



***i*SPAN™ 4539 PMC® Quad T1/E1/J1
Communication Controller
Built-In Self Tests and Monitor
Manual**

Document No. UG04539-04-00-B01-1

Release Date: June 2002

Copyright Notice

© 2002 by Interphase Corporation. All rights reserved.

Printed in the United States of America, 2002.

This manual is licensed by Interphase to the user for internal use only and is protected by copyright. The user is authorized to download and print a copy of this manual if the user has purchased one or more of the Interphase products described herein. All copies of this manual shall include the copyright notice contained herein. No part of this manual, whether modified or not, may be incorporated into user's documentation without prior written approval of

Interphase Corporation
13800 Senlac
Dallas, Texas 75234

Phone: (214) 654-5000
Fax: (214) 654-5506

Disclaimer

Information in this manual supersedes any preliminary specifications, preliminary data sheets, and prior versions of this manual. While every effort has been made to ensure the accuracy of this manual, Interphase Corporation assumes no liability resulting from omissions, or from the use of information obtained from this manual. Interphase Corporation reserves the right to revise this manual without obligation to notify any person of such revision. Information available after the printing of this manual will be in one or more Read Me First documents. Each product shipment includes all current Read Me First documents. All current Read Me First documents are also available on our web site.

THIS MANUAL IS PROVIDED "AS IS." INTERPHASE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING THOSE OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL INTERPHASE BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Trademark Acknowledgments

Interphase® and the Interphase logo are registered trademarks, (*i*)chip™, SynWatch™, FibreView™, ENTIA™, PowerSAN™, SlotOptomizer™, *i*WARE™, *i*NAV™, and *i*SPAN™ are trademarks of Interphase Corporation.

All other trademarks are the property of their respective owners.

Assistance

Product Purchased from Reseller

Contact the reseller or distributor if

- You need ordering, service or any technical assistance.
- You received a damaged, incomplete or incorrect product.

Product Purchased Directly from Interphase Corporation

Contact Interphase Corporation directly for assistance with this, or any other Interphase Corporation product. Please have your purchase order and serial numbers ready.

Customer Support

United States: Telephone: (214) 654-5555
 Fax: (214) 654-5506
 E-Mail: intouch@iphase.com

Europe: Telephone: + 33 (0) 1 41 15 44 00
 Fax: + 33 (0) 1 41 15 12 13

World Wide Web

<http://www.interphase.com>

Anonymous FTP Server

<ftp.interphase.com>

END-USER LICENSE AGREEMENT FOR INTERPHASE CORPORATION SOFTWARE

IMPORTANT NOTICE TO USER—READ CAREFULLY

THIS END-USER LICENSE AGREEMENT FOR INTERPHASE CORPORATION SOFTWARE (“AGREEMENT”) IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR SINGLE ENTITY) AND INTERPHASE CORPORATION FOR THE SOFTWARE PRODUCTS ENCLOSED HEREIN WHICH INCLUDES COMPUTER SOFTWARE AND PRINTED MATERIALS (“SOFTWARE”). BY INSTALLING, COPYING, OR OTHERWISE USING THE ENCLOSED SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, PROMPTLY RETURN, WITHIN THIRTY DAYS, THE UNUSED SOFTWARE TO THE PLACE FROM WHICH YOU OBTAINED IT FOR A FULL REFUND.

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold.

Grant of License: You are granted a personal license to install and use the Software on a single computer solely for internal use and to make one copy of the Software in machine readable form solely for backup purposes.

Restrictions on Use: You may not reverse engineer, decompile, or disassemble the Software. You may not distribute copies of the Software to others or electronically transfer the Software from one computer to another over a network. You may not use the Software from multiple locations of a multi-user or networked system at any time. You may not use this software on any product for which it was not intended. You may not use this software on any non-Interphase product. LICENSEE MAY NOT RENT, LEASE, LOAN, OR RESELL THE SOFTWARE OR ANY PART THEREOF.

Ownership of Software: Interphase or its vendors retain all title to the Software, and all copies thereof, and no title to the Software, or any intellectual property in the Software, is being transferred.

Software Transfer: You may permanently transfer all of your rights under this Agreement, provided you retain no copies, you transfer all the Software, and the recipient agrees to the terms of this Agreement.

Limited Warranty: Interphase Corporation (“Seller”) warrants that (i) the hardware provided to Buyer (“Products”) shall, at the F.O.B. point, be free from defects in materials and workmanship for a period of one (1) year from the date of shipment to Buyer; (ii) the software and/or firmware associated with or embedded in the Products shall comply with the applicable specifications for a period of six (6) months from the date of shipment to Buyer; and (iii) its services will, when performed, be of good quality. Defective and nonconforming Products and software must be held for Seller’s inspection and returned at Seller’s request, freight prepaid, to the original F.O.B. point.

Upon Buyer’s submission of a claim in accordance with Seller’s Return and Repair Policy, Seller will, at its option either (i) repair or replace the nonconforming Product; (ii) correct or replace the software/firmware; (iii) rework the nonconforming services; or (iv) refund an equitable portion of the purchase price attributable to such nonconforming Products, software, or services. Seller shall not be liable for the cost of removal or installation of products or any unauthorized warranty work, nor shall Seller be responsible for any transportation costs, unless expressly authorized in writing by Seller. This warranty does not cover damage to the Product resulting from accident, disaster, misuse, negligence, improper maintenance, or modification or repair of the Product other than by Seller. Any Products or software replaced by Seller will become the property of Seller.

REMEDIES AND EXCLUSIONS. THE SOLE LIABILITY OF SELLER AND BUYER’S SOLE REMEDY FOR BREACH OF THESE WARRANTIES SHALL BE LIMITED TO REPAIR OR REPLACEMENT OF THE PRODUCTS OR CORRECTION OF THAT PART OF THE SOFTWARE, WHICH FAILS TO CONFORM TO THESE WARRANTIES. EXCEPT AS EXPRESSLY STATED HEREIN, AND EXCEPT AS TO TITLE, THERE ARE NO OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, IN CONNECTION WITH OR ARISING OUT OF ANY PRODUCT OR SOFTWARE PROVIDED TO BUYER.

IN NO EVENT SHALL SELLER HAVE ANY LIABILITY FOR INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, ARISING OUT OF THESE WARRANTIES, INCLUDING BUT NOT LIMITED TO LOSS OF ANTICIPATED PROFITS, LOSS OF DATA, USE OR GOODWILL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. (IC-199, 1/97)

Limitation of Liability: NEITHER INTERPHASE NOR ITS LICENSORS SHALL BE LIABLE FOR ANY GENERAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL, OR OTHER DAMAGES ARISING OUT OF THIS AGREEMENT EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Confidentiality: The Software is copyrighted and contains proprietary and confidential trade secret information of Interphase and its vendors. Licensee agrees to maintain the Software in confidence and not to disclose the Software to any third party without the express written consent of Interphase. Licensee further agrees to take all reasonable precautions to prevent access to the Software by unauthorized persons.

Termination: Without prejudice to any other rights, Interphase may terminate this Agreement if you fail to comply with any term or condition of the Agreement. In such event you must destroy the Software together with all copies, updates, or modifications thereof.

Export: You agree to comply with all export and re-export restrictions and regulations of the U.S. Department of Commerce or other applicable U.S. agency. You must not transfer the Software to a prohibited country or otherwise violate any such restrictions or regulations.

U.S. Government Restricted Rights: Use, duplication, or disclosure of the Software to or by the U.S. Government is subject to restrictions as set forth in the applicable U.S. federal procurement regulations covering commercial/restricted rights software. You are responsible for complying with the notice requirements contained in such regulations.

General: You acknowledge that you have read and understand this Agreement, and by installing and using the Software you agree to be bound by the terms and conditions herein. You further agree that this is the complete and exclusive Agreement between Interphase and yourself. No variation of the terms of this Agreement or any different terms will be enforceable against Interphase unless agreed to in writing by Interphase and yourself. The validity of this Agreement and the rights, obligations, and relations of the parties hereunder shall be determined under the substantive laws of the State of Texas. If any provision of this Agreement is held invalid, illegal, or unenforceable, the remaining provisions shall in no way be affected or impaired thereby. All rights in the Software not specifically granted in this Agreement are reserved by Interphase.

Contents

| | |
|--|-------|
| List of Figures | ix |
| List of Tables | xi |
| List of Examples | xiii |
| Using This Manual | xv |
| Purpose | xv |
| Audience | xv |
| Byte Ordering and Bit Coding Convention | xv |
| Type Definition | xv |
| Code Examples | xvi |
| Icon Conventions | xvi |
| Text Conventions | xvi |
| Checking and Downloading from the Interphase WWW/FTP Site | xvii |
| WWW Method | xvii |
| FTP Method | xviii |
| | |
| CHAPTER 1 Introduction | |
| Overview | 1 |
| Configuring TTY or PCI Console | 2 |
| Booting Process | 2 |
| Automatically Running a Flashed Operational Firmware | 4 |
| Running an Operational Firmware Downloaded Through the PCI Bus | 7 |
| Automatically Ethernet-Uploading and Running an Operational Firmware | 8 |
| MONITOR Configuration Parameters | 9 |
| MONITOR Command Categories | 10 |
| Information Category Commands | 10 |
| Configuration Category Commands | 11 |
| Test Category Commands | 12 |
| Utility Category Commands | 13 |
| On-line Help | 13 |
| Shortcuts | 13 |
| | |
| CHAPTER 2 Power-On Self Test (POST) | |
| General | 15 |
| Tests Run in POST | 15 |
| Reporting Tests on the Console | 18 |

| | |
|--|----|
| Reporting Tests with CPU LED | 19 |
| Reporting the Tests in a PCI Controller Register | 19 |
| Setting the POST Requirement Mode (POST Command) | 20 |

