

PCI SDK User's Manual

Release 1.2, initial publishing December 11, 1997.

Copyright © 1997, 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 Host System Requirements..... | 3-1 |
| 3.3 Software Installation..... | 3-2 |
| 3.3.1 WinNT And Win95 Installation Procedures..... | 3-2 |
| 3.3.2 Removing All Software | 3-3 |
| 3.3.3 Troubleshooting | 3-3 |
| 4. WINDOWS APPLICATIONS | 4-1 |
| 4.1 Introduction | 4-1 |
| 4.2 PLXMon97 | 4-1 |
| 4.3 PlxLdr | 4-1 |
| 4.4 PlxLdr97 | 4-2 |
| 4.5 PlxTask | 4-3 |
| 5. IOP APPLICATIONS | 5-1 |
| 5.1 Introduction | 5-1 |
| 5.2 PLXRom Monitor..... | 5-1 |
| 5.2.1 PLXRom Configuration..... | 5-1 |
| 5.3 Downloading Procedures..... | 5-3 |
| 5.4 Sample IOP Program | 5-3 |
| 6. CUSTOMER SUPPORT | 6-1 |
| 7. PLX MONITOR USER'S MANUAL | 7-1 |
| 7.1.1 What is PLXMon97? | 7-1 |
| 7.1.2 What is PLXRom? | 7-2 |

| | | |
|---------|---|------|
| 7.1.3 | Who should use PLXMon97 and PLXRom? | 7-2 |
| 7.1.4 | PLX Disclaimer | 7-2 |
| 7.2 | Getting PLXMon97 Started..... | 7-2 |
| 7.3 | Getting PLXRom Started..... | 7-2 |
| 7.4 | PLXMon97 and PLXRom Basics..... | 7-3 |
| 7.4.1 | Command Line..... | 7-3 |
| 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 | 7-5 |
| 7.4.5 | Macros..... | 7-5 |
| 7.4.6 | Ranges..... | 7-5 |
| 7.4.7 | Repeating Command Lines (Looping)..... | 7-6 |
| 7.4.8 | On-line Help..... | 7-6 |
| 7.4.9 | PLXMon97 'Batch' Files..... | 7-6 |
| 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 PLXRom Resets | 7-7 |
| 7.5 | PLXMon97 and PLXRom Command Set | 7-8 |
| 7.5.1 | Memory Manipulation | 7-8 |
| 7.5.1.1 | Display: d, db, dw, dl, dd..... | 7-8 |
| 7.5.1.2 | Enter: e, eb, ew, el, ed..... | 7-9 |
| 7.5.1.3 | Move: m, mb, mw, ml, md | 7-10 |
| 7.5.1.4 | Compare: c..... | 7-10 |
| 7.5.1.5 | Search: s..... | 7-10 |
| 7.5.2 | Input/Output..... | 7-11 |
| 7.5.2.1 | Input: i, ib, iw, il, id | 7-11 |
| 7.5.2.2 | Output: ob, ow, ol, od | 7-12 |
| 7.5.3 | PCI Commands | 7-12 |
| 7.5.3.1 | Show or Select a Device: dev | 7-12 |
| 7.5.4 | PLX Device Commands..... | 7-12 |
| 7.5.4.1 | PCI Configuration Registers: pcr..... | 7-13 |
| 7.5.4.2 | Local Configuration Registers: lcr..... | 7-13 |
| 7.5.4.3 | Run-Time Registers: rtr | 7-13 |
| 7.5.4.4 | Local DMA Registers: ldr | 7-13 |
| 7.5.4.5 | Messaging Unit Registers: mqr | 7-14 |

| | | |
|-----------|---|------|
| 7.5.4.6 | Interactive DMA programming: idma | 7-14 |
| 7.5.4.7 | DMA Programming: dma | 7-15 |
| 7.5.4.8 | Read Serial EEPROM: re | 7-15 |
| 7.5.4.9 | Write Serial EEPROM: we | 7-16 |
| 7.5.5 | PLXMon97 Internal Commands | 7-16 |
| 7.5.5.1 | Show Variables: vars | 7-16 |
| 7.5.5.2 | Expressions: expr | 7-16 |
| 7.5.5.3 | Define a Macro: define | 7-16 |
| 7.5.5.4 | Show Macros: macs | 7-17 |
| 7.5.5.5 | Repeat: r | 7-17 |
| 7.5.5.6 | echo | 7-18 |
| 7.5.5.7 | wait | 7-18 |
| 7.5.5.8 | base | 7-18 |
| 7.5.5.9 | save | 7-18 |
| 7.5.5.10 | read | 7-19 |
| 7.5.5.11 | range | 7-19 |
| 7.5.5.12 | cmd_edit | 7-19 |
| 7.5.5.13 | quit, q | 7-19 |
| 7.5.5.14 | help, h, ? | 7-19 |
| 7.5.6 | PLXMon97 Testing Menu | 7-20 |
| 7.5.6.1 | Tests | 7-20 |
| 7.5.6.1.1 | Inbound FIFO Test | 7-20 |
| 7.5.6.1.2 | Outbound FIFO Test | 7-20 |
| 7.5.6.1.3 | Inbound and Outbound FIFO Test | 7-21 |
| 7.5.6.1.4 | Empty Queue Test | 7-22 |
| 7.5.6.1.5 | Memory Test | 7-22 |
| 7.5.6.1.6 | Outbound Overflow Test | 7-22 |

