

PCI SDK User's Manual

Release 2.0, initial publishing June 22, 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 SDK; for any loss or claims by third parties which may arise through the use of this SDK; 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: sdkuser.doc

Table of Contents

1. INTRODUCTION	1-1
1.1 About This Manual	1-1
1.2 Where To Go From Here	1-1
1.3 Other PCI SDK Manuals	1-1
2. GENERAL INFORMATION	2-1
2.1 Introduction.....	2-1
2.2 Features.....	2-1
2.3 Terminology.....	2-1
3. PCI SDK INSTALLATION	3-1
3.1 Unpacking.....	3-1
3.2 Minimum System Requirements.....	3-1
3.3 PLX Devices Supported.....	3-1
3.4 Development Requirements.....	3-1
3.4.1 Development Tools Needed.....	3-2
3.5 Software Installation	3-2
3.5.1 WinNT Installation Procedures.....	3-2
3.5.2 Removing All Software.....	3-2
3.5.3 PCI SDK v2.x Compatibility with PCI SDK v1.2.x	3-3
3.5.4 Troubleshooting	3-4
3.5.4.1 Increasing Available System Pages In WinNT.....	3-5
3.5.5 Customer Support.....	3-5
4. IOP SOFTWARE	4-1
4.1 Introduction.....	4-1
4.2 PLXRom Application	4-1
4.2.1 MiniRom Application	4-1
4.2.2 PLXRom Communication Configuration	4-1
4.3 Downloading Procedures.....	4-2
5. WINDOWS BASED SOFTWARE	5-1
5.1 Introduction.....	5-1
5.2 PLXMon98	5-1
5.3 PLXMon97	5-1
5.4 PLXLdr98.....	5-1
5.5 PLXTask	5-2
5.6 Windows NT Device Drivers.....	5-3
5.6.1 Starting And Stopping.....	5-3

5.6.2	Event Logging.....	5-4
5.6.3	Registry Configuration.....	5-5
5.7	Adding PCI SDK Support for Custom Boards	5-6
6.	USING THE PCI SDK WITH A NEW BOARD	6-1
7.	PLX MONITOR USER'S MANUAL	7-1
7.1.1	What is PLXMon97?.....	7-1
7.1.2	What is Back-End Monitor Level 2?.....	7-1
7.1.3	Who should use PLXMon97 and Back-End Monitor Level 2?	7-2
7.1.4	PLX Disclaimer.....	7-2
7.2	Getting PLXMon97 Started.....	7-2
7.3	Getting BemL2 Started	7-2
7.4	PLXMon97 and Back-End Monitor Level 2 Basics	7-2
7.4.1	Command Line.....	7-2
7.4.2	Scrolling and Editing Command Lines	7-3
7.4.3	User-Variables.....	7-3
7.4.4	Expressions	7-4
7.4.4.1	Monadic Operators	7-4
7.4.4.2	Dyadic Operators	7-4
7.4.5	Macros.....	7-5
7.4.6	Ranges.....	7-5
7.4.7	Repeating Command Lines (Looping).....	7-5
7.4.8	On-line Help.....	7-5
7.4.9	PLXMon97 'Batch' Files.....	7-5
7.4.9.1	Command Prompt standard-in redirection.....	7-6
7.4.9.2	PLXMon97's read command.....	7-6
7.4.9.3	Documenting your file	7-6
7.4.10	When PLXMon97 Starts	7-6
7.4.11	When The Back-End Monitor Level 2 Starts.....	7-7
7.5	PLXMon97 and Back-End Monitor Level 2 Command Set.....	7-7
7.5.1	Starting And Stopping Back-End Monitor Level 2.....	7-7
7.5.2	Memory Manipulation	7-7
7.5.2.1	Display: d, db, dw, dl, dd.....	7-8
7.5.2.2	Enter: e, eb, ew, el, ed.....	7-8
7.5.2.3	Move: m, mb, mw, ml, md.....	7-9
7.5.2.4	Compare: c.....	7-9
7.5.2.5	Search: s	7-9
7.5.3	Input/Output	7-10
7.5.3.1	Input: i, ib, iw, il, id	7-10
7.5.3.2	Output: ob, ow, ol, od	7-11

7.5.4	PCI Commands	7-11
7.5.4.1	Show or Select a Device: dev	7-11
7.5.4.2	PCI Configuration Registers: pcr.....	7-11
7.5.5	PLX Device Commands.....	7-12
7.5.5.1	Local Configuration Registers: lcr.....	7-12
7.5.5.2	Run-Time Registers: rtr	7-12
7.5.5.3	Local DMA Registers: ldr.....	7-12
7.5.5.4	Messaging Unit Registers: mqr.....	7-13
7.5.5.5	Interactive DMA programming: idma	7-13
7.5.5.6	DMA Programming: dma	7-13
7.5.5.7	Read Serial EEPROM: re	7-14
7.5.5.8	Write Serial EEPROM: we	7-14
7.5.6	PLXMon97 Internal Commands	7-15
7.5.6.1	Show Variables: vars	7-15
7.5.6.2	Expressions: expr.....	7-15
7.5.6.3	Define a Macro: define	7-15
7.5.6.4	Show Macros: macs	7-15
7.5.6.5	Repeat: r	7-16
7.5.6.6	echo.....	7-16
7.5.6.7	wait	7-16
7.5.6.8	base	7-16
7.5.6.9	save	7-17
7.5.6.10	read.....	7-17
7.5.6.11	range	7-17
7.5.6.12	cmd_edit.....	7-17
7.5.6.13	quit, q.....	7-18
7.5.6.14	help, h, ?	7-18

List of Figures

Figure 3-1 Components of the PCI SDK	3-1
Figure 3-2 Configuration EEPROM Settings For The PCI 9080RDK-860.....	3-4
Figure 3-3 Configuration EEPROM Settings For The PCI 9080RDK-401B.....	3-4
Figure 4-1 PLXRom running within a HyperTerminal session.....	4-2
Figure 5-1 PLXLdr98 Download Window	5-2
Figure 5-2 Example Taskbar With PLXTask	5-3
Figure 5-3 The Devices Utility Window.....	5-4
Figure 5-4 The Event Viewer Window.....	5-4
Figure 5-5 The Event Detail Window.....	5-5
Figure 5-6 The Registry Window	5-5
Figure 5-7 PLXTask's Supported Devices Window	5-6
Figure 5-8 PLXTask's New Supported Device Window.....	5-6

List of Tables

Table 7-1 Monadic Operators (where N refers to a number or variable).....	7-4
Table 7-2 Dyadic Operators (where N and M refer to numbers and variables).....	7-4



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.



PLX Copyright Message Guidelines

The following copyright message must appear in all software products generated and distributed by PLX customers:

“Copyright (c) 1998 PLX Technology, Inc.”

Requirements:

- Arial font,
- Font size 12
- Bold type
- Must appear as shown above in the first section or the so called “Introduction Section” of all manuals
- Must also appear as shown above in the beginning of source code as a comment



1. Introduction

1.1 About This Manual

This manual provides information about the functionality of the PCI SDK. Customers have the choice of using the PCI SDK with any PLX Reference Design Kit (RDK), or a generic device that uses a PLX IC. Users should consult this manual when installing the PCI SDK and for general information.

The PCI SDK has been tested to work with Windows NT 4.0.

1.2 Where To Go From Here

The following is a brief summary of the chapters to help guide your reading of this manual:

Chapter 2, General Information, is an overview of the PCI SDK as well as conventions and terminology used throughout this manual.

Chapter 3, PCI SDK Installation, describes the procedure for installing the PCI SDK software.

Chapter 4, IOP Software, outlines the included IOP applications and provides download instructions for your first IOP application.

Chapter 5, Windows Based Software, outlines the included Windows applications.

Chapter 6, Using The PCI SDK With a New Board, outlines how to use the PCI SDK with new boards.

Chapter 7, PLX Monitor User's Manual, describes the usage of the PLX Monitor application.

1.3 Other PCI SDK Manuals

The PCI SDK includes the following manuals which users should consult for design details:

Programmer's Reference Manual: This manual covers all software design issues regarding the device drivers, Application Programmer's Interface (API) and user applications.