CHAPTER 3 Configuration Commands

| | |
|---|----|
| Monitor Configuration (SOFTWARE Command) | 23 |
| Description | 23 |
| SOFTWARE Command | 25 |
| TTY Parameters | 25 |
| Overview | 25 |
| Changing the Baud Rate (BAUDRATE Command) | 26 |
| Changing the Parity (PARITY Command) | 26 |
| Changing the Stop Bits (STOPBIT Command) | 26 |
| Changing the Echo Mode (ECHO Command) | 27 |
| IP/MAC Addresses Commands | 27 |
| Overview | 27 |
| Displaying the Board Ethernet MAC Address (MAC Command) | 28 |
| Changing the Board IP Local Address (LOCALIP Command) | 28 |
| Changing the Board IP Subnet Mask (MASKIP Command) | 28 |
| Changing the Server IP Address (SERVERIP Command) | 28 |
| Changing the Router IP Address (ROUTERIP Command) | 29 |
| Serial EEPROM Debugging Commands | 29 |
| Overview | 29 |
| Reading a Serial EEPROM Location (SREAD Command) | 29 |
| Writing a Serial EEPROM Location (SWRITE Command) | 30 |
| Dumping a Serial EEPROM Block (SDUMP Command) | 30 |
| POST Command | 30 |
| FWADDR Command | 31 |
| AUTORUN Command | 31 |
| FWCFGADDR Command | 32 |

CHAPTER 4 Information Commands

| | |
|-----------------------|----|
| INFO Command | 33 |
| INFOEQU Command | 35 |
| INFOPCI Command | 36 |
| CHIPREV Command | 37 |

CHAPTER 5 Built-In Self Tests (BIST)

| | |
|--|----|
| Introduction | 39 |
| Memories | 41 |
| 60x Bus Main Memory (M60X Command) | 41 |
| Test Description | 41 |
| M60X Command | 42 |
| Local Memory (MLOC Command) | 43 |
| Test Description | 43 |

| | |
|---|----|
| MLOC Command | 43 |
| CAM (MCAM Command) | 43 |
| Test Description | 43 |
| MCAM Command | 43 |
| Serial EEPROM (MSRPROM Command) | 44 |
| Test Description | 44 |
| MSRPROM Command | 45 |
| FLASH EEPROM (MFPROM Command) | 45 |
| Test Description | 45 |
| MFPROM Command | 45 |
| Chip Detection (CHIPREV Command) | 46 |
| Test Description | 46 |
| Processor Detection Description | 46 |
| PCI Chip Interface Detection Description | 46 |
| Framers Detection Description | 46 |
| Ethernet Interface Detection Description | 46 |
| IMA Chip Detection Description | 46 |
| CAM Detection Description | 47 |
| CHIPREV Command | 47 |
| Chip Reset | 47 |
| Overview | 47 |
| T1/E1/J1 Framers Reset (FRST Command) | 47 |
| Test Description | 47 |
| FRST Command | 48 |
| Ethernet Interface Reset (ERST Command) | 48 |
| Test Description | 48 |
| ERST Command | 48 |
| IMA Reset (IRST Command) | 48 |
| Test Description | 48 |
| IRST Command | 48 |
| CAM Reset (MCAMRST Command) | 49 |
| Test Description | 49 |
| MCAMRST Command | 49 |
| Chip Interrupt | 49 |
| Overview | 49 |
| T1/E1/J1 Framers Interrupt (FINT Command) | 49 |
| Test Description | 49 |
| FINT Command | 50 |
| Ethernet Interface Interrupt (EINT Command) | 50 |
| Test Description | 50 |
| EINT Command | 51 |
| IMA Interrupt (IINT Command) | 51 |
| Test Description | 51 |
| IINT Command | 52 |
| PCI Controller Interrupt (PINT Command) | 52 |
| Test Description | 52 |
| PINT Command | 53 |
| Loopback Tests Using The PowerQUICC II MCCs | 53 |

| | |
|--|----|
| Overview | 53 |
| Framers Operating Modes..... | 53 |
| HDLC Frames Format..... | 56 |
| Framers In Internal Loopback (FIL Command)..... | 57 |
| Test Description..... | 57 |
| FIL Command..... | 58 |
| Framers In External Loopback (FEL Command) | 59 |
| Test Description..... | 59 |
| FEL Command | 60 |
| Framers In Pass-Through Mode (FPT Command) | 62 |
| Test Description..... | 62 |
| FPT Command..... | 63 |
| Framers In Switched Mode (FSW Command)..... | 64 |
| Test Description..... | 64 |
| FSW Command | 64 |
| Loopback Tests For Ethernet..... | 65 |
| Overview | 65 |
| Ethernet In Internal Loopback (EIL Command)..... | 66 |
| Test Description..... | 66 |
| EIL Command | 66 |
| Ethernet In External Loopback (EEL Command)..... | 68 |
| Test Description..... | 68 |
| EEL Command..... | 68 |
| IMA Internal Loopback (IIL Command) | 68 |
| Test Description | 68 |
| IIL Command | 70 |
| IMA External Loopback (IEL Command)..... | 71 |
| Test Description | 71 |
| IEL Command | 72 |
| BERT (FB, FRL Commands)..... | 73 |
| Test Description | 73 |
| FB Command..... | 75 |
| FRL Command..... | 76 |
| Framer Line Status (FPHY Command)..... | 76 |
| Test Description | 76 |
| FPHY Command..... | 77 |
| IMA TDM Clocks (ICLK Command)..... | 78 |
| Test Description | 78 |
| ICLK Command..... | 78 |
| IMA Optional TDMs Clocks (ICLK8 Command) | 78 |
| Test Description | 78 |
| ICLK8 Command..... | 79 |
| IMA Optional TDMs External Loopback (IEL8 Command) | 80 |
| Test Description | 80 |
| IEL8 Command..... | 81 |
| IMA SRAM (ISRAM Command)..... | 81 |
| Test Description | 81 |

| | |
|---|----|
| ISRAM Command..... | 81 |
| LEDs (LED Command)..... | 82 |
| Test Description..... | 82 |
| LED Command..... | 83 |
| UTOPIA External Loopback (UEL Command)..... | 83 |
| Test Description..... | 83 |
| UEL Command..... | 83 |
| AGENCY (AGENCY Command)..... | 84 |
| Test Description..... | 84 |
| AGENCY Command..... | 84 |
| Testing the Product (PROD Command)..... | 85 |
| Test Description..... | 85 |
| PROD Command..... | 88 |
| Running a Command Several Times (LOOP Command)..... | 88 |
| Description..... | 88 |
| LOOP Command..... | 88 |
| Stopping a Command (S Command)..... | 88 |

CHAPTER 6 Utility Commands

| | |
|---|-----|
| Flashing Firmware Using the TTY (LOADFC Command)..... | 91 |
| Description..... | 91 |
| LOADFC – Example with Windows NT HyperTerminal Application..... | 91 |
| LOADFC Command..... | 94 |
| Ethernet Loading (LOADF/LOADM Commands and Auto-Boot)..... | 95 |
| Overview..... | 95 |
| Initial Configuration..... | 95 |
| Local Ethernet/MAC Address..... | 95 |
| Local IP Address and Local IP Mask..... | 96 |
| Server IP Address..... | 96 |
| Firmware Filename..... | 96 |
| Configuration Filename..... | 97 |
| Configuration Management..... | 97 |
| Auto-Boot Process Configuration Management..... | 98 |
| LOADF/LOADM Configuration Management..... | 98 |
| The Firmware Loading Process in Detail..... | 99 |
| RARP Module..... | 100 |
| BOOTP Module..... | 101 |
| TFTP Transfers..... | 101 |
| Starting Firmware Loading..... | 102 |
| Auto-Boot Process..... | 102 |
| LOADF Command..... | 102 |
| LOADM Command..... | 103 |
| Memory Debugging Commands..... | 104 |
| Overview..... | 104 |
| Reading a Memory Location (MREAD Command)..... | 104 |
| Writing a Memory Location (MWRITE Command)..... | 104 |
| Dumping a Memory Block (MDUMP Command)..... | 105 |

| | |
|---|-----|
| Filling a Memory Block (MFILL Command) | 105 |
| Moving a Memory Block (MMOVE Command) | 105 |
| Resetting the Board (RESET Command) | 106 |
| Description | 106 |
| RESET Command | 106 |
| Executing a Code From an Address (GO Command) | 106 |
| Description | 106 |
| GO Command | 106 |
| HELP Command | 107 |

APPENDIX A Mapping

| | |
|----------------------------------|-----|
| Serial EEPROM Mapping | 109 |
| 4539 Boot Firmware Mapping | 111 |
| SDRAM Mapping | 111 |
| FLASH EEPROM Mapping | 112 |

APPENDIX B Interfacing to the PCI Console

| | |
|--------------------|-----|
| Introduction | 115 |
|--------------------|-----|

APPENDIX C Monitor Commands

| | |
|--------------------|-----|
| Introduction | 117 |
|--------------------|-----|

APPENDIX D Boot Firmware

| | |
|--|-----|
| Introduction | 121 |
| Files Description | 121 |
| Recompiling the 4539 Boot Firmware | 126 |

APPENDIX E Re-Flashing Code into the 4539

| | |
|--|-----|
| Flash EEPROM Memory Mapping | 127 |
| Methods for Flashing Firmware into the Board | 128 |
| Flashing the Boot Firmware | 128 |
| Flashing Through the TTY Port | 128 |
| Flashing Through the PCI bus | 131 |
| Under VxWorks | 131 |
| Under Solaris | 133 |
| Flashing the Operational Firmware | 135 |
| Flashing Through the TTY Port | 135 |
| Flashing Through the PCI bus | 136 |
| Under VxWorks | 137 |
| Under Solaris | 139 |