List of Figures

| | |
|---|------|
| Figure 3-1 Components of the PCI SDK..... | 3-1 |
| Figure 4-2 Example window of PlxLdr97..... | 4-2 |
| Figure 4-3 Example task-bar with PlxTask..... | 4-3 |
| Figure 5-1 Example window of PLXRom monitor program (within HyperTerminal)..... | 5-3 |
| Figure 7-1 Inbound FIFO Test..... | 7-20 |
| Figure 7-2 Outbound FIFO Test..... | 7-21 |
| Figure 7-3 Outbound FIFO Test..... | 7-22 |

List of Tables

| | |
|--|-----|
| Table 7-1 Monadic Operators (where N refers to a number or variable)..... | 7-4 |
| Table 7-1 Dyadic Operators (where N and M refer to numbers and variables)..... | 7-5 |



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) 1997 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. 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 either the IBM 401 Reference Design Kit (PCI 9080RDK-401), the Intel 960 Reference Design Kit (PCI 9080RDK-960), the Motorola MPC860 Reference Design Kit (PCI 9080RDK-860), or a generic device that uses the PCI 9080 chip. 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 and Windows 95.

Please contact PLX's Customer Support for information on upgrading to the PCI 9080.

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, Windows Applications, outlines the included Windows applications.

Chapter 5, IOP Applications, outlines the included IOP applications and provides download instructions for your first IOP application.

Chapter 6, Customer Support, provides PCI SDK customer support contact information.

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 9080 Data Sheets & Application Notes (CD-ROM): This CD covers all the functionality of the PCI 9080.

2. General Information

2.1 Introduction

The PCI 9080 is a powerful embedded chip that bridges the PCI bus to Intel and Power PC processors. It provides full I₂O compatibility and is upward compatible with the PCI 9060/9060ES/9060SD. The PCI SDK provides a powerful API, and device drivers for Windows NT and Windows 95 that are used to control the PCI 9080 device. We are confident that through the use of the PCI SDK, your PCI 9080 designs will be brought to market faster and more efficiently.

2.2 Features

The PCI SDK includes the following features:

- An API compatible with Windows NT, Windows 95 and IOP applications including source code;
- Device Drivers for both Windows NT and Windows 95 including source code;
- Boot loader for downloading IOP applications to a PLX Reference Design Kit (either PCI 9080RDK-401, PCI 9080RDK-960, or PCI 9080RDK-860); and
- PLXMon97, a Windows application designed to ease the use of the PCI 9080 chip.

2.3 Terminology

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

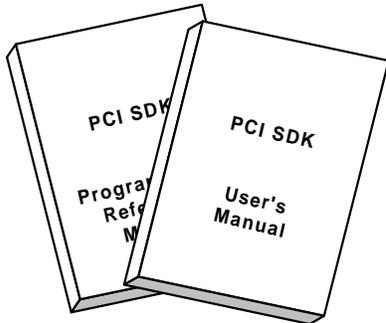
Win32 references are used throughout this manual to mean any application that is compatible with either Windows NT or Windows 95 32 bit environment.

All references to IOP (I/O Platform) throughout this manual denotes the embedded hardware and all references to IOP software denotes 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;
- Five Installation diskettes;
- Registration Letter.

Please take the time now to verify your PCI SDK is complete. If not, please contact Customer Support by phone at (408)774-9060.



Figure 3-1 Components of the PCI SDK

3.2 Minimum Host System Requirements

Minimum host system requirements for the PCI SDK when using the IBM 401 IOP device:

- Windows NT 4.0 or Windows 95;
- 16MB RAM;
- 13MB hard drive space;
- 1 RS 232 port;
- Win32 terminal package;
- IBM ELF PowerPC Development Suite (required to recompile source code);
- Windows NT 4.0 or Windows 95 DDK (required to recompile source code);
- Windows NT 4.0 or Windows 95 SDK (required to recompile source code);
- Microsoft Visual C++ 4.2 (required to recompile source code).

Minimum host system requirements for the PCI SDK when using the i960HA IOP device:



- Windows NT 4.0 or Windows 95;
- 16MB RAM;
- 13MB hard drive space;
- 1 RS 232 port;
- Win32 terminal package;
- Intel i960 Processor Development Tools (required to recompile source code);
- Windows NT 4.0 or Windows 95 DDK (required to recompile source code);
- Windows NT 4.0 or Windows 95 SDK (required to recompile source code);
- Microsoft Visual C++ 4.2 (required to recompile source code).