PCI IC Data Sheets & Application Notes (CD-ROM): This CD covers all the functionality of the PCI IC.

2. General Information

2.1 Introduction

The PLX family of embedded bridge ICs bridge the PCI bus to Intel, Power PC processors and other processors. They provide full I₂O compatibility. The PCI SDK provides a powerful IOP API, and Windows NT device drivers that are used to control PLX devices. We are confident that through the use of the PCI SDK, your PLX designs will be brought to market faster and more efficiently.

2.2 Features

The PCI SDK includes the following features:

- A feature based IOP API;
- Board Support Package (BSP) that allows customization of the PCI SDK;
- Two level Back-End Monitor application debugger;
- IOP DMA Resource Manager that supports three modes of operation;
- A PCI API compatible with Windows NT, and device drivers;
- PLXLdr98, A boot loader for downloading IOP applications to a PLX PCI RDK;
- PLXMon97, a Windows application designed to ease the use of the PLX ICs;
- PLXMon98, A Windows GUI application similar to PLXMon97; and,
- PLXTask, a system tray application that launches PLXMon97, PLXMon98, and other PLX applications.

2.3 Terminology

All references to Windows NT assume Windows NT 4.0 or higher and may be denoted as WinNT.

Win32 references are used throughout this manual to mean any application that is compatible with the Windows NT 32 bit environment.

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

3. PCI SDK Installation

3.1 Unpacking

The PCI SDK comes complete with the following items (see Figure 3-1):

- User's Manual (this document);
- Programmer's Reference Manual;
- 1 CD-ROM;
- Registration Letter.

Please take the time now to verify your PCI SDK is complete. If not, please contact Customer Support.

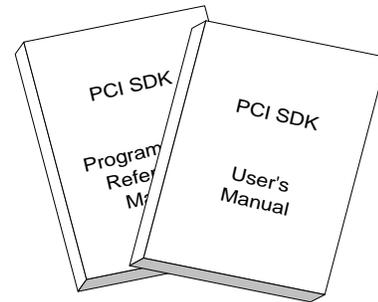


Figure 3-1 Components of the PCI SDK

3.2 Minimum System Requirements

Minimum host system requirements for the PCI SDK are as follows:

- Windows NT 4.0 with Service Pack 3;
- 32MB RAM (when used with only one PLX PCI RDK. Additional memory may be required if more than one PLX PCI RDK are in the system);
- 15MB hard drive space;
- 1 RS 232 port;
- Win32 terminal package;

3.3 PLX Devices Supported

The PCI SDK currently supports the following PLX devices:

- PCI 9080 Rev 3.0

Although the PCI SDK only supports the devices above, the architecture is open to support future PLX devices and more PLX devices will be added in future releases.

3.4 Development Requirements

The PCI SDK was developed for Window NT 4.0 operating system.

PLXMon97 was developed using Microsoft Developer Studio, supplied with the Microsoft Visual C++ 5.0 and the Microsoft Win32 Software Development Kit for Windows NT 3.51 and Windows 95.

The PCI API was developed using Microsoft Developer Studio, supplied with Microsoft Visual C++ 5.0 and the Microsoft Win32 Software Development Kit for Windows NT 3.51 and Windows 95.



The WinNT device driver was developed using the Microsoft Windows NT DDK, version 4.0 and Microsoft Visual C++ 5.0.

3.4.1 Development Tools Needed

Development tools needed for the PCI SDK that are not supplied:

- Microsoft Visual C++ 5.0, with Microsoft Developer Studio;
- Microsoft Win32 Software Development Kit for Windows NT 3.51 and Windows 95;
- Microsoft Windows NT Device Driver Kit, version 4.0;
- Diab Data, Inc. Compiler and Linker for the MPC860, version 4.0b;
- Aisys Ltd. DriveWay-MPC860, version 1.5;
- IBM High C/C++ PowerPC Cross-Compiler, version 1.0 (7/31/96); and,
- 401 EVB Software Support Package, version 1.6.4 (4/1/97).

3.5 Software Installation

3.5.1 WinNT Installation Procedures

To install the PCI SDK Support Software, complete the following:

Note: All previous PCI SDK versions located on the computer must be removed before installing a PCI SDK update. Refer to section 3.5.2 for more details.

1. Insert the CD-ROM into the appropriate CD-ROM drive.
2. Run "D:\Setup.exe" either by typing it at a command prompt or by choosing the Run option of the Start Menu (where "D:" is the drive letter for the CD-ROM Drive). The interactive installation program will install all files.
3. At the "Select Components" dialog box, choose which IOP platform, or platforms, for the desired PLX PCI RDK board software. Currently two choices are available: software for the IBM 401, or for the Motorola MPC860.
4. Reboot the computer.

Note: For proper WinNT installation, the PCI SDK should be installed by a user with "administrator" user rights.

The default installation directory may be changed from default path (C:\Plx\PciSdk) to any drive and path that is desired. This document uses "<INSTALLPATH>" to denote the installation directory.

Take the time now to read the "<INSTALLPATH>\update.txt" for last minute updates to the PCI SDK Support Software documentation and the "<INSTALLPATH>\filelist.txt" to ensure that all the necessary files have been installed properly.

This completes the software installation for WinNT.

3.5.2 Removing All Software

To remove all PCI SDK Software, complete the following:



5. Stop all PLX applications (including the PLXTask application);
6. Open the Windows Control Panel;
7. Double click on the Add/Remove Programs icon in the Control Panel window;
8. Choose the PCI SDK package from the item list; and
9. Click the Add/Remove... button.

Note: This only removes the files that were originally installed by the PCI SDK installation program. For proper removal in WinNT, the PCI SDK should be removed by a user with "administrator" user rights.

This completes the software removal for WinNT.

3.5.3 PCI SDK v2.x Compatibility with PCI SDK v1.2.x

PCI SDK v2.x is NOT compatible with previous PCI SDK versions. The following information is only relevant to users of previous versions. If you are a new user, you may proceed to the next section.

Before using any PCI SDK software, you must upgrade the FLASH and configuration EEPROM images on your PLX RDK. If this step is not done, the PCI SDK will operate unpredictably.

To upgrade your PLXRom image follow the steps below:

- User's of the PCI 9080RDK-401B may use PLXLdr98, as described in section 5.4, to download the "helloworld" sample to the PCI 9080RDK-401B. The image should be programmed at FLASH offset 0x60000.
- User's of the PCI 9080RDK-860 must use a FLASH chip programmer to download the "helloworld" sample to the PCI 9080RDK-860. The image should be programmed at FLASH offset 0x0.

To upgrade your configuration EEPROM follow the steps below:

- User's of the PCI 9080RDK-860 need to reprogram the configuration EEPROM with the settings shown in Figure 3-2.
- User's of the PCI 9080RDK-401B need to reprogram the configuration EEPROM with the settings shown in Figure 3-3.



Serial Eeprom							
PCI Configuration Registers							
Vendor ID (00h)	10B5	Device ID (02h)	0860	PCI Base Address for Local Expansion ROM (30h)	00000000		
Subsystem Vendor ID (2Ch)	10B5	Subsystem ID (2Eh)	9080	Interrupt Line (3Ch)	01	Interrupt Pin (3Dh)	01
Revision (08h)	01	Class Code (09h)	068000	Min_Gnt (3Eh)	00	Max_Lat (3Fh)	00
Local Configuration Registers							
Range for PCI to Local Address Space 0 (00h)	FF000000	Local Base Address for Direct Master to PCI Memory (20h)	40000000				
Remap for PCI to Local Address Space 0 (04h)	10000001	Local Bus Address for Direct Master to PCI IO/CFG (24h)	50000000				
Local Arbitration Register (08h)	0101000C	PCI Base Address (Remap) for Direct Master to PCI (28h)	00000583				
Local Bus Big/Little Endian Descriptor Register (0Ch)	00000024	PCI Config. Addr. Reg. for Direct Master to PCI/CFG (2Ch)	00000000				
Range for PCI to Local Expansion ROM (10h)	00000000	Range for PCI to Local Address Space 1 (1MB) (F0h)	FF000000				
Remap for PCI to Local Expansion ROM (14h)	00000010	Remap for PCI to Local Address Space 1 (F4h)	10000001				
Bus Region Descriptor for PCI to Local Accesses (18h)	02430043	Local Space1 for PCI to Local Accesses (F8h)	00000043				
Range for Direct Master to PCI (1Ch)	FF000000						
Runtime Registers							
Mailbox 0 (User Defined) (78h)	00000000	Mailbox 1 (User Defined) (7Ch)	00000000				
				Close	Write	Refresh	

Figure 3-2 Configuration EEPROM Settings For The PCI 9080RDK-860.