APPENDIX F Server Consideration

Overview 141

Solaris Operating System 141

 RARP 141

 Configuration 141

 Execution 142

 BOOTP 142

 TFTP 143

 Configuration 144

 Execution 144

Red Hat Linux Release 6.x 144

 RARP 144

 BOOTP 144

 Configuration 144

 Execution 146

 TFTP 146

 Configuration 146

 Execution 147

Windows NT/Win32 147

 RARP 147

 BOOTP 147

 TFTP 147

APPENDIX G External Loopback Connectors

T1/E1/J1 Loopback Connector 149

Ethernet Loopback Connector 149

P3LBTstConn and P4LBTstConn Loopback Connectors 150

List of Figures

| | | |
|--------------|--|-----|
| Figure 1-1. | Boot Process..... | 2 |
| Figure 1-2. | POST Result and General Information Displayed by the MONITOR | 3 |
| Figure 1-3. | MONITOR Menu..... | 4 |
| Figure 1-4. | 4539 Boot Firmware Size Displayed by the INFO Command | 5 |
| Figure 1-5. | Information Category Commands..... | 10 |
| Figure 1-6. | Configuration Category Commands | 11 |
| Figure 1-7. | Test Category Commands..... | 12 |
| Figure 1-8. | Utility Category Commands | 13 |
| Figure 5-1. | Multiplexed Direct Mode Configuration | 53 |
| Figure 5-2. | Independent Direct Mode Configuration | 54 |
| Figure 5-3. | Switched Mode Configuration | 54 |
| Figure 5-4. | Pass-Through Mode configuration (0->1 & 2->3)..... | 55 |
| Figure 5-5. | Pass-Through Mode Configuration (1->0 & 3->2)..... | 55 |
| Figure 5-6. | Framers Internal Loopback Test Configuration..... | 58 |
| Figure 5-7. | Framers External Loopback Test Configuration..... | 60 |
| Figure 5-8. | Configuration for Pass-Through Port 0 to Port 1 and Port 2 to Port 3 Test..... | 62 |
| Figure 5-9. | Configuration for Pass-Through Port 1 to Port 0 and Port 3 to Port 2 Test..... | 63 |
| Figure 5-10. | Framers in Switched Mode Test Configuration..... | 64 |
| Figure 5-11. | Ethernet Frame Structure | 65 |
| Figure 5-12. | Ethernet Internal Loopback Between MII and Digital Block..... | 66 |
| Figure 5-13. | IMA internal Loopback Test Configuration | 70 |
| Figure 5-14. | IMA External Loopback Test Configuration | 72 |
| Figure 5-15. | BERT Scenario | 74 |
| Figure 5-16. | ICLK8 Test – Clock Configuration with Loopback Connector on P4 | 79 |
| Figure 5-17. | IEL8 Test – IMA TDMs 4 to 7 Configuration with Loopback Connectors on P4 | 80 |
| Figure 5-18. | LED Configuration | 82 |
| Figure 6-1. | Auto-Boot Configuration Management | 98 |
| Figure 6-2. | LOADF/LOADM Configuration Management | 99 |
| Figure 6-3. | Firmware Loading Process..... | 100 |
| Figure 6-4. | Example of Building the Name of the Configuration File..... | 101 |
| Figure E-1. | Flash EEPROM Mapping | 128 |
| Figure G-1. | P3LBTstConn - Example | 150 |
| Figure G-2. | P4LBTstConn - Example | 151 |
| Figure G-3. | P3LBTstConn and P4LBTstConn Mounted on a 4539 | 152 |
| Figure G-4. | A 4539 Equipped with P3/4LBTstConn on a PCI Carrier..... | 153 |

List of Tables

| | | |
|------------|---|-----|
| Table 2-1. | POST Tests Run for Each 4539 Version | 15 |
| Table 2-2. | BIST Equivalent Tests For POST Tests | 16 |
| Table 2-3. | POST Tests Map in PowerSpan MBR2 Register | 19 |
| Table 3-1. | Commands to Change Software Configuration | 23 |
| Table 4-1. | INFO Command Field Descriptions | 33 |
| Table 4-2. | INFOPCI Command Field Descriptions | 36 |
| Table 5-1. | BIST Tests | 39 |
| Table 5-2. | Memory Pseudo-Random Pattern 1 | 41 |
| Table 5-3. | Memory Pseudo-Random Pattern 2 | 41 |
| Table 5-4. | CAM Entries for MCAM Test | 43 |
| Table 5-5. | Frame Content | 56 |
| Table 5-6. | Framer Alarms | 76 |
| Table 5-7. | LED Definitions | 82 |
| Table 5-8. | PROD Command Test List | 85 |
| Table 6-1. | Configuration Parameters | 97 |
| Table A-1. | 4539 Serial EEPROM General Mapping | 109 |
| Table A-2. | Serial EPROM Byte 0xC2 Mapping | 110 |
| Table A-3. | Serial EPROM Byte 0xC2 Details | 110 |
| Table A-4. | Serial EPROM Byte 0xC3 Mapping | 111 |
| Table A-5. | Serial EPROM Byte 0xC3 Details | 111 |
| Table A-6. | SDRAM Mapping | 112 |
| Table A-7. | 4 MB FLASH EEPROM Mapping | 112 |
| Table C-1. | Monitor Commands | 117 |
| Table D-1. | 4539 Boot Firmware Source File Descriptions | 121 |
| Table G-1. | T1/E1/J1 RJ48C Connector | 149 |
| Table G-2. | Ethernet 10/100 RJ45 Connector | 149 |
| Table G-3. | P3LBTstConn - Connections | 150 |
| Table G-4. | P4LBTstConn - Connections | 151 |

List of Examples

- Example 1-1. Flashed Operational Firmware Mapping 6
- Example 1-2. Commands to Automatically Run an Operational Flashed Firmware 7
- Example 2-1. Normal POST Result Display 18
- Example 2-2. Test 24 Failure 19
- Example 2-3. Quick POST Result Display 21
- Example 2-4. Disable POST Result Display 21
- Example 4-1. General Information Display 33
- Example 4-2. Equipment Information Display on a 4539-000 or 4549-001 35
- Example 4-3. Equipment Information Display on a 4539-002 or 4539-003 35
- Example 4-4. Equipment Information Display on a 4539-004 or 4539-005 35
- Example 4-5. PCI General Information Display 36
- Example 4-6. Chip Revision Display 37
- Example 5-1. Stopping a Command 89
- Example 6-1. LOADFC - Step 1 92
- Example 6-2. LOADFC - Step 2 92
- Example 6-3. LOADFC - Step 3 92
- Example 6-4. LOADFC - Step 4 93
- Example 6-5. LOADFC - Step 5 94

Using This Manual

Purpose

This *4539 Built-In Self Tests and Monitor Manual* is one of the documents provided with the 4539 Board Development Kit (BDK). This manual provides high-level information on how to use the 4539 Boot Firmware. Other documents in the BDK are the *4539 Hardware Reference Manual* that provides all the necessary information to develop an on-board software from scratch (also referred to as operational firmware), and the *4539 Board Installation and Maintenance Manual* that provides instruction for installing and maintaining the board.

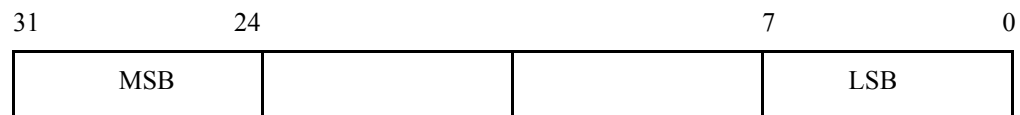
Audience

This guide was written assuming that readers have extensive knowledge of the C programming language and of methods for developing and installing software drivers.

Byte Ordering and Bit Coding Convention

The PCI bus uses the Little Endian Byte ordering: byte 0 in a 32-bit word is the Least Significant Byte (LSB) from an arithmetic point of view and is noted D(7:0). The PowerPC architecture uses the Big Endian Byte ordering: byte 0 in a 32-bit word is the Most Significant Byte (MSB) from an arithmetic point of view and is noted D(31:24).

The PowerPC architecture uses the very unusual Little Endian Bit convention, where bit 0 is on the left and is the most significant bit. Unless otherwise noted, this document does not use this convention. Instead, it uses the classical bit coding convention, where bit 0 (on the right) is the least significant bit and bit i is the 2^i weight bit. This is the Big Endian Bit convention. This coding convention applies to data, addresses, and bit fields. In the following figure, MSB means Most Significant Byte and LSB Least Significant Byte:



The standard C convention is used to identify the numeric format of arithmetical values:

- No prefix for decimal values
- 0x prefix for a hexadecimal value

For example $0x12 = 18$.

Type Definition

Only a few basic types are used:

- byte: unsigned, coded as 8 bits
- word: unsigned, coded as two contiguous bytes, most significant first
- dword: unsigned, coded as two contiguous words, most significant first

Code Examples

This document provides several algorithm descriptions presented in PowerPC assembly language and in C language.

Icon Conventions

Icons draw your attention to especially important information:



NOTE

The Note icon indicates important points of interest related to the current subject.



CAUTION

The Caution icon brings to your attention those items or steps that, if not properly followed, could cause problems in your machine's configuration or operating system.



WARNING

The Warning icon alerts you to steps or procedures that could be hazardous to your health, cause permanent damage to the equipment, or impose unpredictable results on the surrounding environment.

Text Conventions

The following conventions are used in this manual. Computer-generated text is shown in typewriter font. Examples of computer-generated text are: program output (such as the screen display during the software installation procedure), commands, directory names, file names, variables, prompts, and sections of program code.

Computer-generated text example

Commands to be entered by the user are printed in **bold Courier** type. For example:

```
cd /usr/tmp
```

Pressing the return key (↵ **Return**) at the end of the command line entry is assumed, when not explicitly shown. For example:

```
/bin/su
```

is the same as:

```
/bin/su ↵ Return
```

Required user input, when mixed with program output, is printed in **bold Courier** type. References to UNIX programs and manual page entries follow the standard UNIX conventions.

When a user command, system prompt, or system response is too long to fit on a single line, it will be shown as

```
Do you want the new kernel moved into  
\vmunix? [y]
```

with a backslash at either the beginning of the continued line or at the end of the previous line.

Checking and Downloading from the Interphase WWW/FTP Site

The latest production software drivers, firmware, and documentation (in Adobe Acrobat PDF or text format) for our current products are available on our WWW / FTP site. Interphase recommends our customers visit the web site often to verify that they have the latest version of driver, firmware, or documentation.

WWW Method

1. Access the following web page:
<http://www.ipphase.com>
2. Move the mouse (or other pointer) and click on the `Products` option. A menu will appear on the left side with `Telecom` and `Server I/O` options. Choose the appropriate menu item.
3. A new web page with a list of the currently offered products will appear. Choose the correct communications controller for your system bus and configuration needs by clicking on the product number (i.e. 4538, 6535, 4575, etc.).
4. The Product Description page appears for that product number. At the left side of the page is a list showing further information web pages for that product. Choose the `SW Downloads` item.
5. A new web page appears with a list of the latest released drivers available for that adapter based on the operating system. Click on the line that describes your driver requirement. Depending on your browser configuration, the driver will now

download to your system. If this doesn't work correctly, try to 'right-click' on the proper driver and choose an option that will save the file to your local file system.