Minimum host system requirements for the PCI SDK when using the MPC860 IOP device:

- Windows NT 4.0 or Windows 95;
- 16MB RAM;
- 13MB hard drive space;
- 1 RS 232 port;
- Win32 terminal package;
- DriveWay-860 development tools (required to recompile source code);
- Diab C Compiler for the MPC 860 (required to recompile source code);
- Windows NT 4.0 or Windows 95 DDK (required to recompile source code);
- Windows NT 4.0 or Windows 95 SDK (required to recompile source code);
- Microsoft Visual C++ 4.2 (required to recompile source code).

3.3 Software Installation

3.3.1 WinNT And Win95 Installation Procedures

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

- Insert the first PCI SDK diskette into the appropriate floppy drive.
- Run “a:\setup.exe” either by typing it at a command prompt or by choosing the *Run* option of the Start Menu. The interactive installation program will install all files.
- At the “Select Components” dialog box, choose which IOP platform, or platforms, you wish to support. Currently four choices are available: IBM 401, Intel i960HA, Motorola MPC 860, and a generic platform.
- 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 “c:\Plx\PciSdk” to any drive and path that you desire. This document uses “<INSTALLPATH>” to abbreviate the installation directory.

It is recommended that you remove any previous PCI SDK versions located on your computer before installing a PCI SDK update. Refer to Section 3.3.2 for more details.

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 Windows NT and Windows 95.

3.3.2 Removing All Software

To remove all PCI SDK Software, complete the following:

- Open the Windows Control Panel;
- Double click on the Add/Remove Programs icon in the Control Panel window;
- Choose the PCI SDK package from the item list; and
- 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 and Win95.

3.3.3 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 and/or Win95 operated properly before the PCI SDK installation.

If problem’s persist, please contact Customer Support.

4. Windows Applications

4.1 Introduction

The PCI SDK contains four Windows applications. They are as follows:

- PLXMon97, an application used to monitor and modify PCI 9080 registers;
- PlxLdr, an application used to download IOP software to an PCI 9080RDK;
- PlxLdr97, a GUI application used to download software to an PCI 9080RDK; and
- PlxTask, a taskbar pop-up that eases launching of PCI SDK applications.

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.

4.2 PLXMon97

This application is covered in detail in Chapter 7.

4.3 PlxLdr

PlxLdr has been designed to download applications only to the PCI 9080RDK-401, PCI 9080RDK-960, and PCI 9080RDK-860 evaluation boards. Complete source code is provided in this SDK and designers are encouraged to modify the software to function with their own hardware. The PCI SDK Programmer’s Reference Manual has design information that will be relevant to designers who wish to modify PlxLdr.

PlxLdr has the following syntax options:

- Use ‘PlxLdr <-401> | <-960> | <-860> <filename>’ to download an application;
- Use ‘PlxLdr <-401r> | <-960r> | <-860r>’ to reset the PCI 9080RDK;
- Use ‘PlxLdr <-960> | <-860> -f <offset> <filename>’ to download a new FLASH image (see warning below). *offset* should be 0x60000 for the PCI 9080RDK-960 board and *offset* should be 0x0 for the PCI 9080RDK-860 board;
- Use ‘PlxLdr <-960> | <-860> -fr <offset> <filename> ‘ to read the FLASH contents and store it to a file;

Warning: *The -f option will reprogram the FLASH. This will erase PLXRom and the PCI 9080RDK may not function afterwards. This command should only be used by advanced users. If the PCI 9080RDK does not operate correctly after downloading to FLASH, reprogram the FLASH with the original PLXRom Monitor provided in the PCI SDK. It is important to do this before rebooting the computer. If all else fails, remove the FLASH and reprogram it using a device programmer. Your computer will not boot without a valid FLASH program that sets the Local Init bit of the PCI 9080 chip. (Refer to the PCI 9080 Data Sheets and PCI SDK Programmer’s Reference Manual for more information on setting the Local Init bit.)*



PlxLdr currently supports ELF images for the PCI 9080RDK-401 and COFF images for the PCI 9080RDK-960 and PCI 9080RDK-860.

4.4 PlxLdr97

PlxLdr97 is a GUI version of PlxLdr. Currently PlxLdr97 supports the PCI 9080RDK-401, PCI 9080RDK-960 and PCI 9080RDK-860 evaluation boards. Figure 4-2 shows the GUI interface used by PlxLdr97.

1. Select the PCI 9080RDK that you wish to download to. By default PlxLdr97 locates all PCI 9080RDKs and displays the first one found.
2. Select the image that you wish to download. Images must be in the correct file format (ELF for the 9080RDK-401, and COFF for the 9080RDK-960 and 9080RDK-860).
3. Click on the 'Download IOP' button to initiate the download. Users may also reset the PCI 9080RDK by clicking on the 'Reset IOP' button.

Status messages are printed in a status window.