Serial Eeprom							
PCI Configuration Registers							
Vendor ID (00h)	10B5	Device ID (02h)	0401	PCI Base Address for Local Expansion ROM (30h)	00000000		
Subsystem Vendor ID (2Ch)	10B5	Subsystem ID (2Eh)	9080	Interrupt Line (3Ch)	00	Interrupt Pin (3Dh)	01
Revision (08h)	01	Class Code (09h)	068000	Min_Gnt (3Eh)	00	Max_Lat (3Fh)	00
Local Configuration Registers							
Range for PCI to Local Address Space 0 (00h)	FF000000	Local Base Address for Direct Master to PCI Memory (20h)	B8000000				
Remap for PCI to Local Address Space 0 (04h)	00000001	Local Bus Address for Direct Master to PCI IO/CFG (24h)	B0000000				
Local Arbitration Register (08h)	0021001C	PCI Base Address (Remap) for Direct Master to PCI (28h)	00000003				
Local Bus Big/Little Endian Descriptor Register (0Ch)	000000C1	PCI Config. Addr. Reg. for Direct Master to PCI/CFG (2Ch)	00000000				
Range for PCI to Local Expansion ROM (10h)	00000000	Range for PCI to Local Address Space 1 (1MB) (F0h)	FF000000				
Remap for PCI to Local Expansion ROM (14h)	00000000	Remap for PCI to Local Address Space 1 (F4h)	00000001				
Bus Region Descriptor for PCI to Local Accesses (18h)	47400343	Local Space1 for PCI to Local Accesses (F8h)	00000343				
Range for Direct Master to PCI (1Ch)	F8000000						
Runtime Registers							
Mailbox 0 (User Defined) (78h)	00000000	Mailbox 1 (User Defined) (7Ch)	00000000				
				Close	Write	Refresh	

Figure 3-3 Configuration EEPROM Settings For The PCI 9080RDK-401B.

3.5.4 Troubleshooting

If you experience difficulty during the installation of the PCI SDK software please:



- Verify that there is enough hard drive space for all software; and
- Verify that WinNT operated properly before the PCI SDK installation.

Warning: *User may experience difficulties when using the PCI SDK with Windows NT with low memory and multiple PLX PCI RDKs. It is recommended that users increase the amount of available system pages in their System Registry. For more information, refer to section 3.5.4.1.*

3.5.4.1 Increasing Available System Pages In WinNT

To change number of system pages in the System Registry:

10. From a command prompt type: regedt32. This will bring up the Registry Editor window. (This editor looks similar to the Windows Explorer application.)
11. Select the HKEY_LOCAL_MACHINE on Local Machine window from within the Registry Editor.
12. Open the SYSTEM folder.
13. From the SYSTEM folder, open the CurrentControlSet folder.
14. From the CurrentControlSet folder, open the Control folder.
15. From the Control folder, open the Session Manager folder.
16. From the Session Manager folder, open the Memory Management folder.
17. From the Memory Management folder, change the value of the SystemPages key from 0x0 to 0x13880.

If problems persist, please contact Customer Support.

3.5.5 Customer Support

Prior to contacting customer support, please ensure you have the following information:

1. You are situated close to the computer that has the PCI SDK installed;
2. Serial Numbers of the PLX PCI RDKs (if there is any in use with the PCI SDK);
3. Type of processor on the PLX PCI RDK;
4. Operating System version and type; and
5. 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>



You may send email to one of the following addresses:

west-apps@plxtech.com
midwest-apps@plxtech.com
east-apps@plxtech.com
euro-apps@plxtech.com
asia-apps@plxtech.com

4. IOP Software

4.1 Introduction

The PCI SDK includes several samples of IOP applications. Their purpose is to demonstrate how designers can interact with the PCI IC from IOP software. The IOP applications are user-interactive and require a Win32 terminal package or a stand-alone terminal with a serial link.

The IOP applications are designed specifically to run on a PLX PCI RDK. However, the IOP applications can be used as a good starting point for designers using their own hardware device.

4.2 PLXRom Application

All PLX PCI RDKs contain an application that is preprogrammed into the FLASH. The application blinks an LED and prints “Hello World” in a continuous loop. Complete source code for this application is provided in the PCI SDK. For further details, users should refer to the PCI SDK Programmer’s Manual.

By default the Back End Monitor Level 2 (BemL2) module is disabled. The BemL2 module is an advanced debugger similar to PLXMon97. To enable BemL2 type, refer to section 7.3.

4.2.1 MiniRom Application

MiniRom is included in the PCI SDK to provide a good starting point for users who have an untested hardware device and for this reason, it is limited in features and functionality. It provides bare minimum boot up code for most boards. This application configures the microprocessor, the PCI IC, and proceeds to blink the LED that is connected to one of the PCI IC’s USER pins. To use the MiniRom application, you should program the binary image into the FLASH using a FLASH chip programmer. Once the FLASH is programmed, reboot the board and if the LED blinks then the MiniRom application configured the board properly. If this test is successful, the FLASH can be reprogrammed with the PCI SDK PLXRom image (supplied with the PCI SDK).

Note: This ROM application is provided as a bare bones ROM application useful for confirming the functionality of new boards. It does not contain any PCI SDK features that are described in any PCI SDK manual.

4.2.2 PLXRom Communication Configuration

Follow the setup instructions below to configure a Windows compatible terminal package for use with PLXRom:

1. Configure a Windows compatible terminal package with the following settings:
 - 38400 baud;
 - 8 data bits;
 - 1 stop bit;
 - No parity;
 - No flow control; and,
 - Set to the appropriate COM port.



Note: You may also use the 'PLX' icon located on the Windows taskbar to launch a HyperTerminal session that is compatible with your PLX PCI RDK.

2. Connect an RS 232 cable to the serial connector on the PLX PCI RDK and to a free serial communications port on the host computer.

You are now ready to use PLXRom. Figure 3-1 shows the HyperTerminal window when the Back End Monitor Level 2 (BemL2) is enabled. Consult section 7.3 to enable BemL2.

