regnum	register number (long-word aligned)
regval	register value (long-word)

With no parameters, this command displays all of the currently selected device's Messaging Unit Registers. When a legal *regnum* is specified, one register is displayed. Apply the *regval* parameter to set a Messaging Unit Register to a specific value.

- **Note:** *mqr* does not prohibit writing to unwritable registers. Exercise caution when using this command in order to avoid corrupting your registers.

4.10.4.6 Interactive DMA programming: *idma*

`idma`

Interactive DMA programming helps you through the process of programming the DMA. (The same results are generated using register and memory manipulation commands.)

idma only applies to PLX devices that allow their DMA registers to be programmed from the PCI side.

When using *idma*, it is a good idea to have PLX's DMA register description at hand. Performing this command will bring up *Interactive DMA Transfer* screen. The screen needs information about:

- DMA channel
- Transfer Direction
- Chained mode

- PCI Address
- Local Address
- Count

When chained mode selected, you need to provide:

- Descriptor in PCI space
- Descriptor address
- End of chain

idma is generic for all of PLX's DMA implementations so it may generate questions that are not appropriate for your device.

Each field shows the current setting; make any necessary change and press on the button with an arrow.

Bear in mind that you are responsible for all memory management. This can get confusing, especially in chained mode, because you must manage memory for chain descriptors and data buffers, in at least two physical memory spaces, the system memory and the *IOP* memory space.

Chained mode: *idma* helps you manage descriptors by suggesting addresses to store the descriptors. The suggested addresses are based on user-variables:

- suggested PCI-side descriptor storage – *hbuf*
- suggested local-side descriptor storage – *membase*

You can change these variables to match your system.

Chained mode: when you finish entering the last descriptor, *idma* shows all the descriptors in the chain. It's a good idea to review the chain's transfer details and descriptor locations before starting the DMA transfer.

Chained mode trick: to make the DMA chain loop forever, make the last descriptor refer to the first descriptor, and use memory manipulation to reset the 'end-of-chain' bit in the last descriptor. Pressing OK will save the configuration. Kick off the DMA using the *dma* command (see section 4.10.4.7).

4.10.4.7 DMA Programming: *dma*

dma

Non-interactive DMA programming. This command simply kicks off a DMA transfer, the chain descriptors, mode, address and count registers are used 'as-is'.

You can use *idma* (section 4.10.4.6) to set up and test a DMA transfer, then use *dma* to repeat the transfer quickly.

4.10.4.8 Read Serial EEPROM: *re*

re [range]



range	destination range (defaults to user-variable 'hbuf', for 0x40 words)
-------	--

Read selected PCI device's Serial EEPROM into *range* (see section 4.6) of PCI memory. (Applies to PLX PCI devices with Serial EEPROMs connected.)

With no parameters, this commands reads 0x40 words (0x80 bytes) into PCI memory specified by the *hbuf* variable. If *range* includes the destination address, but not the length, the length defaults to 0x40 words.

This command was written specifically for National Semiconductor's NMC93CS06 and NMC93CS46 Serial EEPROM devices. Refer to National Semiconductor documentation for device details.

Unlike other commands, the range always specifies words to match the NMC93CS06/CS46 word width.

For more information on writing to the Serial EEPROM, see section 4.10.4.9.

4.10.4.9 Write Serial EEPROM: we

we <range>

range	source range
-------	--------------

Write *range* (see section 4.6) of PCI memory to the selected PCI device's Serial EEPROM. Applies to PLX PCI devices with Serial EEPROMs connected.

This command was written specifically for National Semiconductor's NMC93CS06 and NMC93CS46 Serial EEPROM devices. Refer to National Semiconductor documentation for device details.

Unlike other commands, the range always specifies words to match the NMC93CS06/CS46 word width.

Unlike the Serial EEPROM *read* command, this command requires the source range to be specified. The length must resolve to 0x40 words or less.

This command could fail for many reasons; the NMC93CS06/CS46 includes features to prevent accidental writes, including the ability to permanently prevent writes to a portion of its memory. Before writing a value to the NMC93CS06/CS46, this command issues a WREN (Write Enable) instruction to the NMC93CS06/CS46. After writing the value, a WDS (Write Disable) instruction is issued.

To read, modify and write the Serial EEPROM, use *read* (See section 4.10.4.8), and then use the memory manipulation functions followed by this command.

4.10.5 PLXMon98 Internal Commands

4.10.5.1 Show Variables: vars

vars

Show all variables.

For more information on how to define variables, see section 4.3.

4.10.5.2 Expressions: expr

expr <valexpr>

valexpr	expression to be evaluated
---------	----------------------------

Evaluates *valexpr* and shows the result in hexadecimal and signed-decimal.

Expressions can be evaluated without this (or any) command. See section 4.4.

4.10.5.3 Define a Macro: define

define <macname> [macbody]

macname	macro name
macbody	macro body

Define (or delete) a macro.

The first parameter names the macro while the rest of the line is stored as the macro body. The macro body is not checked for validity and can include commands, variables and other macros.

- **Note:** Referencing the same macro within a macro works, but it will probably overflow the stack.)