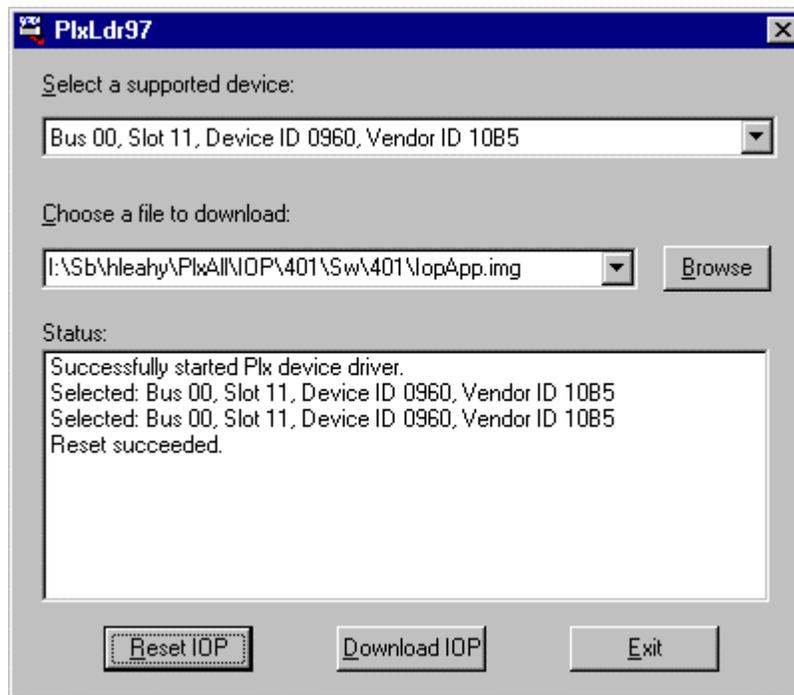


Figure 4-2 Example window of PlxLdr97.

Presently, PlxLdr97 does not support any FLASH capabilities. Users should refer to Section 4.3 to reprogram the FLASH.

4.5 PlxTask

PlxTask is a task-bar popup program that can launch the following PCI SDK applications:

- PlxLdr97;
- PLXMon97;
- PLXRom-401, a HyperTerminal session to use with the PCI 9080RDK-401 evaluation board;
- PLXRom-960, a HyperTerminal session to use with the PCI 9080RDK-960 evaluation board; and,
- PLXRom-860, a HyperTerminal session to use with the PCI 9080RDK-860 evaluation board.

Note: All HyperTerminal sessions assume COM2 is being used for the serial link.

Right-clicking on the popup activates a menu as shown below. The properties box allows users to customize the path for PlxLdr97 and PLXMon97. The default path is set to the “<INSTALLPATH>\bin” directory.

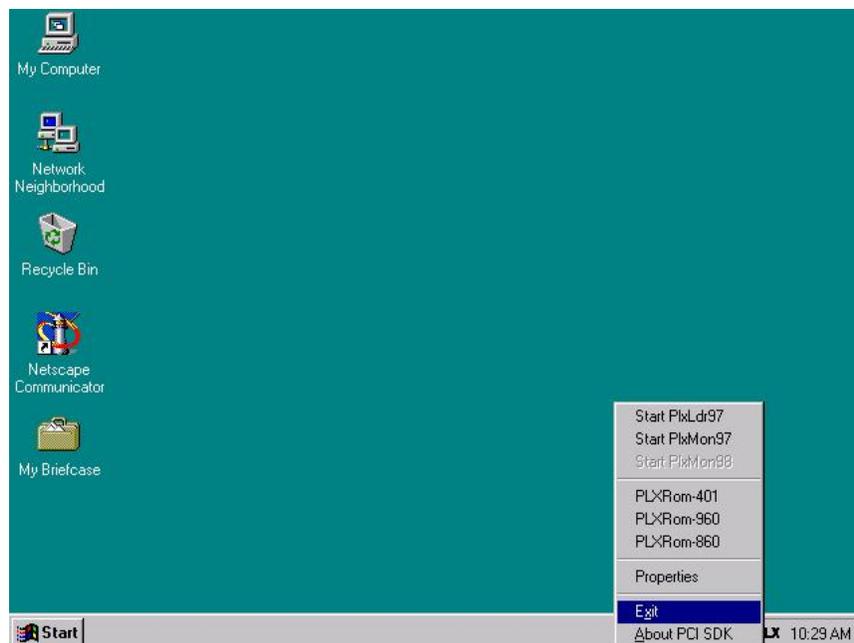


Figure 4-3 Example task-bar with PlxTask.

5. IOP Applications

5.1 Introduction

The PCI SDK includes general purpose IOP applications and several samples. Their purpose is to demonstrate how designers can interact with the PCI 9080 chip 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 be used with a PLX Reference Design Kit (RDK). Presently, the PCI SDK supports three PCI 9080RDKs: PCI 9080RDK-960, PCI 9080RDK-401, or PCI 9080RDK-860. However, the IOP applications can be used as a good starting point for designers using their own hardware device.

5.2 PLXRom Monitor

All PCI 9080RDKs contain a debug monitor program located in on-board FLASH. Upon reset, the microprocessor executes the monitor software. IOP Applications written by users can be downloaded into on-board RAM and PLXRom can be instructed to “jump” to the downloaded code segment. A debug menu is available using a serial link. Please proceed to Section 5.2.1 for serial setup information.