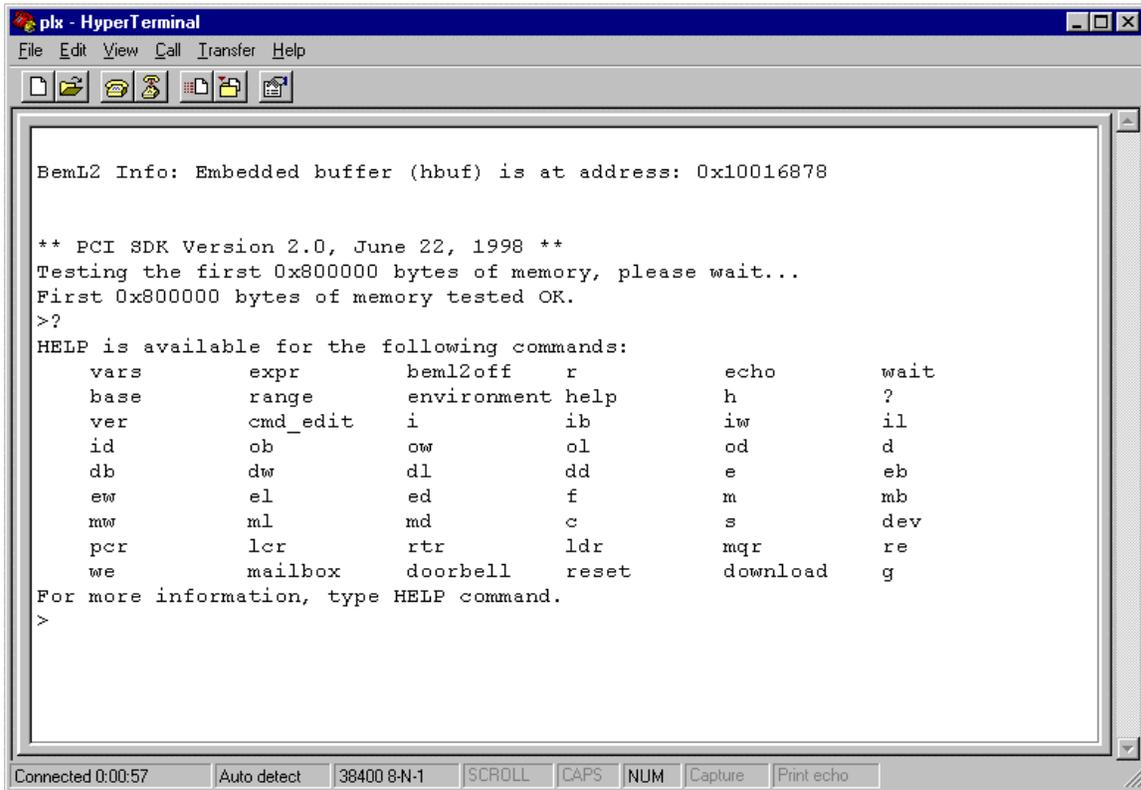


Figure 4-1 PLXRom running within a HyperTerminal session.

4.3 Downloading Procedures

Users should follow the instructions given in section 5.4 to successfully download IOP applications to the PLX PCI RDKs.

5. Windows Based Software

5.1 Introduction

The PCI SDK contains four Windows applications, an API and a device driver. They are as follows:

- PLXMon98, a Graphical User Interface (GUI) application that can be used to monitor and modify PCI IC registers;
- PLXMon97, a command line based application used to monitor and modify PCI IC registers;
- PLXLdr98, a Graphical User Interface (GUI) application used to download software to a PLX PCI RDK;
- PLXTask, a taskbar pop-up that eases launching of PCI SDK applications;
- PCI API, a powerful API compatible with all PLX devices; and
- PLX WinNT Device Driver.

All Win32 executables included in the PCI SDK are located in the “<INSTALLPATH>\bin” directory. Furthermore, this path is added to the environment variables when the PCI SDK is installed.

5.2 PLXMon98

This application is covered in detail in the PLXMon98 User’s Manual.

5.3 PLXMon97

This application is covered in detail in Chapter 6.

5.4 PLXLdr98

PLXLdr98 is a program that is used to download applications to a PLX PCI RDK. Figure 5-1 shows the GUI interface used by PLXLdr98.

Note: Currently PLXLdr98 supports the PCI 9080RDK-401B and the PCI 9080RDK-860 only. Future releases of the PLXLdr98 will support more PLX PCI RDKs as they are available. PLXLdr98 currently supports ELF images for the PCI 9080RDK-401B and COFF images for the PCI 9080RDK-860.

To Use PlxLdr98:

1. You must first select the PLX PCI RDK that you wish to use. By default PLXLdr98 locates all PLX PCI RDKs and displays the first one found.
2. Next, select an image that you wish to use. Images must be in the correct file format (ELF for PLX PCI RDKs with the IBM 401 microprocessor and COFF for PLX PCI RDKs with the Motorola MPC860 microprocessor).
3. Now choose 1 of three options:
 - Click on the ‘Download to RAM’ button to initiate a download to RAM.

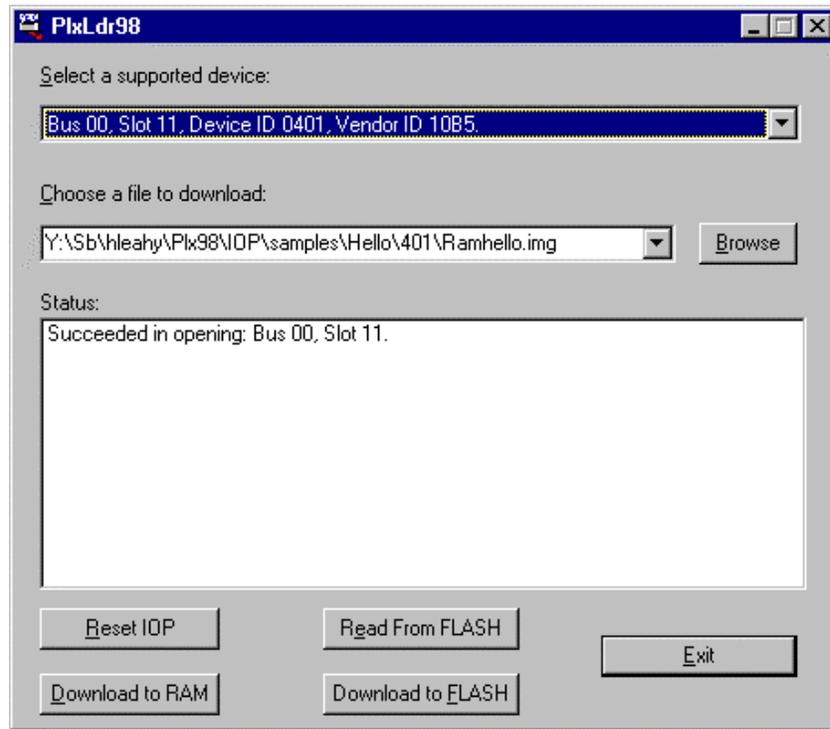


Figure 5-1 PLXLdr98 Download Window

- Click on the ‘*Download to FLASH*’ button to initiate a download to FLASH.
- Click on the ‘*Read From FLASH*’ button to read the FLASH.

Users may also reset the PLX PCI RDK by clicking on the ‘*Reset IOP*’ button.

5.5 PLXTask

PLXTask is a system tray application that can launch the following PCI SDK applications:

- PLXLdr98;
- PLXMon97;
- PLXMon98;
- PLXRom COM1, a HyperTerminal session to use with all PLX PCI RDKs connected to serial communication port 1; and,
- PLXRom COM2, a HyperTerminal session to use with all PLX PCI RDKs connected to serial communication port 2.

Right or left clicking on the popup activates a menu as shown in Figure 5-2. The properties box allows users to customize the path for the executables. The default path is set to the “<INSTALLPATH>\bin” directory.