FTP Method

1. From your command line, enter the following:
`ftp://ftp.ipphase.com`
2. At the Login: prompt, enter **anonymous**
3. At the Password: prompt, enter your E-mail address.
4. At the ftp prompt, enter **binary**
5. Enter **cd pub**
6. To list all the available product technology directories, enter **dir**. The `00README.txt` file in each directory gives a description of the files and subdirectories in that directory.
7. To access the directory for the technology that you require, enter **cd** followed by the appropriate directory name. Most technology directories also have bus and operating system subdirectories. In these cases, you must choose the proper bus and operating system by typing **cd <directory>** for the appropriate subdirectories.
8. To download one or more files to your local directory, enter **get <filename>**
9. To exit the FTP site, enter **quit** or **bye**

Overview

The 4539 PMC[®] Quad T1/E1/J1 Communication Controller is delivered with Boot Firmware stored in the FLASH EEPROM of the controller, which includes the following software elements:

- **PowerQUICC II[™] and 4539 minimal initialization code (STARTUP)**

STARTUP is the entry point after a power-on or a reset exception. STARTUP configures the PowerQUICC II processor and several other critical elements such as SDRAM memories. For more information, see the *4539 Hardware Reference Manual*.

- **Power-On Self Tests (POST)**

POST is a set of tests that are executed after each power-on of the communication controller. For more information, see [Chapter 2 Power-On Self Test \(POST\) on page 15](#).

- **Interactive Monitor (MONITOR)**

MONITOR allows configuring the Boot Firmware, executing Built-In Self Tests (BIST), displaying communication controller information, dumping memories, and downloading and running firmware.

MONITOR access is done either through the PCI Interface or a TTY console. The interactive monitor is described in this chapter.

- **A Set of Built-In Self Tests (BIST)**

BIST is intended to qualify the communication controller at production time or in the field. For more information, see [Chapter 5 Built-In Self Tests \(BIST\) on page 39](#).



NOTES

The Software Developer can decide to use the Interphase Boot Firmware or to develop its own Boot Firmware. The first solution is recommended for the following reasons:

- PowerQUICC II and 4539 initial hardware configuration code is already developed.
- Interphase Boot Firmware provides several ways to download and execute developer's Operational Firmware.
- Interphase Boot Firmware can be used during the life of the product for Operational Firmware updates and field unit tests.

For the second solution, refer to the *4539 Hardware Reference Manual* which describes 4539 hardware and Interphase Boot Firmware in detail.

Configuring TTY or PCI Console

To access MONITOR through a TTY console, do the following:

- Connect a TTY cable between a TTY console and 4539 TTY port (see the *4539 Hardware Reference Manual* for TTY connector pin-out).
- Configure the TTY console with the following parameters: 9600 Baud, 8 Data Bits, 1 Stop Bit, No parity, No flow control.

To access MONITOR through the PCI host, see the VxWorks and Solaris Maintenance Tools Appendices in the *4539 Board Installation and Maintenance Manual*.

Booting Process

The Boot Firmware boot process depends on the MONITOR Configuration that is stored in the 4539 Serial EEPROM (for more information on Configuration Parameters, see [Chapter 3 Configuration Commands on page 23](#)). [Figure 1-1](#) describes the boot process.

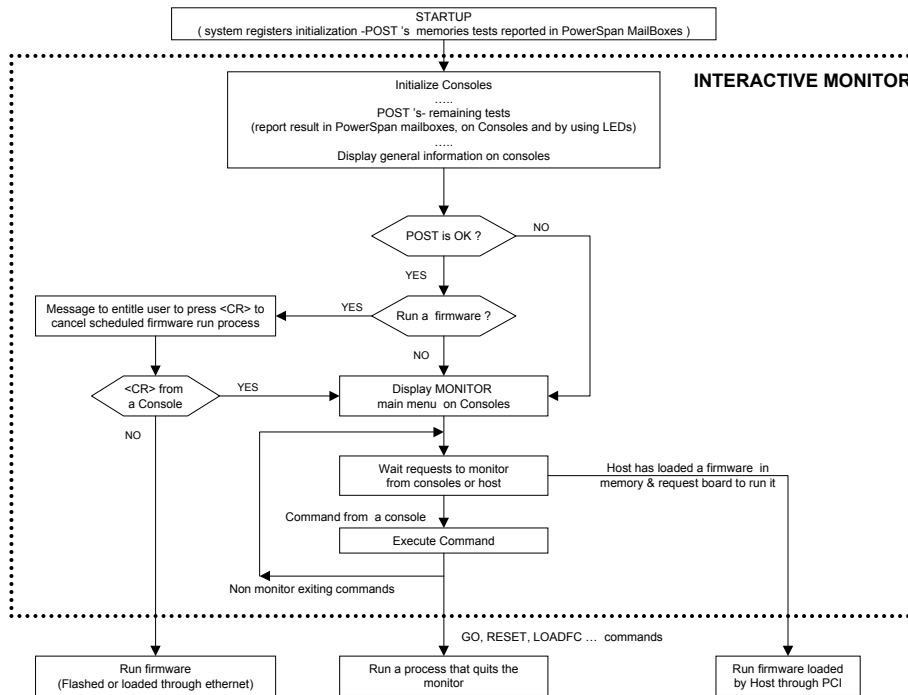


Figure 1-1. Boot Process

STARTUP is executed after each power-on of the communication controller (and after each communication controller reset) and initializes the processor and memories. STARTUP also executes memories' Power-On Self Test (SDRAM, CAM, etc.). STARTUP is written in assembly language. Once STARTUP has been executed, code written in a high-level language such as “C” can be executed (for more information on Boot Firmware source code, see the *4539 Hardware Reference Manual*)

After the STARTUP phase, MONITOR initializes PCI and TTY consoles, executes remaining POST, and displays communication controller information on each console as shown on [Figure 1-2](#), this information can also be retrieved later using the INFO command. (For more information, see [Chapter 4 Information Commands on page 33](#).)

```

***** POST *****
1 - Test Boot Firmware compatibility..... OK
2 - Test Main SDRAM ..... OK
3 - Test Local SDRAM ..... OK
4 - Test CAM memory ..... OK
5 - Test Serial EEPROM ..... OK
6 - Test CAM chip detection ..... OK
7 - Test Framers chip detection ..... OK
8 - Test Ethernet chip detection ..... OK
9 - Test IMA chip detection ..... OK
10 - Test Framers chip reset signal ..... OK
11 - Test Ethernet chip reset signal ..... OK
12 - Test IMA chip reset signal ..... OK
13 - Test CAM chip reset signal ..... OK
14 - Test PCI chip local interrupt signal ..... OK
15 - Test Framers chip interrupt signal ..... OK
16 - Test Ethernet chip interrupt signal ..... OK
17 - Test IMA chip interrupt signal ..... OK
18 - Test Framers internal loop-back - independent mode .. OK
19 - Test Framers internal loop-back - multiplexed mode .. OK
20 - Test Framers pass-through mode (0->1,2->3) ..... OK
21 - Test Framers pass-through mode (1->0,3->2) ..... OK
22 - Test Ethernet internal loop-back ..... OK
23 - Test IMA ports clocks ..... OK
24 - Test IMA port 0 internal loop-back ..... OK
25 - Test IMA port 1 internal loop-back ..... OK
26 - Test IMA port 2 internal loop-back ..... OK
27 - Test IMA port 3 internal loop-back ..... OK
28 - Test IMA SRAM ..... OK
** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK **

General Information:

Identification      : 4539
Hardware revision   : 0
Serial Number       : 1428130
Manufacturing date  : 10/13/2000
EPLD                : SX00598-A00
Boot firmware       : SX00595-A00 V1.00
  - Start Address   : 0xFFF00100
  - Stop Address    : 0xFFF359E0
  - Size (in bytes) : 0x000358E0
Ports               : Ethernet 10/100
                    : 4 T1/E1/J1 on P4 (Rear Access)
                    : ATM UTOPIA on P3 - PTENB# asserted (=0)
                    : 4 IMA/UNI TDMs on P4
MAC address         : 00.00.77.95.CA.A2
POST result         : OK (0x00000000)

Press <CR> to get menu ...

Jump to firmware at address FF810100

```

Figure 1-2. POST Result and General Information Displayed by the MONITOR

If POST fails, or if POST succeeds and MONITOR is not configured to run an Operational Firmware (a flashed or Ethernet-uploaded firmware), the MONITOR Menu is displayed showing available command categories, as shown in [Figure 1-3](#).

Commands' categories:

```
I : Information
C : Configuration
T : Tests
U : Utilities
```

To show all commands' on a category, type 'help <category>'

>

Figure 1-3. MONITOR Menu

When MONITOR has been configured to run an Operational Firmware, the user still has a way to stop the boot process (for example, to update a new flashed Operational Firmware, or to execute BIST). To stop the boot process and access MONITOR, press <CR> (Carriage Return) on the TTY console during POST (<CR> is usable until one second after POST ends).

The MONITOR ends when one of the following items occurs:

- POST succeeds and MONITOR is configured to run an Operational Firmware (assuming that no <CR> has been pressed during POST) that is either flashed in FLASH EEPROM (see [Automatically Running a Flashed Operational Firmware](#)) or uploaded through ethernet (see [Automatically Ethernet-Uploading and Running an Operational Firmware on page 8](#)).
- MONITOR commands that exit from the MONITOR are executed (GO, RESET, etc.).
- Host has loaded an Operational Firmware in memory and requested it to run. For more information, see [Running an Operational Firmware Downloaded Through the PCI Bus on page 7](#).

Automatically Running a Flashed Operational Firmware

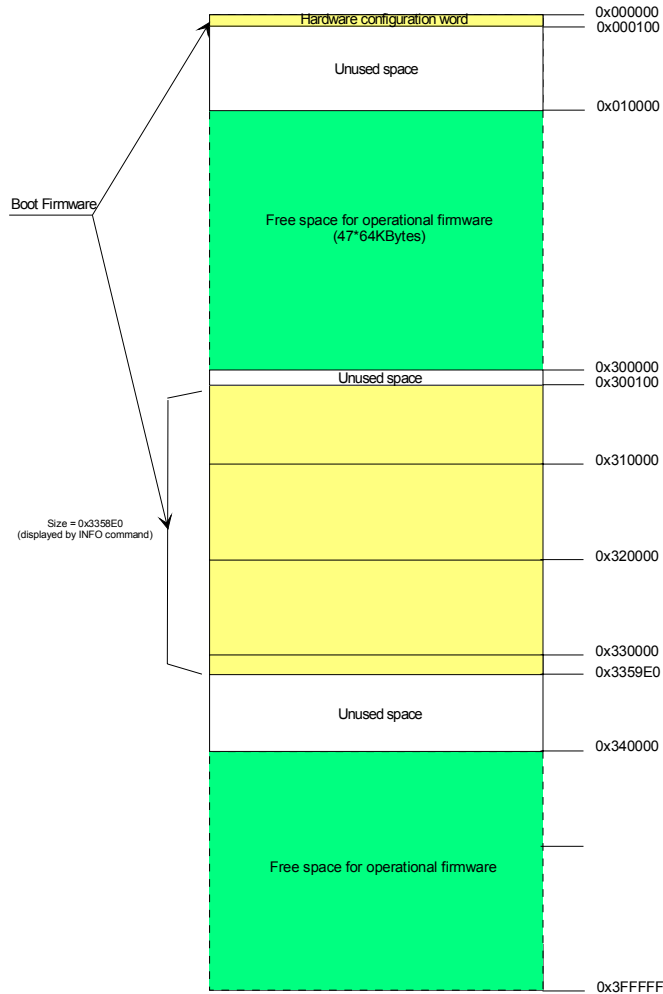
The MONITOR offers the ability to automatically run a flashed Operational Firmware after the POST. If the POST is not successful, this operation cannot be done automatically and the user has to use the GO MONITOR command (see [Executing a Code From an Address \(GO Command\) on page 106](#)) to force the jump to the flashed Operational Firmware.