PLXRom’s command set is very similar to PLXMon97. Users should refer to Chapter 7 for more information.

5.2.1 PLXRom 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:
 - 115200 baud (PCI 9080RDK-401, and PCI 9080RDK-960);
 - 38400 baud (PCI 9080RDK-860);
 - 8 data bits;
 - 1 stop bit;
 - No parity;
 - Xon/Xoff flow control (PCI 9080RDK-401);
 - No flow control (PCI 9080RDK-960, and PCI 9080RDK-860); and,
 - Set to the appropriate COMM port.

Note: You may also use the ‘PLX’ icon located on the Windows taskbar to launch a HyperTerminal session that is compatible with your PCI 9080RDK.



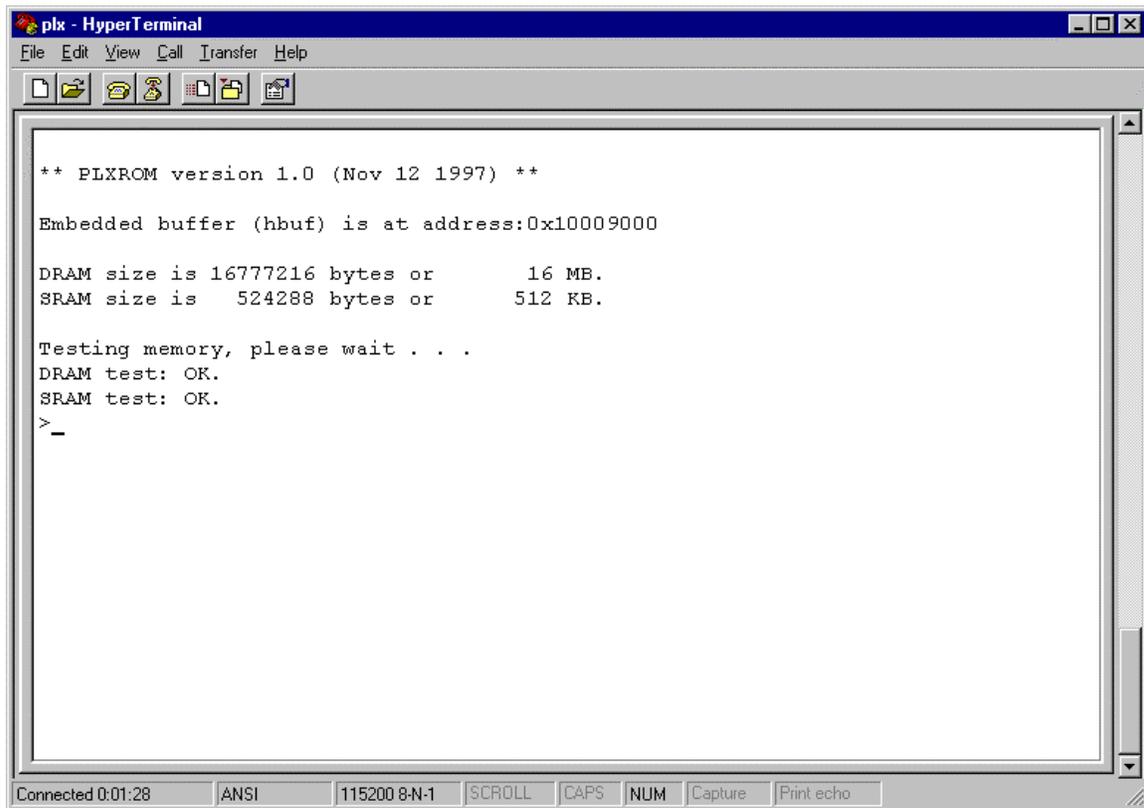
2. Connect a RS 232 cable to the serial connector on the PCI 9080RDK and to a free serial communications port on the host computer.

You are now ready to use PLXRom.

```
plx - HyperTerminal
File Edit View Call Transfer Help
** PLXRom version 1.0 (Nov 12 1997) **
Embedded buffer (hbuf) is at address:0x10009000
DRAM size is 16777216 bytes or      16 MB.
SRAM size is  524288 bytes or      512 KB.
Testing memory, please wait . . .
DRAM test: OK.
SRAM test: OK.
>
_
```

Connected 0:01:28 ANSI 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

Figure 5-1 shows the PLXRom monitor program window after reset, and before an IOP application is downloaded to the IOP processor.



```
plx - HyperTerminal
File Edit View Call Transfer Help
** PLXRom version 1.0 (Nov 12 1997) **
Embedded buffer (hbuf) is at address:0x10009000
DRAM size is 16777216 bytes or      16 MB.
SRAM size is  524288 bytes or      512 KB.
Testing memory, please wait . . .
DRAM test: OK.
SRAM test: OK.
>
_
```

Connected 0:01:28 ANSI 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

Figure 5-1 Example window of PLXRom monitor program (within HyperTerminal).

5.3 Downloading Procedures

Users should follow the instructions given in Section 4.3 or Section 4.4 to successfully download IOP applications.