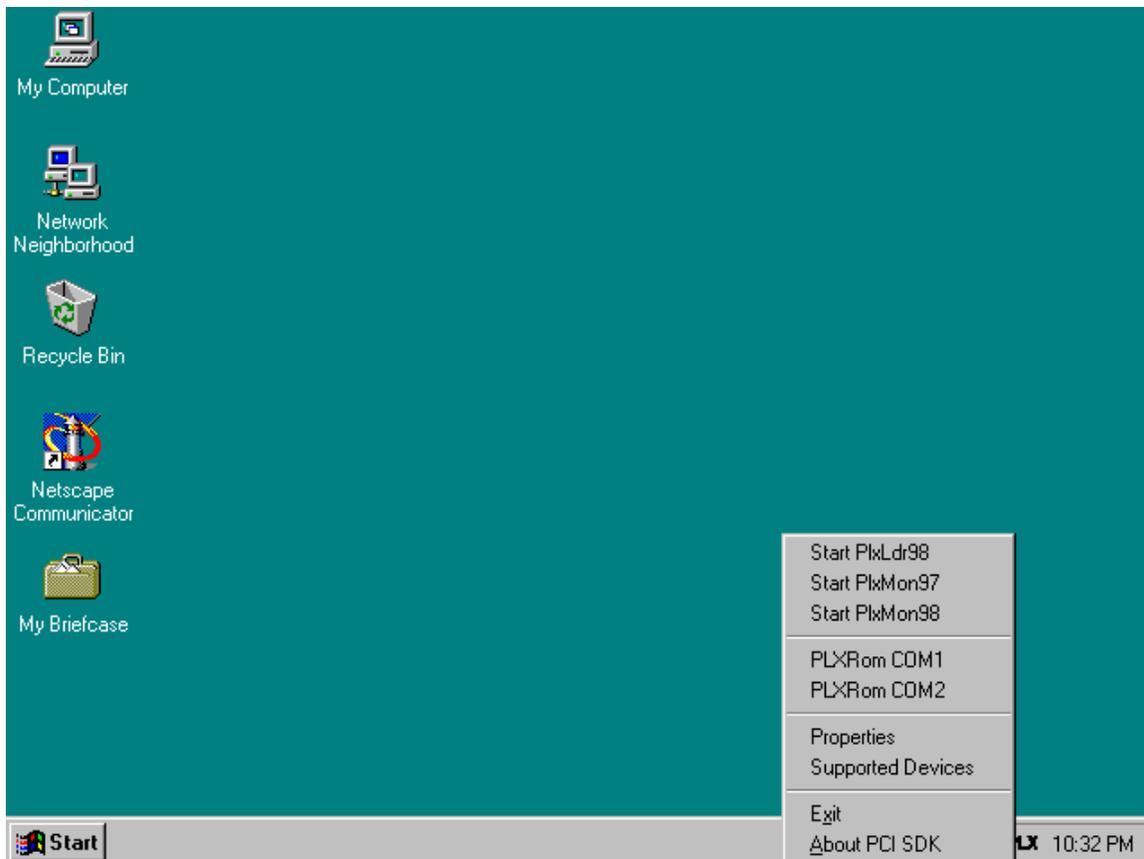


Figure 5-2 Example Taskbar With PLXTask

5.6 Windows NT Device Drivers

The PCI SDK includes Windows NT device drivers for each PLX device. All device drivers are located in the “<WINDOWS SYSTEM DIR>\system32\drivers” directory. The naming convention used for the device drivers is: “Pci<DeviceType>.sys”. For example, the device driver for the PCI9080 device is named “Pci9080.sys”

Consult section 3.3 for a complete list of PLX devices that this PCI SDK supports.

5.6.1 Starting And Stopping

There will be times when you will need to restart the Windows NT device driver. For instance, you must restart the device driver after changing the supported device list.

To restart the Windows NT device driver you should use the Windows NT Control Panel. The Control Panel contains a utility called ‘Devices’ that allows you to start and stop the device driver (see Figure 5-3).

Note: Before stopping the device driver, all PCI SDK applications should be closed.

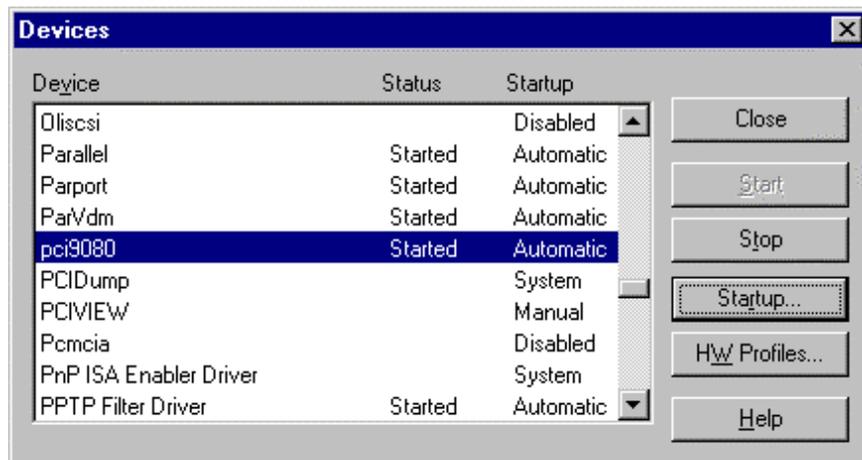


Figure 5-3 The Devices Utility Window.

By default, the device driver is configured to startup automatically at Windows NT boot time. You may configure the device driver to start manually by selecting the 'Startup...' button. However, no PCI SDK applications will function without the device driver being started.

5.6.2 Event Logging

The Windows NT Device Driver has the capability to record errors into the Windows NT Event Viewer. When trouble shooting problems with the device driver it is recommended that the event viewer be used.

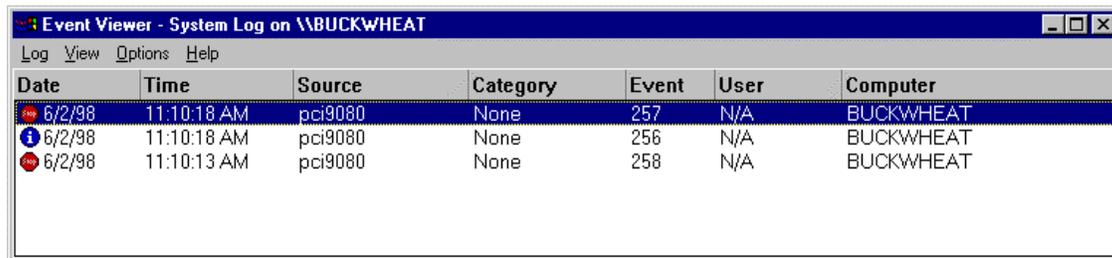


Figure 5-4 The Event Viewer Window.

Events can be viewed by selecting and event item. Figure 5-4 shows an example of the event viewer and Figure 5-5 shows details of an event.

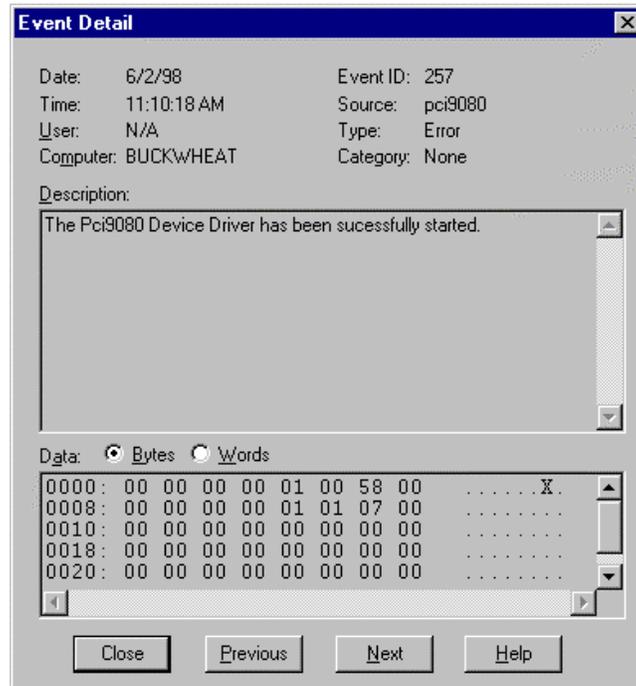


Figure 5-5 The Event Detail Window

5.6.3 Registry Configuration

Every Windows NT device driver requires an entry in the registry editor. The registry editor contains information required by the operating system as well as information required by the device driver. The name in the registry will be the same as the driver name. For instance, the *pci9080.sys* device driver has a *pci9080* registry item as shown in Figure 5-6. All device drivers are located under the “*LocalMachine\System\CurrentControlSet\Services*” tree.

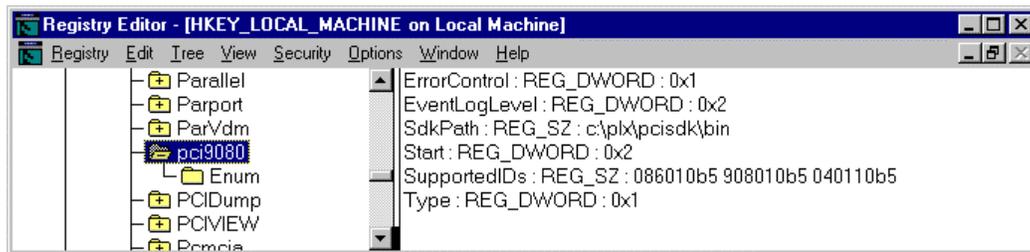


Figure 5-6 The Registry Window

Note: *The registry editor should only be modified by advanced users with administrative rights. Therefore, it is recommended that you do not change any values contained in the registry editor.*



5.7 Adding PCI SDK Support for Custom Boards

The PCI SDK software assumes that a PLX PCI RDK is being used. To help support users that wish to use the PCI SDK with new boards that contain PLX ICs, a mechanism for adding custom devices to the PCI SDK support list is provided with PLXTask (see Figure 5-7). By default the following Reference Design Kits are supported by the PCI SDK:

- PCI 9080RDK-860: Vendor ID is 10B5, Device ID is 0860
- PCI 9080RDK-401B: Vendor ID is 10B5, Device ID is 0401
- Generic PLX Device: Vendor ID is 10B5, Device ID is 9080

Use PLXTask to add new devices to the list of supported devices.