The 4539 Boot Firmware is located at the beginning of the last Megabyte of the FLASH EEPROM. The user must use the INFO MONITOR command to retrieve its size. [Figure 1-4](#) shows this information in bold: the size displayed (it can change from one Boot Firmware version to another) is 0x000358E0 meaning that the code uses between 3 and 4

64 kbytes sectors allowing Operational Firmware to locate from the 5th sector of the last Megabyte, but Operational Firmware can also be programmed in a free space of 47 64-KByte sectors between the Hardware Configuration word and the 4539 Boot Firmware.

```
>info
General Information:
  Identification      : 4539
  Hardware revision   : 0
  Serial Number       : 1428130
  Manufacturing date  : 13/10/2000
  EPLD                : SX00598-A00
  Boot firmware       : SX00595-A00 V1.00
    - Start Address   : 0xFFF00100
    - Stop Address    : 0xFFF359E0
    - Size (in bytes) : 0x000358E0
  Ports               : Ethernet 10/100
                       4 T1/E1/J1 on P4 (Rear Access)
                       ATM UTOPIA on P3 - PTENB# asserted (=0)
                       4 IMA/UNI TDMs on P4
  MAC address         : 00.00.77.95.CA.A2
  POST result         : OK (0x00000000)
>
```

Figure 1-4. 4539 Boot Firmware Size Displayed by the INFO Command



Example 1-1. Flashed Operational Firmware Mapping

Storage of Operational Firmware into 4539 FLASH EEPROM can be done using three methods:

- Through the PCI bus using the `iphsetup` utility from a PCI master system board (this is the card that is plugged in the system slot of the CompactPCI chassis). For more information, see the VxWorks and Solaris Maintenance Tools Appendices in the *4539 Board Installation And Maintenance Manual*.
- Through the TTY using the `LOADFC` command of MONITOR. For more information, see [Flashing Firmware Using the TTY \(LOADFC Command\)](#) on page 91.
- Through the Ethernet using the `LOADF` command of MONITOR. For more information, see [LOADF Command](#) on page 103.

Once the Operational Firmware is programmed in FLASH EEPROM and if, for example, it starts at address 0x40000 (4th sector of the FLASH), considering the fact that the FLASH EEPROM is mapped at address 0xFF800000 (see [4539 Boot Firmware Mapping on page 111](#)), the user must program 0xFF840000 in Serial EEPROM.

To do this address programming, use the `FWADDR MONITOR` command (see [Example 1-2](#)).

To instruct the MONITOR to automatically jump to this address at the next restart of the 4539 board, use the `AUTORUN` command with `f` option (see [AUTORUN Command on page 31](#)) and reset the board by using the `RESET` command (see [RESET Command on page 106](#)).

Example 1-2. Commands to Automatically Run an Operational Flashed Firmware

```
>faddr FF840000
>autorun f
>reset
```

Running an Operational Firmware Downloaded Through the PCI Bus

The 4539 Boot Firmware is able to run an Operational Firmware downloaded by the host. First, the user must verify that the automatic flashed firmware run configuration is disabled: Command: `autorun d` (for more information, see [AUTORUN Command on page 31](#)).

The entire sequence to download and run an Operational Firmware through the PCI bus is as follows:

1. The host resets the 4539 board processor by enabling the `-SRESET` input signal of the 4539 PowerQUICC II processor: this is done by setting the `DB5_EN` bit of PowerSpan register `IER0`.
2. The host enables the `-ATN (-IRQ1)` signal to the PowerQUICC II by setting the `DB2_EN` bit in PowerSpan register `IER0`.
3. The host runs the 4539 board processor by disabling the `-SRESET` input signal of the 4539 PowerQUICC II processor: this is done by resetting the `DB5_EN` bit of PowerSpan register `IER0`.
4. The PowerQUICC II boots and completes the POST. Once finished, it places the error code in PowerSpan mailbox register `MBOX2` and then it resets the `DB2_EN` bit in PowerSpan register `IER0` (which disables the `-ATN` signal).
5. The host polls PowerSpan register `ISR0` every millisecond until it sees the `DB2` bit reset.
6. The host reads the POST result in PowerSpan mailbox register `MBOX2` and if no error was reported, it continues. If POST reports errors, it is up to the host to decide whether it can continue or not.

7. The host downloads the operational code directly into the 4539 SDRAM memory, through the PowerSpan “PCI to local window 0”. It also places, at local address 0, the PowerQUICC II op-code of a jump to the start of this operational code. Note: Take care not to overlap with the 4539 Boot Firmware that is run in SDRAM. See [4539 Boot Firmware Mapping on page 111](#).
8. During this time, the PowerQUICC II blinks its CPU LED and waits until the ATN signal is set again.
9. The host sets the –ATN signal to indicate to the PowerQUICC II that the operational code has been loaded and that it must now jump to local address 0, where a jump op-code will place it at the beginning of the operational code.
10. The PowerQUICC II sees that the –ATN signal is set, so it jumps to local address 0.
11. Somewhere in the operational code, the –ATN signal is reset to indicate that the operational code initialization is finished.

During development phase, developers could use this sequence to download an Operational Firmware in SDRAM. Once the Operational Firmware is stabilized, developers can download the code permanently in the 4539 FLASH EEPROM as described in the previous section.

Automatically Ethernet-Uploading and Running an Operational Firmware

After the POST, the MONITOR offers the ability to automatically upload an Operational Firmware through the Ethernet (this process is also called Auto-Boot) and run it. If the POST is not successful, this operation cannot be done automatically but can be manually forced by the user with the `LOADM` command – this section provides the main steps of the Auto-Boot process; additional details can be found in [Ethernet Loading \(LOADF/LOADM Commands and Auto-Boot\) on page 95](#).

The steps for the Ethernet-upload and run of an Operational Firmware are as follows:

1. The Operational Firmware Ethernet-uploading process checks if a local IP address is set in the Serial EEPROM (see [Changing the Board IP Local Address \(LOCALIP Command\) on page 28](#) and [Changing the Board IP Subnet Mask \(MASKIP Command\) on page 28](#)). If this IP address is FF.FF.FF.FF, it sends up to three RARP requests on the Ethernet device to get it. If no reply is received, it then tries BOOTP protocol by sending up three BOOTP requests. If no BOOTP reply is received, load process fails. RARP and BOOTP requests are always broadcast on the local network.
2. As the local IP address is known, the process requests a TFTP host whose IP address is set in the Serial EEPROM (see [Changing the Server IP Address \(SERVERIP Command\) on page 28](#)) for a configuration file. The IP address of that host can be FF.FF.FF.FF (broadcast value) – in that case, if several TFTP servers are listening on the local network, the process handles the first positive reply of any

of them and discards all other replies (errors or other positive replies)

The name of that configuration file is:

- Built from the local IP address by concatenating the four IP digits expressed in hexadecimal and converted to a string (for example, the hexadecimal form of 157.175.33.170 is 9D.AF.21.AA, so the name of the requested configuration file will be “9DAF21AA”).
- Read from the BOOTP reply if BOOTP server has been configured to return an explicit filename.

3. Once the configuration file has been loaded, it is parsed to find the following fields:

- FIRMWARE =... Specifies the name of the file firmware to download.
- TFTPSEVER =... Specifies the IP address of the host running a TFTP server which is able to send the firmware file.

Note that no space nor blank characters are allowed in these fields. All characters found behind the equal ‘=’ sign are parsed as field value until a non-printable character is found. The FIRMWARE= field is mandatory in the configuration file while the TFTPSEVER= is optional: if not present, the address of the host that has sent this configuration file is used to download the firmware file.

4. Once the firmware file has been downloaded, it is parsed as an ELF format firmware that is intended to be loaded in RAM. ELF section addresses are checked to be in that range. If that load succeeds, a jump is done to the ELF entry point address.

To instruct the MONITOR to automatically upload an Operational Firmware at the next restart of the 4539 board, the user must use the `AUTORUN` command with `e` option (see [AUTORUN Command on page 31](#)) and reset the board by using the `RESET` command (see [Resetting the Board \(RESET Command\) on page 106](#)).

```
>autorun e
>reset
```

MONITOR Configuration Parameters

The MONITOR manages the following configuration parameters:

- MAC address of the 4539 communication controller (read-only parameter)
- TTY Parameters
- Flag for an automatic jump to a Flashed Operational Firmware after POST
- Flag for an automatic Ethernet-upload and run of an Operational Firmware after POST
- POST Mode (Normal, Quick, or Disabled)
- Operational Firmware Address in FLASH
- Specific Configuration Address in FLASH

For more information, see [Configuration Commands on page 23](#).

MONITOR Command Categories

As shown on [Figure 1-3 on page 4](#), MONITOR commands are grouped into four categories:

- Information: Provides a set of commands to display 4539 general communication controller information.
- Configuration: Provides a set of commands to display (or change) MONITOR configuration.
- Tests: Provides a set of commands to execute Built-In Self Tests.
- Utilities: Provides a set of commands for general purposes.

The main MONITOR menu can be displayed at any time by pressing <CR>.

The list of available commands for each category can be displayed by entering the first letter of the category followed by a <CR> (Carriage Return).

Information Category Commands

```
>i  
  
I category commands:  
  
INFO      : General information  
INFOPCI   : PCI information  
INFOEQU   : Equipment information  
CHIPREV   : Chips' revisions  
  
For help, type 'help <command>'  
  
>
```

Figure 1-5. Information Category Commands

For more information, see [Information Commands on page 33](#).