5.4 Sample IOP Program

The IOP application supplied with the PCI SDK has the following features and functionality:

- Performs reads and writes to all registers of the PCI 9080 chip;
- Performs Direct Master reads and writes to PCI memory space;
- Performs Direct Master reads and writes to PCI I/O space;
- Sets up DMA transfers between system memory and local memory;
- Display PCI 9080 register group values;
- Accesses any location of memory on the PCI 9080RDK;
- Tests PCI memory accesses, tests local memory, and fills local memory with either fixed values or incremental values;



- Performs reads and writes to the on-board Serial EEPROM; and,
- Services interrupts for:
 - DMA termination
 - Message FIFO interrupts
 - Doorbell interrupts



6. 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 Number of the PCI 9080RDK;
3. Type of processor on the PCI 9080RDK;
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>



7. PLX Monitor User's Manual

This section covers two applications that are very similar in use. PLXMon97 is a Windows application, and PLXRom is an IOP application. 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 both Windows 95 and 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 chained and non-chained 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 PLXRom?

PLXRom is a powerful IOP debug monitor application. It has been designed to be very similar to PLX's popular PLXMon97 application. In most cases the commands operate in the identical way.

Three versions of PLXRom currently exist (PLXRom-401, PLXRom-960, and PLXRom-860). Each version is designed to work with its respective PCI 9080RDK.

7.1.3 Who should use PLXMon97 and PLXRom?

PLXMon97 and PLXRom are PCI engineering tools that can be useful in identifying PCI related problems. New and experienced PCI users will find both programs to be 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 chips and PCI 9080RDKs.

7.1.4 PLX Disclaimer

PLXMon97 and PLXRom 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 9080 chip. The operation of PLXMon97 with other PCI bridge chips can be unpredictable. Therefore, caution should be taken when using PLXMon97 with other PCI bridge chips.

7.2 Getting PLXMon97 Started

PLXMon97 must run on a PC running Windows NT or Windows 95.

1. Install the PCI SDK as instructed in Section 3.3.
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 taskbar 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 PLXRom Started

PLXRom requires a terminal package to be connected to the serial port of the PCI 9080RDK. Please refer to section 5.2.1 for setup information.



7.4 PLXMon97 and PLXRom Basics

7.4.1 Command Line

PLXMon97's and PLXRom'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' which 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 7.5.1.1). 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: PLXRom 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 PLXRom 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 7.5.5.1.

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 PLXRom 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).

| Syntax | Operation | Example (Hex) | Result |
|--------|--------------|---------------|---|
| ~N | Inverse of N | ~1 | Returns 0xFFFFFFFFE (One's complement) |
| -N | 0 minus N | -1 | Returns 0xFFFFFFFFF (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 |

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

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).

| 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 7-2 Dyadic Operators (where N and M refer to numbers and variables)

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 over and over again.

For more information on creating and using macros, see section 7.5.5.3.

In PLXMon97, macros, as well as variables can be saved to a file, then restored in a future session. See sections 7.5.5.9 and 7.5.5.10 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 PLXRom 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 7.5.5.5 for more details.

7.4.8 On-line Help

PLXMon97 and PLXRom offer built-in help messages for every command. See section 7.5.5.14 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 7.5.5.13).

7.4.9.2 PLXMon97's read command

Use PLXMon97's `read` command to read and execute your PLXMon97 command file. Refer to section 7.5.5.10 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 7.5.5.6).

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 7.5.3.1 .
3. Selects a PCI device. Searches registered devices for PLX's vendor ID (0x10b5). 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 7.5.3.1.
4. Allocates a 64KB buffer 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. Sets a user-variable called 'membase'. This variable refers to an address in the local memory that is available to load chained DMA descriptors. If you are using a PCI 9080 on your own board, and you want to load chained DMA descriptors from the PCI side, you may need to change the value of this variable to match your board's local memory map.
7. 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.
8. Displays PLXMon97 prompt (&).
9. PLXMon97 is ready for user input. Type commands, variables or expressions followed by the Enter key.

7.4.11 When PLXRom Resets

PLXRom takes the following actions when it starts:

1. PLXRom prints its sign-on banner followed by the release date.
2. Allocates a 64KB buffer for the user to transfer data in and out. Sets a user-variable called 'hbuf' to refer to the buffer.
3. The memory is scanned to determine the size of memory. DRAM and SRAM are scanned.
4. The memory is tested and all unused addresses are cleared to zero
5. Displays PLXRom prompt (>).
6. PLXRom is ready for user input. Type commands, variables or expressions followed by the Enter key.



7.5 PLXMon97 and PLXRom Command Set

PLXMon97 and PLXRom 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 (‘...’) indicates that you can extend the command with more parameters of the same type.

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

7.5.1 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 PLXRom 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.

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

```
d [range]
```

```
db [range]
```

```
dw [range]
```

```
dl [range]
```

```
dd [range]
```

| | |
|-------|--|
| Range | memory address and count (see section 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.)