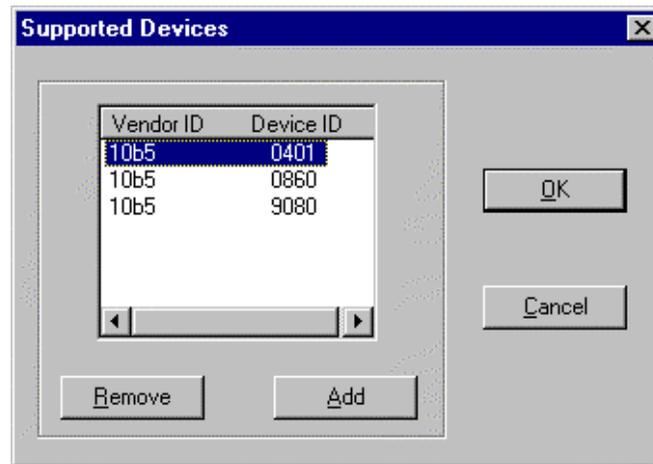


Figure 5-7 PLXTask's Supported Devices Window

To remove a device from the list, highlight the unwanted device and select the *Remove* button. To add new devices, select the *Add* button and enter the new Vendor and Device IDs for the new board (see Figure 5-8). The new board is added to the list of supported devices.



Figure 5-8 PLXTask's New Supported Device Window

Note: The PLX Device Driver must be restarted before the changes will take effect.



6. Using The PCI SDK With a New Board

The following steps can be used as a guide on how to use the PCI SDK with a new board.

6. Program the desired Vendor and Device IDs into the configuration EEPROM.
7. Add these Vendor and Device IDs into the PCI SDK's Support List using PLXTask (see section 5.7 for more information on adding new devices to the PCI SDK's Support List).
8. Edit the MiniRom application as necessary to support the new board.
9. Program the board's FLASH with the modified MiniRom application binary image file.
10. PLXMon98 can now access the board's configuration EEPROM.
11. Using PLXMon98's EEPROM Configuration window, customize the EEPROM settings for the new board and reboot the system for the changes to take effect.
12. Try accessing IOP memory by using the Direct Slave memory accesses to the board.

When the following steps have been performed and are working properly, modify the IOP Board Support Package module to begin porting the PCI SDK to the new board. Consult the PCI SDK Programmer's Manual for more information.



7. PLX Monitor User's Manual

This section covers two applications that are very similar in use. PLXMon97 is a Windows application, and Back-End Monitor Level 2 (BemL2) is an IOP task. Refer to the following sections for more information.

7.1.1 What is PLXMon97?

PLXMon97 is a powerful user-interactive Windows-based program designed for engineers working with the PCI bus, PCI devices, I/O, memory and the PLX family of PCI devices. PLXMon97 uses a simple, yet versatile, command line architecture, and is an extension of the DOS-based PLXMon. PLXMon97 is compatible with Windows NT 4.0.

General PCI, I/O and memory manipulation commands include:

- Select any PCI device on any PCI bus,
- Examine and modify device's PCI Configuration Registers,
- Display, modify, copy, fill, compare and search memory (flat 32 bit addressing),
- Input and output (32 bit addressing),
- And more.

PLX device family commands include:

- Examine and modify the Local Configuration Registers,
- Examine and modify the Run-Time Registers,
- Examine and modify the Local DMA Registers,
- Examine and modify the Messaging Unit Registers,
- Program SGL and Block DMA,
- Read from and write to the Serial EEPROM,
- And more.

Commands are generally short and mnemonic; one line can contain multiple commands.

PLXMon97 supports user-defined macros, user-defined variables and an extensive set of expression evaluation operators.

With PLXMon97's unique repeat command, command lines can be repeated forever or for a user-specified number of iterations. Sections of a command line can be repeated within a command line including sections within sections.

7.1.2 What is Back-End Monitor Level 2?

Back-End Monitor Level 2 is a powerful IOP debug monitor task. It has been designed to be very similar to PLXMon97 application. In most cases, the commands operate in the identical way.



7.1.3 Who should use PLXMon97 and Back-End Monitor Level 2?

PLXMon97 and Back-End Monitor Level 2 are PCI engineering tools that can be useful in identifying PCI related problems. New and experienced PCI users will find both programs helpful when debugging PCI problems. Engineers using the PLX family of PCI interface products will enjoy the command set that is made specifically for PLX ICs and PLX PCI RDKs.

7.1.4 PLX Disclaimer

PLXMon97 and Back-End Monitor Level 2 are engineering tools developed by PLX Technology. PLX makes no warranty about the performance, accuracy or correctness of the program, documentation or source code.

Note: Although PLXMon97 can interact with any PCI device on a PCI bus, it was specifically designed to work best with the PLX PCI ICs. The operation of PLXMon97 with other PCI bridge ICs can be unpredictable. Therefore, caution should be taken when using PLXMon97 with other PCI bridge ICs.

7.2 Getting PLXMon97 Started

PLXMon97 must run on a PC running Windows NT.

1. Install the PCI SDK as instructed in section 3.4.
2. From the command prompt, type `PLXMon97` and press Enter. PLXMon97 does not acknowledge any parameters from the command line. Alternatively, use the icon in the PCI SDK folder in the Start Menu.

Note: You may also use the 'PLX' icon located on the Windows system tray to launch PLXMon97.

3. PLXMon97 starts up. Refer to section 7.4.10 for startup details.
4. PLXMon97 is ready for input. Consult section 7.4 to learn how to use PLXMon97 to its full potential.

7.3 Getting BemL2 Started

BemL2 requires a terminal package to be connected to the serial port of the PLX PCI RDK. Please refer to section 4.2.2 for setup information.

Note: By default BemL2 is disabled. To enable BemL2 you must type '~beml2on' at a HyperTerminal Session Window.

7.4 PLXMon97 and Back-End Monitor Level 2 Basics

7.4.1 Command Line

PLXMon97's and Back-End Monitor Level 2's command line is like DOS, DEBUG and other command line programs. You type in commands then press the Enter key.

The difference between the command line and command lines in other programs is that you can enter as many commands as you wish up to 80 characters. In the case of ambiguity between commands and parameters on a command line, enter a semicolon (;) between the commands.



An ambiguity may arise when a numeric parameter could also be a command. In hexadecimal a digit includes characters zero '0' through 'f' that may be confused with, for example, one of the display commands, 'db'.

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

```
db 100;i 200
```

The semicolon explicitly says that 'i 200' is a new command and not a numeric parameter for the display command (which can accept zero, one or two numeric parameters, see section 7.5.2.1). In this case, however, the semicolon is not needed because 'i' cannot be interpreted as a number. Therefore, entering:

```
db 100 i 200
```

would be interpreted exactly the same as the first example.

The semicolon is required in this example:

```
db 100; db 200
```

since the second 'db' could be interpreted as a numeric parameter for the first command.

7.4.2 Scrolling and Editing Command Lines

Both programs allow you to scroll to previous command lines you have entered, just like DOS' DOSKEY.COM and NDOSEDIT.COM. To scroll, use the up and down arrow keys.

You can edit any line by using the left and right arrow keys, backspace and escape.

PLXMon97 defaults to OVERWRITE mode. Press the INSERT or the INS key to toggle between OVERWRITE and INSERT modes. (Please note, there is no visual indication, such as a changed cursor, to tell you which mode you are in.)

Note: BemL2 is always set to OVERWRITE mode.

7.4.3 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 (=) 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 =
```

You can use virtually any name of any length for your variables. Be careful of duplicate names; commands and macros have priority over variables.

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]
```



PLXMon97 and Back-End Monitor Level 2 scan the built-in command list before the variable list, so if your variable name matches a built-in command, the command will be executed, and your variable will be ignored.