Configuration Category Commands

```
>C
C category commands:

CONFIGURATION   BOARD       CONSOLE     S-EEPROM    OTHERS
-----
SOFTWARE        MAC        PARITY      SREAD       POST
                LOCALIP    STOPBIT     SWRITE      FWADDR
                MASKIP    BAUDRATE    SDUMP       FWCFGADDR
                SERVERIP  ECHO       AUTORUN
                ROUTERIP

For help, type 'help <command>'

>
```

Figure 1-6. Configuration Category Commands

For more information, see [Configuration Commands](#) on page 23.

Test Category Commands

=

category commands:

| MEMORIES | FRAMERS | ETHERNET | IMA | OTHERS |
|----------|---------|----------|-----------|----------|
| ----- | ----- | ----- | --- | ----- |
| M60X | FB | EIL | ICLK | LED |
| MLOC | FEL | EEL | IIL | PROD (*) |
| MCAM | FIL | EINT | IINT | PINT |
| MSRPROM | FINT | ERST | IRST | AGENCY |
| MFPROM | FPHY | | ISRAM | UEL (*) |
| MCAMRST | FRL | | IEL | LOOP |
| | FRST | | ICLK8 (*) | S |
| | FSW (*) | | IEL8 (*) | |
| | FPT | | | |

(*) : Caution - to be successful, tests with (*) have to be run with board inserted in a PC on a PCI carrier board with special loop-back test connectors on P3/P4 (Refer to BIST & Monitor Manual).

For help, type 'help <command>'

Figure 1-7. Test Category Commands

For more information, see [Built-In Self Tests \(BIST\)](#) on page 39.

Utility Category Commands

```

U category commands:

LOAD      MEMORY      OTHERS
-----
LOADF     MREAD        RESET
LOADFC    MWRITE      GO
LOADM     MDUMP        HELP
           MFILL
           MMOVE

For help, type 'help <command>'

```

Figure 1-8. Utility Category Commands

For more information, see [Utility Commands on page 91](#).

On-line Help

All MONITOR commands have an on-line help available by entering `HELP <Command>` (for example: `HELP RESET` will display on-line help for the `RESET` command).

Shortcuts

Some MONITOR commands have a shortcut command (for example: `BR` is the shortcut for the `BAUDRATE` command). For a complete list of command shortcuts, see [Monitor Commands on page 117](#).

General

The Power-On Self Test is run at the start of the board power-up sequence.

Tests Run in POST

POST is a set of elementary tests. Each test has a unique identifier that does not change among 4539 boards that are differently equipped. [Table 2-1](#) provides a list of the tests run in POST; For each test and 4539-XXX version (4539-001, 4539-002, etc.) an X indicates that the test is run for that version.

Table 2-1. POST Tests Run for Each 4539 Version

| No. | Title | 4539 Version | | |
|-----|---|--------------|-------------|-------------|
| | | 000/ 001 | 002/ 003 | 004/ 005 |
| 1 | Boot Firmware compatibility | X | X | X |
| 2 | Main SDRAM | X | X | X |
| 3 | Local SDRAM | X | | |
| 4 | CAM memory | X | | |
| 5 | Serial EEPROM | X | X | X |
| 6 | CAM chip detection | X | | |
| 7 | Framer chip detection | X | X | X |
| 8 | Ethernet chip detection | X | X | X |
| 9 | IMA chip detection | X | | |
| 10 | Framer chip reset signal | X | X | X |
| 11 | Ethernet chip reset signal | X | X | X |
| 12 | IMA chip reset signal | X | | |
| 13 | CAM chip reset signal | X | | |
| 14 | PCI chip local interrupt signal | X | X | X |
| 15 | Framers chip interrupt signal | X | X | X |
| 16 | Ethernet chip interrupt signal | X | X | X |
| 17 | IMA chip interrupt signal | X | | |
| 18 | Framer internal loopback – independent mode | X | X | X |
| 19 | Framer internal loopback – multiplexed mode | X | X | X |
| 20 | Framer pass-through mode (0->1,2->3) | X | X | X |

Table 2-1. POST Tests Run for Each 4539 Version (cont)

| No. | Title | 4539 Version | | |
|-----|--------------------------------------|--------------|-------------|-------------|
| | | 000/ 001 | 002/ 003 | 004/ 005 |
| 21 | Framer pass-through mode (1->0,3->2) | X | X | X |
| 22 | Ethernet internal loopback | X | X | X |
| 23 | IMA port clocks | X | | |
| 24 | IMA port 0 internal loopback | X | | |
| 25 | IMA port 1 internal loopback | X | | |
| 26 | IMA port 2 internal loopback | X | | |
| 27 | IMA port 3 internal loopback | X | | |
| 28 | IMA SRAM | X | | |

Table 2-2 provides the BIST equivalent test for each POST test.

Table 2-2. BIST Equivalent Tests For POST Tests

| No. | POST Test Title | BIST Equivalent Test | References/Comments |
|-----|-----------------------------|--------------------------|---|
| 1 | Boot Firmware compatibility | No equivalent BIST test. | This test checks that the Boot Firmware is compatible with the board's PCI device and revision identifiers. |
| 2 | Main SDRAM | M60X | See <i>60x Bus Main Memory (M60X Command)</i> on page 41. |
| 3 | Local SDRAM | MLOC | See <i>Local Memory (MLOC Command)</i> on page 43. |
| 4 | CAM memory | MCAM | See <i>CAM (MCAM Command)</i> on page 43. |
| 5 | Serial EEPROM | MSRPROM | See <i>Serial EEPROM (MSRPROM Command)</i> on page 44. |
| 6 | CAM chip detection | Part of CHIPREV | See <i>CAM Detection Description</i> on page 47 on page. |
| 7 | Framers chip detection | Part of CHIPREV | See <i>Framers Detection Description</i> on page 46. |
| 8 | Ethernet chip detection | Part of CHIPREV | See <i>Ethernet Interface Detection Description</i> on page 46. |

Table 2-2. BIST Equivalent Tests For POST Tests (cont)

| No. | POST Test Title | BIST Equivalent Test | References/Comments |
|-----|--|----------------------|---|
| 9 | IMA chip detection | Part of CHIPREV | This test also checks the ATM over Fractional T1/E1/J1 register. See <i>IMA Chip Detection Description</i> on page 46. |
| 10 | Framers chip reset signal | FRST | See <i>T1/E1/J1 Framers Reset (FRST Command)</i> on page 47. |
| 11 | Ethernet chip reset signal | ERST | See <i>Ethernet Interface Reset (ERST Command)</i> on page 48. |
| 12 | IMA chip reset signal | IRST | See <i>Ethernet Interface Interrupt (EINT Command)</i> on page 50. |
| 13 | CAM chip reset signal | MCAMRST | See <i>CAM Reset (MCAMRST Command)</i> on page 49. |
| 14 | PCI chip local interrupt signal | PINT | See <i>PCI Controller Interrupt (PINT Command)</i> on page 52. |
| 15 | Framers chip interrupt signal | FINT | See <i>T1/E1/J1 Framers Interrupt (FINT Command)</i> on page 49. |
| 16 | Ethernet chip interrupt signal | EINT | See <i>Ethernet Interface Interrupt (EINT Command)</i> on page 50. |
| 17 | IMA chip interrupt signal | IINT | See <i>IMA Interrupt (IINT Command)</i> on page 51. |
| 18 | Framers internal loopback – independent mode | FIL IND | See <i>Framers In Internal Loopback (FIL Command)</i> on page 57. |
| 19 | Framers internal loopback – multiplexed mode | FIL MUL | See <i>Framers In Internal Loopback (FIL Command)</i> on page 57. |
| 20 | Framers pass-through mode (0->1,2->3) | FPT P0123 | See <i>Framers In Pass-Through Mode (FPT Command)</i> on page 62. |
| 21 | Framers pass-through mode (1->0,3->2) | FPT P1032 | See <i>Framers In Pass-Through Mode (FPT Command)</i> on page 62. |
| 22 | Ethernet internal loopback | EIL | See <i>Ethernet In Internal Loopback (EIL Command)</i> on page 66. |
| 23 | IMA port clocks | ICLK | See <i>IMA TDM Clocks (ICLK Command)</i> on page 78. |
| 24 | IMA port 0 internal loopback | IIL P0 | See <i>IMA Internal Loopback (IIL Command)</i> on page 68. |
| 25 | IMA port 1 internal loopback | IIL P1 | See <i>IMA Internal Loopback (IIL Command)</i> on page 68. |
| 26 | IMA port 2 internal loopback | IIL P2 | See <i>IMA Internal Loopback (IIL Command)</i> on page 68. |

Table 2-2. BIST Equivalent Tests For POST Tests (cont)

| No. | POST Test Title | BIST Equivalent Test | References/Comments |
|-----|------------------------------|----------------------|--|
| 27 | IMA port 3 internal loopback | IIL P3 | See <i>IMA Internal Loopback (IIL Command)</i> on page 68. |
| 28 | IMA SRAM | ISRAM | See <i>IMA Optional TDMs External Loopback (IEL8 Command)</i> on page 80. Note that, the BIST ISRAM test lasts nearly 25 seconds: in the POST a simpler version is run. |

Reporting Tests on the Console

The result of the POST is displayed on the consoles (TTY and PCI) as shown in [Example 2-1](#).

Example 2-1. Normal POST Result Display

```

***** POST *****
1 - Test Boot Firmware compatibility..... OK
2 - Test Main SDRAM ..... OK
3 - Test Local SDRAM ..... OK
4 - Test CAM memory ..... OK
5 - Test Serial EEPROM ..... OK
6 - Test CAM chip detection ..... OK
7 - Test Framers chip detection ..... OK
8 - Test Ethernet chip detection ..... OK
9 - Test IMA chip detection ..... OK
10 - Test Framers chip reset signal ..... OK
11 - Test Ethernet chip reset signal ..... OK
12 - Test IMA chip reset signal ..... OK
13 - Test CAM chip reset signal ..... OK
14 - Test PCI chip local interrupt signal.....OK
15 - Test Framers chip interrupt signal ..... OK
16 - Test Ethernet chip interrupt signal ..... OK
17 - Test IMA chip interrupt signal ..... OK
18 - Test Framers internal loop-back - independent mode .. OK
19 - Test Framers internal loop-back - multiplexed mode .. OK
20 - Test Framers pass-through mode (0->1,2->3) ..... OK
21 - Test Framers pass-through mode (1->0,3->2) ..... OK
22 - Test Ethernet internal loop-back ..... OK
23 - Test IMA ports clocks ..... OK
24 - Test IMA port 0 internal loop-back ..... OK
25 - Test IMA port 1 internal loop-back ..... OK
26 - Test IMA port 2 internal loop-back ..... OK
27 - Test IMA port 3 internal loop-back ..... OK
28 - Test IMA SRAM ..... OK
** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK **

```

At the end of the POST, a repeated text pattern on one line indicates the global result; if the POST is successful, the pattern is `**POST OK else ** POST FAILED`.