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

```
e  addrexpr [INC expr] [expr ...]
eb addrexpr [INC expr] [expr ...]
ew addrexpr [INC expr] [expr ...]
el addrexpr [INC expr] [expr ...]
ed addrexpr [INC expr] [expr ...]
```

| | |
|----------|----------------------------------|
| addrexpr | memory address to start entering |
| incexpr | address increment |
| valexpr | 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 PLXRom 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, for example, want to enter a new values every 100 bytes starting at your starting address:

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

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



7.5.1.3 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

7.5.1.4 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.

7.5.1.5 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.

7.5.2 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.

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

```

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

```

| | |
|-------|--|
| range | port address and count (see section 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 PLXRom 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 PLXRom generate Direct Master I/O cycles to the specified port address.



7.5.2.2 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 PLXRom generate Direct Master I/O cycles to the specified port address.

7.5.3 PCI Commands

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

7.5.3.1 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(*).

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 PLXRom, it only displays PCI devices located on the same bus that the PCI 9080RDK is located. You may not select different devices like you can in PLXMon97.

7.5.4 PLX Device Commands

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

7.5.4.1 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.

7.5.4.2 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.

7.5.4.3 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.

7.5.4.4 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.



7.5.4.5 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.

7.5.4.6 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)?
- Chained mode (yes or no)?
- Local to PCI (yes or no)?
- PCI Address?
- Local Address?
- Count?

If you choose chained mode, you will also be asked:

- Descriptor in PCI space (yes or no)?
- Descriptor address?
- End of chain (yes or no)?

(*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.

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. Kick off the DMA using the *dma* command (see section 7.5.4.7).

7.5.4.7 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 chain descriptors, mode, address and count registers are used 'as-is'.

You can use Interactive DMA programming (section 7.5.4.6) to set up and test a DMA transfer, then use *dma* to repeat the transfer quickly.

7.5.4.8 Read Serial EEPROM: *re*

re [*range*]

| | |
|--------------|--|
| <i>range</i> | 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 commands 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 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 7.5.4.9.



7.5.4.9 Write Serial EEPROM: we

we range

| | |
|-------|--------------|
| 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 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 the Read Serial EEPROM command (see re, section 7.5.4.8) then use the memory manipulation functions followed by this command.

7.5.5 PLXMon97 Internal Commands

7.5.5.1 Show Variables: vars

vars

Show all variables.

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

7.5.5.2 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.

7.5.5.3 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 PLXRom 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 7.5.5.9 for information.

7.5.5.4 Show Macros: macs

macs

Show all macros. See section 7.5.5.3 for information on defining macros.

7.5.5.5 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 PLXRom 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.

7.5.5.6 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)

7.5.5.7 wait

wait expr

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

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

7.5.5.8 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.

7.5.5.9 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, then execute the file at any time using the read command (see section 7.5.5.10).



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

7.5.5.10 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 7.5.5.9.

7.5.5.11 range

`range`

Note: This command is only available on PLXMon97.

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

7.5.5.12 cmd_edit

`cmd_edit`

Note: This command is only available on PLXMon97.

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

7.5.5.13 quit, q

Note: This command is only available on PLXMon97.

Quit PLXMon97.

7.5.5.14 help, h, ?

`help [command]`

`h [command]`

`? [command]`

| | |
|---------|--------------|
| command | command name |
|---------|--------------|

Display on-line help for PLXMon97 and PLXRom 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 an parameter within square braces ('[' and ']'). Such a parameter is optional for the command.

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

7.5.6 PLXMon97 Testing Menu

7.5.6.1 Tests

Note: This command is only available on PLXMon97.

PLXMon97 includes several tests that allow users to test board functionality. Most of the tests can only be used with the PCI 9080 device because they test the messaging FIFO's. All tests require that an IOP image file be downloaded before the test can be run. There are two image files required and each one is noted below when it is used.

7.5.6.1.1 Inbound FIFO Test

The Inbound FIFO test continuously posts and frees messages using the Inbound FIFO's as described below (also refer to Figure 7-1 Inbound FIFO Test). This test requires the test1_2 image file to be downloaded before the test is conducted.

1. Host: Post a message to the Inbound Port;
2. IOP: Retrieve a message from Inbound Port;
3. IOP: Free message by posting to Inbound Port;
4. Host: Get an empty message frame from Inbound Port;
5. Host: Compare with what was previously posted. Start test again if compare it valid.

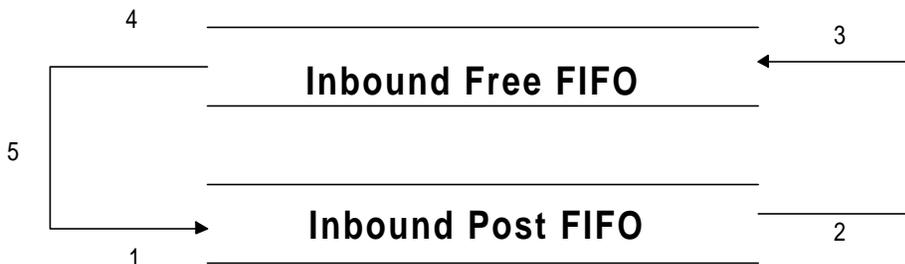


Figure 7-1 Inbound FIFO Test

7.5.6.1.2 Outbound FIFO Test

The Outbound FIFO test continuously posts and frees messages using the Outbound FIFO's as described below (also refer to

Figure 7-2). This test requires the test1_2 image file to be downloaded before the test is conducted.