For more information on expressions, see section 7.4.4.

For more information on how to show variables, see section 7.5.6.1.

7.4.4 Expressions

Expressions are legal (and intuitive) arrangements of numbers, variables and operators. (See section 7.4.3 for information on variables.) PLXMon97 and Back-End Monitor Level 2 let 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:

7.4.4.1 Monadic Operators

Monadic operators include one’s complement, two’s complement, etc. as listed in Table 7-1 Monadic Operators (where N refers to a number or variable).

Table 7-1 Monadic Operators (where N refers to a number or variable)

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 the value pointed to by 12345678

7.4.4.2 Dyadic Operators

Dyadic operators include addition, subtraction, etc. as listed in Table 7-2 Dyadic Operators (where N and M refer to numbers and variables).

Table 7-2 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

Syntax	Operation	Example (Hex)	Result
N M	N OR M	7 3	Returns 0x00000007
N^M	N XOR M	7^3	Returns 0x00000004

7.4.5 Macros

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

For more information on creating and using macros, see section 7.5.6.3.

In PLXMon97, macros, as well as variables can be saved to a file, then restored in a future session. See sections 7.5.6.9 and 7.5.6.10 for information on the `save` and `read` commands.

7.4.6 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
```

implies a count of 20 (4020-4000).

```
db 4000 1 20
```

explicitly specifies a count of 20. (Note '1' can be lower-case or upper case.)

```
db 4000 20
```

explicitly specifies a count of 20 too. (Note the count must be smaller than the starting address. Otherwise, use one of the other two range formats).

All three examples yield identical results.

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 starting address and count if you do not provide them.

7.4.7 Repeating Command Lines (Looping)

A unique feature of PLXMon97 and Back-End Monitor Level 2 is the repeat (`r`) command. The `r` command allows you to repeat all or part of a command line for a number of iterations you specify or forever. Further, you can nest your repeat loops using nesting tokens '[' and ']'. See section 7.5.6.5 for more details.

7.4.8 On-line Help

PLXMon97 and Back-End Monitor Level 2 offer built-in help messages for every command. See section 7.5.6.14 for more information.

7.4.9 PLXMon97 'Batch' Files

You can create a text file full of PLXMon97 command lines and pass the file to PLXMon97 for execution in one of two ways;



1. DOS standard-in redirection and
2. PLXMon97's read command (recommended)

7.4.9.1 Command Prompt standard-in redirection

At the command prompt, use command prompts standard-in redirection operator ('<') to submit the file; example:

```
C:\PLX> PLXMON97 < MYFILE.TXT
```

The last command in your file should be the quit command (see section 7.5.6.13).

7.4.9.2 PLXMon97's read command

Use PLXMon97's read command to read and execute your PLXMon97 command file. Refer to section 7.5.6.10 for details.

7.4.9.3 Documenting your file

It is a good idea to add comments to your command file. This is done using PLXMon97's echo command (section 7.5.6.6).

7.4.10 When PLXMon97 Starts

PLXMon97 takes the following actions when it starts:

1. PLXMon97 prints its sign-on banner followed by important release notes.
2. Locates all PCI devices on all PCI buses. Quits with message if no PCI devices are found. This step has a command-line equivalent; see section 7.5.4.1 .
3. Selects a PCI device. Searches registered devices for a PLX PCI RDK. Selects the first PLX device found, if any. User-variables 's0', 's1', 's2', and 's3', are set 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. This step has a command-line equivalent; see section 7.5.4.1.
4. Allocates a common buffer (default size is 0x50000) for the user to transfer data in and out. Sets a user-variable called 'hbuf' to refer to the buffer.
5. Sets a user-variable called 'regbase'. This variable refers to the memory-mapped address of the Local Configuration Registers on the local side. If you are using a PCI 9080 on your own board, you may need to change the value of this variable to match your board's local memory register mapping.
6. Displays list of all devices on all buses showing each device's vendor ID, device ID, and a few selected PCI Configuration Registers. The currently selected device is checked.
7. Displays PLXMon97 prompt (&).
8. PLXMon97 is ready for user input. Type commands, variables or expressions followed by the Enter key.

7.4.11 When The Back-End Monitor Level 2 Starts

The Back-End Monitor Level 2 takes the following actions when it starts:

1. Displays the monitor's prompt (>).
2. The Back-End Monitor is ready for user input. Type commands, variables or expressions followed by the Enter key.

7.5 PLXMon97 and Back-End Monitor Level 2 Command Set

PLXMon97 and Back-End Monitor Level 2 commands are listed by category below.

Parameters that are listed with an “*expr*” substring, such as *valexpr*, can be numbers, variables, or expressions. See sections 7.4.3 and 7.4.4 for more information about variables and expressions.

Parameters listed within square braces (‘[’ and ‘]’) are optional parameters, otherwise the parameter is necessary.

Parameters listed with the vertical bar (‘|’) indicate that you must provide “one or the other” parameter, but not both, as in:

```
command xexpr | yexpr
```

Parameters listed with ellipses (‘...’) indicate that you can extend the command with more parameters of the same type.

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

7.5.1 Starting And Stopping Back-End Monitor Level 2

The Back-End Monitor Level 2 can be started and stopped at any time, as long as it is linked into the application. To start the monitor, send *~beml2on* through the serial port, and send *beml2off* to turn off the monitor.

7.5.2 Memory Manipulation

The memory manipulation commands allow you to display, modify, search and compare memory in several intuitive and generally accepted ways. These commands have counterparts in DOS DEBUG.EXE.

From the user's perspective, all memory is referenced as a flat, non-segmented, 4 gigabyte space. Please note, it is easy to crash your system as there are no restrictions on memory accesses.

Most of the memory manipulation commands accept range parameters. See section 7.4.6 for information on specifying ranges.

Display, enter and move have variations for byte, word and long word sizes. PLXMon97 and BemL2 use the appropriate byte, word or long word assembler instruction (hence the appropriate bus cycle) for each variation.

The *d* and *e* commands adopt the size (byte, word or long word) of the most recently used display or enter command; that is, if you just used the *e1* command, the *d* command will display long words.



7.5.2.1 Display: d, db, dw, dl, dd

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

Range	Memory address and count (see section 7.4.6)
-------	--

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 7.4.6)	displays the specified memory range
One	Address only	display memory starting at the address specified for the default count (0x80 bytes)
Zero		display 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.)

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

e *address* [INC *expr*] [*expr* ...]
 eb *address* [INC *expr*] [*expr* ...]
 ew *address* [INC *expr*] [*expr* ...]
 el *address* [INC *expr*] [*expr* ...]
 ed *address* [INC *expr*] [*expr* ...]

<i>address</i>	Memory address to start entering
<i>incexpr</i>	Address increment
<i>valexpr</i>	Value to enter

Use *enter* 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, PLXMon97 and BemL2 interactively request information. This mode is the *interactive-enter* mode. The interactive-enter mode displays the value at the starting address and allows you 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 key
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 want, for example, 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.

7.5.2.3 Move: m, mb, mw, ml, md

```
m range texpr
mb range texpr
mw range texpr
ml range expr
md range expr
```

range	source address and count (see section 7.4.6)
destexpr	destination address

Use *move* commands to move blocks of memory from one place 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. See the beginning of this section (7.5.1)

Note: Unlike the d and e commands, the m command 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

7.5.2.4 Compare: c

```
c range expr
```

range	left address and count (see section 7.4.6)
addrexp	right address

The *compare* command is useful for checking the result of a move or DMA.

This command performs a byte-by-byte compare of data starting at the left and right address locations. Both the addresses and data bytes of any mismatches are shown.

7.5.2.5 Search: s

```
s range expr [...] | "text"
```

range	search address and count (see section 7.4.6)
expr	first byte of search pattern



...	subsequent bytes of search pattern
"text"	text pattern being searched

Search for a data pattern. 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.

Text must be typed between quote (") characters and the search is case-sensitive.

7.5.3 Input/Output

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.

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

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

range	port address and count (see section 7.4.6)
-------	--

The *input* group of commands reads and displays 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 7.4.6)	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 simply specify the port address (one parameter) as in:

```
iw 100
```

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

PLXMon97 and BemL2 display the ASCII values of the port data in a separate column. This column displays in byte-order, regardless of the size specified (byte, word, long word or double.)

Note: I/O commands performed using BemL2 generate Direct Master I/O cycles to the specified port address.

7.5.3.2 Output: ob, ow, ol, od

```
ob expr(port) expr(value)
ow expr(port) expr(value)
ol expr(port) expr(value)
od expr(port) expr(value)
```

expr(port)	port address
expr(value)	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.)

Note: I/O commands performed using BemL2 generate Direct Master I/O cycles to the specified port address.

7.5.4 PCI Commands

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

7.5.4.1 Show or Select a Device: dev

```
dev [expr]
```

Devnum	Logical device number
--------	-----------------------

With no parameters, this 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(*). Supported Devices are marked with an S (S).

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 device's various local address spaces.

The user-variables 's0', 's1', 's2', 's3', combined with PLXMon97's '+' operator, makes it easy to access the device's memory. For example, to display a portion of the current device's memory at offset 1000, you could use:

```
d s0+1000
```

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

Note: When the dev command is used on BemL2, it only displays PCI devices located on the same bus that the PLX PCI RDK is located. You may not select different devices like you can in PLXMon97.

7.5.4.2 PCI Configuration Registers: pcr

```
pcr [reg expr] [val expr]
```

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.

7.5.5 PLX Device Commands

PLX device commands are designed for use with the PLX family of PCI devices. These commands will not work on non-PLX devices. Devices that are considered PLX devices are marked with an S (S) when the 'dev' command is typed.

7.5.5.1 Local Configuration Registers: lcr

lcr [reg expr] [val expr]

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.

7.5.5.2 Run-Time Registers: rtr

rtr [reg expr] [val expr]

Regnum	register number (long-word aligned)
Regval	register value (long-word)

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

Note: rtr does not prohibit writing to unwritable registers.

7.5.5.3 Local DMA Registers: ldr

ldr [reg expr] [val expr]

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.

7.5.5.4 Messaging Unit Registers: mqr

mqr [reg expr [val expr]

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.

7.5.5.5 Interactive DMA programming: idma

idma

Note: This command is only available on PLXMon97.

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. The command will ask:

- DMA channel (0 or 1)?
- Local to PCI (yes or no)?
- PCI Address?
- Local Address?
- Count?

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

Each question shows the current setting; press the 'Enter' key to accept the setting, or type in a new one followed by Enter. *Idma* does not allow you to go backwards through the questions; press <Ctrl C> to stop PLXMon97, and start again.

The Windows NT architecture enforces the requirement that all PCI memory used for DMA be allocated within the device driver. Allocating memory in an application does not guarantee a contiguous non-paged block of memory and therefore application memory cannot be used for DMA transfers. The device driver that is included with the PCI SDK allocates a common buffer that can be used for DMA purposes. The default buffer size is 0x50000 but there is no guarantee that the driver was able to obtain the complete buffer size. The PLXMon97 variable hbuf is set by the 'dev' command to point to the base address of common DMA buffer in PCI Space.

Kick off the DMA using the dma command (see section 7.5.5.6).

7.5.5.6 DMA Programming: dma

dma



Note: This command is only available on PLXMon97.

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

You can use Interactive DMA programming (section 7.5.5.5) to set up and test a DMA transfer, then use `dma` to repeat the transfer quickly.

7.5.5.7 Read Serial EEPROM: `re`

`re [EEPROM Type] [range]`

EEPROM Type	CS46, or CS56 (defaults to CS46)
Range	destination range (defaults to user-variable 'hbuf', for 0x40 words)

Read selected PCI device's Serial EEPROM into `range` (see section 7.4.6) of PCI memory. (Applies to PLX PCI devices with Serial EEPROMs connected.)

With no parameters, this command reads 0x40 words (0x80 bytes) into PCI memory specified by the 'hbuf' user-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 NMC93CS46 and NMC93CS56 Serial EEPROM devices. Refer to National Semiconductor documentation for device details.

Unlike other commands, the range always specifies words to match the NMC93CS46/CS56 word width.

For more information on writing to the Serial EEPROM, see section 7.5.5.8.

7.5.5.8 Write Serial EEPROM: `we`

`we [EEPROM Type] range`

EEPROM Type	CS46, or CS56 (defaults to CS46)
Range	source range

Write `range` (see section 7.4.6) 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 NMC93CS46 and NMC93CS56 Serial EEPROM devices. Refer to National Semiconductor documentation for device details.

Unlike other commands, the range always specifies words to match the NMC93CS46/CS56 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 NMC93CS46/CS56 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 NMC93CS46/CS56, this command issues a WREN (Write Enable) instruction to the NMC93CS46/CS56. After writing the value, a WDS (Write Disable) instruction is issued.



To read, modify and write the Serial EEPROM, use the Read Serial EEPROM command (see re, section 7.5.5.7) then use the memory manipulation functions followed by this command.

7.5.6 PLXMon97 Internal Commands

7.5.6.1 Show Variables: vars

```
vars
```

Show all variables.

For more information on how to define variables, see section 7.4.3.

7.5.6.2 Expressions: expr

```
expr expression
```

Valexpr	expression to be evaluated
---------	----------------------------

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

Note: expressions can be evaluated without this (or any) command. See section 7.4.4.

7.5.6.3 Define a Macro: define

```
define macro name macro body
```

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. The macro body can include commands, variables and other macros. (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.

PLXMon97 and BemL2 scan the built-in command list before the macro list, 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 about 25 macros at 80 characters per macro.

Macros can be saved to a file for future use. See section 7.5.6.9 for information.

7.5.6.4 Show Macros: macs

```
macs
```

Show all macros. See section 7.5.6.3 for information on defining macros.



7.5.6.5 Repeat: r

r [expr]

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

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

This is one of the niftiest commands. In its simplest application, you append the repeat command to the end of your command line; PLXMon97 and BemL2 will repeat the entire command line over and over forever or until you press a key:

```
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. (An 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:

```
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 if a count is specified.

7.5.6.6 echo

echo "string"

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

Echo the rest of the command line. Use this command to document a PLXMon97 command file. (See section 7.4.9)

7.5.6.7 wait

wait expr

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

Wait while the processor down-counts *countexpr*. The wait loop is uncalibrated; PLXMon97 and BemL2 simply sit in a software loop until the *countexpr* hits zero.

The longest wait is obtained by passing zero (0) as the parameter. This is equivalent to `wait 100000000`.

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

7.5.6.8 base

base [new]

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



Set PLXMon97's radix to a new base. If baseexpr is a number (not a variable or an expression), it must be in base 10. PLXMon97's highest base (and its 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.

Use base with no parameters to determine PLXMon97's current radix.

7.5.6.9 save

```
save [filename]
```

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

Note: This command is only available on PLXMon97.

Save current variables and macros to *filename*. The file is saved as simple, editable text. The variables and macros are saved as PLXMon97 commands; you can edit the file, even add PLXMon97 commands, and then execute the file at any time using the read command (see section 7.5.6.10).

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

7.5.6.10 read

```
read [filename]
```

Filename	name of command file
----------	----------------------

Note: This command is only available on PLXMon97.

Read and execute a PLXMon97 command file. PLXMon97 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*.

The save command saves PLXMon97's macros and variables in a format ready for the read command, see section 7.5.6.9.

7.5.6.11 range

```
range
```

Note: This command is only available on PLXMon97.

This is a placeholder for PLXMon97's on-line help for using ranges. See section 7.4.6 for information on specifying ranges.

7.5.6.12 cmd_edit

```
cmd_edit
```

Note: This command is only available on PLXMon97.

This command is a placeholder for PLXMon97's on-line help for the editing command lines. See section 7.4.2 for information on editing command lines.



7.5.6.13 quit, q

Note: This command is only available on PLXMon97.

Quit PLXMon97.

7.5.6.14 help, h, ?

help [command]

h [command]

? [command]

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

Display on-line help for PLXMon97 and BemL2 commands. With no parameter, these 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 ('[' and ']'). 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.



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.