When a POST fails, details are displayed.

[Example 2-2](#) is an example of test '24 - Test IMA port 0 internal loop-back' that failed due to a time-out on port 0.

Example 2-2. Test 24 Failure

```
24 - Test IMA port 0 internal loop-back .....
    Port 0: failed - diag: timeout
    Failed
```



NOTE

If the 'Boot Firmware compatibility' test fails, the consoles are not used and the others tests are not run: after setting PowerSpan Mailbox register 2 to 0x00000001 (see [Reporting Tests with CPU LED](#) below), the Boot Firmware loops indefinitely.

Reporting Tests with CPU LED

The software programmable CPU LED (Yellow LED 5) on the front panel is used to report POST final results. If a test fails, the LED blinks quickly (every 125 ms) at the end of the POST. The user must use one of the available consoles to identify tests that failed (see [Reporting Tests on the Console on page 18](#)). If the POST succeeds, the CPU LED blinks normally (every 500 ms), signalling that the POST has successfully completed.

Reporting the Tests in a PCI Controller Register

The 4539 PCI chip has eight 32-bit mailbox registers. One of them, Mailbox Register 2, is used to report the POST result to the host. When a test fails, a bit is set in the register. Some tests may use the same bit to report a failure. Unused bits are set to 0. [Table 2-3](#) provides the mapping between the tests and the bits used in the register.

Table 2-3. POST Tests Map in PowerSpan MBR2 Register

| Test | Bit Mask |
|--------------------------------------|------------|
| 1 - Test Boot Firmware compatibility | 0x00000001 |
| 2 - Test Main SDRAM | 0x08000000 |
| 3 - Test Local SDRAM | 0x04000000 |
| 4 - Test CAM memory | 0x00004000 |
| 5 - Test Serial EEPROM | 0x00000002 |
| 6 - Test CAM chip detection | 0x00000004 |

Table 2-3. POST Tests Map in PowerSpan MBR2 Register (cont)

| Test | Bit Mask |
|--|------------|
| 7 - Test Framers chip detection | 0x00000010 |
| 8 - Test Ethernet chip detection | 0x00000020 |
| 9 - Test IMA chip detection | 0x00000040 |
| 10 - Test Framers chip reset signal | 0x00000100 |
| 11 - Test Ethernet chip reset signal | 0x00000200 |
| 12 - Test IMA chip reset signal | 0x00000400 |
| 13 - Test CAM chip reset signal | 0x00000800 |
| 14 - Test PCI chip local interrupt signal | 0x00001000 |
| 15 - Test Framers chip interrupt signal | 0x00002000 |
| 16 - Test Ethernet chip interrupt signal | 0x00010000 |
| 17 - Test IMA chip interrupt signal | 0x00020000 |
| 18 - Test Framers internal loopback - independent mode | 0x00080000 |
| 19 - Test Framers internal loopback - multiplexed mode | 0x00080000 |
| 20 - Test Framers pass-through mode (0->1,2->3) | 0x00080000 |
| 21 - Test Framers pass-through mode (1->0,3->2) | 0x00080000 |
| 22 - Test Ethernet internal loopback | 0x00100000 |
| 23 - Test IMA port clocks | 0x00200000 |
| 24 - Test IMA port 0 internal loopback | 0x00400000 |
| 25 - Test IMA port 1 internal loopback | 0x00400000 |
| 26 - Test IMA port 2 internal loopback | 0x00400000 |
| 27 - Test IMA port 3 internal loopback | 0x00400000 |
| 28 - Test IMA SRAM | 0x00800000 |

Setting the POST Requirement Mode (POST Command)

The user may chose between three types of POST:

- A normal POST (command `POST N`) that runs an exhaustive set of tests. [Example 2-1 on page 18](#) shows the resulting display of a normal POST on a 4539-000 board.
- A quick POST (command `POST Q`) that runs only the first five tests of the normal POST or less if the board is equipped with one CAM or not at all. [Example 2-3 on page 21](#) shows the result of a quick POST on a 4539-000 board.

- A POST reduced to the Boot Firmware compatibility test (command `POST D - D` means disable). [Example 2-4 on page 21](#) shows the result of a disabled POST.

Example 2-3. Quick POST Result Display

```
***** POST *****
1 - Test Boot Firmware compatibility..... OK
2 - Test Main SDRAM ..... OK
3 - Test Local SDRAM ..... OK
4 - Test CAM memory ..... OK
5 - Test Serial EEPROM ..... OK
** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK **
```

Example 2-4. Disable POST Result Display

```
***** POST *****
1 - Test Boot Firmware compatibility..... OK
** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK ** POST OK **
```


Monitor Configuration (SOFTWARE Command)

Description

The MONITOR software configuration is stored in the serial EEPROM. Its global content can be displayed with the SOFTWARE commands as shown below. Most of the individual parameters can be modified with a specific command.

```
>software

-- Eeprom Software Configuration version 1.03 --

Ethernet MAC address      : 00-00-77-96-A3-55
Local IP address          : 255.255.255.255
Mask IP address           : 255.255.255.255
Server IP address         : 255.255.255.255
Router IP address         : 255.255.255.255
Run operational firmware  : disable
TTY Parity                 : none
TTY Stop bits             : 1 stop bit
TTY Baudrate              : 9600 bauds
Local echo                 : enable
Power On Self Test        : normal
Firmware Address          : FF840100
Firmware Configuration Address: FF840100
>
```

Table 3-1. Commands to Change Software Configuration

| Information | Command | Serial EEPROM Corresponding Information Field (see Serial EEPROM Mapping on page 109) |
|---------------------------------------|--|--|
| EEPROM Software Configuration version | No individual command available | MONP_VMAJOR, MONP_VMINOR |
| Local IP address | LOCALIP - see Changing the Board IP Local Address (LOCALIP Command) on page 28 | MONP_LOCIPADD |
| Mask IP address | MASKIP - see Changing the Board IP Subnet Mask (MASKIP Command) on page 28 | MONP_MASKIPADD |

Table 3-1. Commands to Change Software Configuration (cont)

| Information | Command | Serial EEPROM Corresponding Information Field (see <i>Serial EEPROM Mapping</i> on page 109) |
|--------------------------------|---|--|
| Server IP address | SERVERIP - see <i>Changing the Server IP Address (SERVERIP Command)</i> on page 28 | MONP_SERVIPADD |
| Router IP address | ROUTERIP - see <i>Changing the Router IP Address (ROUTERIP Command)</i> on page 29 | MONP_ROUTIPADD |
| Run Operational Firmware | AUTORUN - see <i>AUTORUN Command</i> on page 31 | MONP_RUN_FW |
| TTY Parity | PARITY - see <i>Changing the Parity (PARITY Command)</i> on page 26 | MONP_PARITY |
| TTY Stop bits | STOPBIT - see <i>Changing the Stop Bits (STOPBIT Command)</i> on page 26 | MONP_STOPBITS |
| TTY Baudrate | BAUDRATE - see <i>Changing the Baud Rate (BAUDRATE Command)</i> on page 26 | MONP_BAUDRATE |
| Local echo | ECHO - see <i>Changing the Echo Mode (ECHO Command)</i> on page 27 | MONP_ECHO |
| Power-On Self Test | POST - see the chapter titled <i>Setting the POST Requirement Mode</i> in the <i>4539 Board Installation and Maintenance Manual</i> | MONP_QUICK_POST |
| Firmware Address | FWADDR - see <i>FWADDR Command</i> on page 31 | MONP_FWADDR |
| Firmware Configuration Address | FWCFGADDR - see <i>FWCFGADDR Command</i> on page 32 | MONP_FWCFGADDR |

SOFTWARE Command

Name: SOFTWARE

Syntax: `software|cfg [d]`

Description: Displays the current serial EEPROM software configuration or flashes a default one (except for MAC addresses that cannot be modified):

| | |
|--------------------------------|-------------|
| Ethernet MAC Address | (mac) |
| Local IP address | (localip) |
| Mask IP address | (maskip) |
| Server IP address | (serverip) |
| Router IP address | (routerip) |
| Run operational firmware | (run) |
| TTY parity | (parity) |
| TTY stop bits | (stopbit) |
| TTY baudrate | (baudrate) |
| Local echo | (echo) |
| Power on self test | (post) |
| Firmware address | (fwaddr) |
| Firmware configuration address | (fwcfgaddr) |

Options: a Restore the default software configuration:

| | |
|--------------------------------|-----------------|
| Local IP address: | 255.255.255.255 |
| Mask IP address: | 255.255.255.255 |
| Server IP address | 255.255.255.255 |
| Router IP address | 255.255.255.255 |
| Run operational firmware | disable |
| TTY parity | none |
| TTY stop bits | 1 stop bit |
| TTY baudrate | 9600 Baud |
| Local echo | enable |
| Power on self test | normal |
| Firmware address | invalid |
| Firmware configuration address | invalid |

TTY Parameters

Overview

The following are the TTY parameters (with associated command in parenthesis):

- Baud rate (BAUDRATE)
- Parity (PARITY)
- Stop Bits (STOPBIT)
- Echo (ECHO)

Baud Rate, Parity, and Stop bits can be changed using the corresponding command, but the changes apply on the next reset. Echo modification has immediate effect.

Changing the Baud Rate (BAUDRATE Command)

Name: BAUDRATE

Syntax: baudrate|br [mode]

Description: Displays the TTY console baud-rate selection if no parameter is specified, or changes it if specified.

Option: mode Speed selection as follows:

| | |
|--------|-------------|
| 9600 | 9600 baud |
| 19200 | 19200 baud |
| 38400 | 38400 baud |
| 57600 | 57300 baud |
| 115200 | 115200 Baud |

Examples:

```
>br
TTY Baudrate      : 9600 bauds
>br 19200
>br
TTY Baudrate      : 19200 bauds
```

Changing the Parity (PARITY Command)

Name: PARITY

Syntax: parity|p [mode]

Description: Displays the TTY console parity selection if no parameter is specified, or changes it if specified.