1. Host: Free a message frame by posting to the Outbound Port;
2. IOP: Get an empty message frame from Outbound Port;
3. IOP: Post a message by posting to Outbound Port;
4. Host: retrieve a message from Outbound Port;
5. Host: Compare with what was previously posted. Start test again if compare it valid.

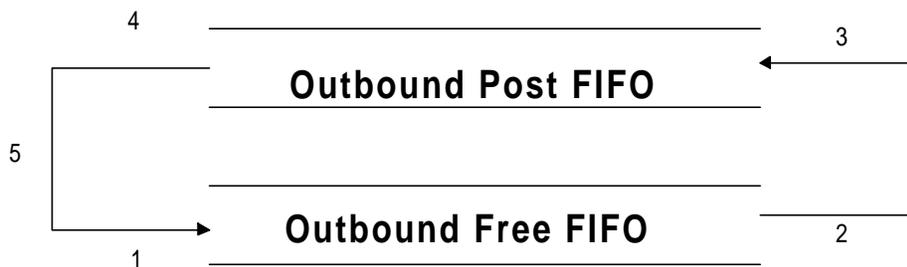


Figure 7-2 Outbound FIFO Test

7.5.6.1.3 Inbound and Outbound FIFO Test

The Inbound and Outbound FIFO test continuously posts and frees messages using both FIFO's as described below (also refer to

Figure 7-1). This test requires the test3 image file to be downloaded before the test is conducted.

1. Host: Post a message by posting to the Inbound Port;
2. IOP: Retrieve a message from the Inbound Port;
3. IOP: Post a message by posting to Outbound Port;
4. Host: retrieve a message from Outbound Port;
5. Host: Compare with what was previously posted. Continue if compare is valid;
6. Host: Free a message frame by posting to the Outbound Port;
7. IOP: Get an empty message frame from the Outbound Port;
8. IOP: Post a message by posting to the Inbound Port;
9. Host: Retrieve a message fro the Inbound Port;
10. Host: Compare with what was previously posted. Continue if compare is valid;

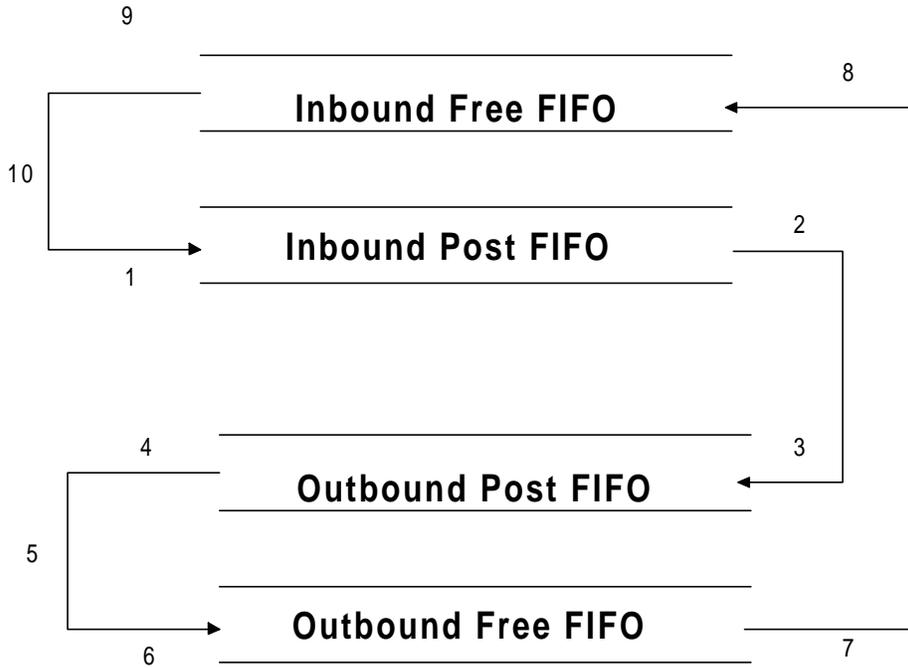


Figure 7-3 Outbound FIFO Test

7.5.6.1.4 Empty Queue Test

The Empty Queue Test tests the PCI 9080 empty FIFO status by reading two messages from the Inbound Port. If this test passes the first read should return a valid message and the second read should return 0xFFFFFFFF.

7.5.6.1.5 Memory Test

The Memory Test tests Local Address Space 0 memory cycles. It continuously tests memory access until an error occurs by doing 32-bit read and writes.

7.5.6.1.6 Outbound Overflow Test

The Outbound Overflow Test tests the PCI 9080 overflow status by continuously posting messages to the Outbound Port. An IOP interrupt should occur when the Outbound FIFO overflows.



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.