To execute a macro, type the macro name anywhere you would normally type a command.

Once a macro is defined, it can be deleted by using the define command with the macro name as the only parameter. A macro can be redefined by entering the macro name followed by a new macro body.

PLXMon98 scans the built-in command list before the macro list, and so if your macro name matches a built-in command, the command will be executed, and your macro will be ignored.

There is room for approximately 25 macros at 80 characters per macro.

Macros can be saved to a file for future use. See section 4.10.5.9 for information.

4.10.5.4 Show Macros: macs

macs



Show all macros. See section 4.10.5.3 for information on defining macros.

4.10.5.5 Repeat: r

r [count]

count	iteration count (number, variable or expression)
-------	--

Repeat all or part of a command line infinitely or for a specific number of iterations.

In its simplest application, you append the repeat command to the end of your command line, and PLXMon98 will process the entire command line repeatedly until you press a key. In the middle of the process, place mouse pointer on PLXMon98's display area and press and hold down left mouse button to hold repeat command until the pressed mouse button is released on PLXMon98's display area.

```
ob 100 ab; r
```

Adding a *count* limits the number of iterations:

```
ob 100 ab; r 10
```

You can use nesting tokens '[' and ']' to repeat part of a command line:

```
ib 200 [ob 100 ab; r 10]
```

This example inputs one byte from port 200, then outputs AB to port 100 10 times. The absence of nesting tokens tells the application to repeat starting at the beginning of the line.

There is no limit to the number of nesting tokens you can apply, for example:

```
d 1000 [ib 200 [ob 100 ab; r 10]]r 5
```

- **Note:** the repeat command can be used in a macro body, but it only makes sense to do so if a *count* is specified.

4.10.5.6 echo

echo <text>

text	any text up to the enter key
------	------------------------------

Echo the rest of the command line. Use this command to document a PLXMon98 command file. (See section 4.9)

4.10.5.7 wait

wait <countexpr>

countexpr	number of cycles to wait
-----------	--------------------------

Wait while the processor down-counts *countexpr*. The wait loop is uncalibrated; PLXMon98 sits in a software loop until *countexpr* hits zero.

The longest wait is obtained by passing zero as the parameter. This is equivalent to *wait 100000000*.

- *Note:* There is no way to break out of a wait, so be careful using a large waitexpr.

4.10.5.8 base

base [baseexpr]

baseexpr	new radix (base 16 max)
----------	-------------------------

Set PLXMon98's radix to a new base. If *baseexpr* is a number (not a variable or an expression), it must be in base 10. PLXMon98's highest (and default) base is base 16.

Entering numeric parameters to a command using digits greater than the radix (but less than F) will generate an error message. However, the command will continue and the offending digits are truncated to zero.

This command, however, doesn't change base for values displayed register screens.

Use *base* with no parameters to determine PLXMon98's current radix.

4.10.5.9 save

save [filename]

filename	name of command file
----------	----------------------

Save current variables and macros to *filename*. The file is saved as plain text. The variables and macros are saved as PLXMon98 commands; you can edit the file and add PLXMon98 commands, and then execute the file at any time using *read* (See section 4.10.5.10).

With no parameters, *save* will use SAVE.MON as the default filename.

4.10.5.10 read

read [filename]

filename	name of command file
----------	----------------------

Read and execute a PLXMon98 command file. PLXMon98 reads characters from the file as if they were typed-in command lines.

With no parameters, *read* will use SAVE.MON as the default *filename*.

save saves PLXMon98's macros and variables in a format ready for *read*, (See section 4.10.5.9).



4.10.5.11 range

range

This is a placeholder for PLXMon98's on-line help for using ranges. See section 4.6 for information on specifying ranges.

4.10.5.12 cmd_edit

cmd_edit

This command is a placeholder for PLXMon98's on-line help for the editing command lines. See section 0 for information on editing command lines.

4.10.5.13 help, h, ?

help [command]

h [command]

? [command]

command	command name
---------	--------------

Display on-line help for PLXMon98 commands. With no parameter, the help commands print a complete list of commands. Enter a command as a parameter to show the on-line help message for that command.

Enter an asterisk as a parameter to show all on-line help messages for all commands.

A command's help text may show a parameter within square braces. Such a parameter is optional for the command.

A command's help text may show ellipses following a parameter. The ellipses indicate that more parameters of the same type can follow the first parameter.

5. Customer Support

Prior to contacting customer support, please ensure that you are situated close to the computer that has PLXMon98 installed and that you have the following information:

1. Serial Number of the PCI 9080RDK
2. Type of processor on the PCI 9080RDK
3. Operating System version and type, and
4. Description of problem.

You may contact PLX customer support at:

Address: PLX Technology, Inc.
390 Potrero Avenue
Sunnyvale, CA 94086

Phone: 408-774-9060
Fax: 408-774-2169
Web: <http://www.plxtech.com>



This software design kit has been developed and tested by Vitana Corporation.
For more information regarding SDK and RDK designs, please contact:

Vitana Corporation
Tel: 613-749-4445
Email: rdk@vitana.com
Web: www.vitana.com

For technical support questions, please contact PLX Customer Support.