Option: mode Parity mode as follows:

| | |
|---|-------------|
| n | No parity |
| o | Odd parity |
| e | Even parity |

Examples:

```
>p
TTY Parity        : none
>p o
>p
TTY Parity        : odd
```

Changing the Stop Bits (STOPBIT Command)

Name: STOPBIT

Syntax: stopbit|sb [number]

| | |
|--------------|--|
| Description: | Displays the TTY console stop bit setting if no parameter is specified, or changes it if specified. |
| Option: | number Number of stop bits; 1 or 2. |
| Examples: | <pre>>sb >TTY Stop bits : 1 stop bit >sb 2 >sb >TTY Stop bits : 2 stop bits</pre> |

Changing the Echo Mode (ECHO Command)

| | |
|--------------|--|
| Name: | ECHO |
| Syntax: | echole [mode] |
| Description: | Displays the console local echo selection if no parameter is specified, or changes it if specified. |
| Option: | mode Echo mode as follows: <ul style="list-style-type: none"> e Local echo is enabled d Local echo is disabled |
| Examples: | <pre>>e Local echo : enable >e d >e Local Echo : disable</pre> |

IP/MAC Addresses Commands

Overview

IP/MAC Addresses Commands allows displaying/modifying some parameters stored in the Serial EEPROM. The parameters are the following (the Serial EEPROM configuration parameter mnemonics are provided in parenthesis—see [Serial EEPROM Mapping on page 109](#)):

- MAC address (MONP_MACCADD)
- Board Local IP Address (MONP_LOCIPADD)
- Board IP Subnet mask (MONP_MASKIPADD)
- Server IP Address (MONP_SERVIPADD)
- Router IP Address (MONP_ROUTIPADD) (This parameter is not used by the Boot Firmware but is relevant for further use)

MAC address parameter cannot be modified unless using [Serial EEPROM Debugging Commands on page 29](#).

Displaying the Board Ethernet MAC Address (MAC Command)

Name: MAC
Description: Displays the board MAC address.
Syntax: mac

Changing the Board IP Local Address (LOCALIP Command)

Name: LOCALIP
Syntax: localip|lip [addr]
Description: Displays the local IP address if no parameter is specified, or changes it if specified.
Option: addr New board local IP address, in decimal with dot separation.
Example:

```
>lip
Local IP address : 255.255.255.255
>lip 192.134.47.90
>lip
Local IP address : 192.134.47.90
```

Changing the Board IP Subnet Mask (MASKIP Command)

Name: MASKIP
Syntax: maskip|mip [addr]
Description: Displays the board subnet mask if no parameter is specified, or changes it if specified.
Option: addr New board subnet mask in decimal with dot separation.
Example:

```
>mip
Mask IP address : 0.0.0.0
>mip 255.255.255.0
>mip
Mask IP address : 255.255.255.0
```

Changing the Server IP Address (SERVERIP Command)

Name: SERVERIP
Syntax: serverip|sip [addr]
Description: Displays the board server IP address if no parameter is specified, or changes it if specified.
Option: addr New board server IP address in decimal with dot separation.

```

Example:      >sip
              Server IP address   : 255.255.255.255
              >sip 192.134.47.90
              >sip
              Server IP address   : 192.134.47.90

```

Changing the Router IP Address (ROUTERIP Command)

```

Name:         ROUTERIP
Syntax:       routerip|rip [addr]
Description:  Displays the board default gateway address if no parameter is
              specified, or changes it if specified.
Option:       addr      New board router IP address in decimal with dot separation
Example:      >rip
              Router IP address   : 255.255.255.255
              >rip 192.134.47.90
              >rip
              Router IP address   : 192.134.47.90

```

Serial EEPROM Debugging Commands

Overview

Serial EEPROM Debugging commands allow reading or writing bytes in the Serial EEPROM:

- SREAD
- SWRITE
- SDUMP

User should not need to modify the Serial EEPROM content using the SWRITE command unless using some locations that are not used by 4539 Boot Firmware (but a future release of 4539 Boot Firmware might use some locations that currently are free). When the content of the Serial EEPROM is modified, its checksum (MONP_CHECKSUM) is automatically updated. Note these commands start with the letter **S** for Serial.

Reading a Serial EEPROM Location (SREAD Command)

```

Name:         SREAD
Syntax:       sread|sr addr
Description:  Reads one byte at a specified serial EEPROM address. The returned
              value is in hexadecimal.

```

`addr` Serial EEPROM address to read, in hexadecimal (without 0x).

Example: `>sr b0`
`B0 -> FF`

Writing a Serial EEPROM Location (SWRITE Command)

Name: `SWRITE`

Syntax: `swrite|sw addr value`

Description: Writes one byte at the specified serial EEPROM address.

`addr` Serial EEPROM address to write to, in hexadecimal (without 0x).

`value` Value to write, in hexadecimal (without 0x).

Example: `>sw b0 ff`

Dumping a Serial EEPROM Block (SDUMP Command)

Name: `SDUMP`

Syntax: `sdump |sd [addr length]`

Description: Dumps the entire serial EEPROM memory if no parameter is specified or a length-wide block memory starting at the specified address (`addr`).

Options: `addr` Serial EEPROM block memory starting address (in hexadecimal without 0x)

`length` Serial EEPROM block memory size in bytes (in decimal)

Example: `>sd b0 6`
EEPROM memory: 00 00 77 95 39 DB

POST Command

Name: `POST`

Syntax: `post [mode]`

Description: Displays POST requirement mode if no parameter is specified, or changes it as required.

Three types of POST are available:

- Normal (n): Performs an exhaustive test of the board.
- Quick (q): Performs the following tests:
 - Boot firmware compatibility
 - Main SDRAM

- Local SDRAM

- CAM

- Serial EEPROM

- Disabled (d): Performs only the boot firmware compatibility test.

Options: mode (n, q, d)

n Normal POST

q Quick POST

d Disable POST

Examples: **>post**
 Power On Self Test : normal
>post q
 Power On Self Test : quick

FWADDR Command

Name: FWADDR

Syntax fwa|fwaddr [Addr]

Description: Displays the firmware address in flash if no parameter is specified or changes it if specified.

Option: Addr New firmware address in flash in hexadecimal (without 0x).

Examples: **>fwa**
 firmware address : FF840100
>fwa ff860000
>fwa
 firmware address : FF860000

AUTORUN Command

Name: AUTORUN

Syntax: autorun|ar|run [mode]

Description: Displays the firmware start requirement if no parameter is specified or changes it if specified.

Options: mode (d | e | f)

d Firmware autorun disable

e Firmware autorun from Ethernet

f Firmware autorun in flash at FWADDR location

Examples: **>run**
 run operational firmware : disable
 >run f
 >run
 run operational firmware : in flash

FWCFGADDR Command

The Interphase operational firmware, also called on-board software, needs to be configured. This configuration can be done by the host through the PCI bus or by the on-board software itself that reads the address of the configuration in the Serial EEPROM. The firmware configuration address can be read or modified with this command.

Name: FWCFGADDR

Syntax: fwcfgaddr|fwca [addr]

Description: Displays the firmware configuration address in flash if no parameter is specified, or changes it if specified.

Option: addr New firmware configuration address in flash expressed in Hexadecimal (without 0x).

Examples: **>fwca**
 Firmware Configuration Address: FF840000
 >fwca ff8e0000
 >fwca
 Firmware Configuration Address: FF8E0000

INFO Command

This command displays board general information as shown in [Example 4-1](#). The various items listed are described in [Table 4-1](#).

Example 4-1. General Information Display

```
>info
General Information:

Identification      : 4539-000 or 4539-001
Hardware revision   : 0
Serial Number       : 1428130
Manufacturing date  : 20/12/2000
EPLD                : SX00598-A00
Boot firmware       : SX00595-A00 V1.00
  - Start Address   : 0xFFFF00100
  - Stop Address    : 0xFFFF359E0
  - Size (in bytes) : 0x000358E0
Ports               : Ethernet 10/100
                    : 4 T1/E1/J1 on P4 (Rear Access)
                    : ATM UTOPIA on P3 - PTENB# asserted (=0)
                    : 4 IMA/UNI TDMs on P4
MAC address         : 00.00.77.95.CA.A2
POST result         : OK (0x00000000)
>
```

Table 4-1. INFO Command Field Descriptions

| Field | Description |
|--------------------|---|
| Identification | Board commercial identification (obtained from the board hard-coded revision register). |
| Hardware revision | Board revision number (obtained from the board hard-coded revision register). |
| Serial number | Board serial number. Matched to CARD_SERIAL_L and CARD_SERIAL_H fields in the Serial EEPROM. |
| Manufacturing date | First serial EEPROM programming date. This information is the CARD_DATE_LOC field in the Serial EEPROM. |
| EPLD | Version number of the file used to program the board's Erasable Programmable Logic Device (EPLD). This information is the ISP_SX_LOC field in the Serial EEPROM. If the EEPROM was not programmed using standard Interphase manufacturing processes, the date of programming located in ISP_UPDATE_LOC is displayed instead. |
| Boot firmware | Version number of the file used to program the 4539 Boot Firmware. This information is the BOO_SX_LOC field in the Serial EEPROM. If the Boot Firmware was not programmed using standard Interphase manufacturing processes, the date of programming located in BOO_UPDATE_LOC is displayed instead. The Boot Firmware version, start address, stop address, and size in FLASH EEPROM are also displayed (this information is coded in the Boot Firmware itself). |

Table 4-1. INFO Command Field Descriptions (cont)

| Field | Description |
|--------------|--|
| Ports | Indicates the type and number of ports. PTENB# signal assertion informs the user that the carrier board has enabled UTOPIA signals on P3. IMA/UNI TDMs on P4 PMC connector are optional and so may be not displayed. |
| MAC address | Ethernet MAC address |
| POST result | POST result (OK or Failed) with corresponding 32-bit value stored in Mailbox Register 2 Of of PCI Controller. For more information, see Chapter 2 Power-On Self Test (POST) on page 15 . |

INFOEQU Command

This command displays board equipment information retrieved from the Serial EEPROM Board Equipment register (see *4539 Hardware Reference Manual*) as shown in [Example 4-2](#), [Example 4-3](#) and [Example 4-4](#)

Example 4-2. Equipment Information Display on a 4539-000 or 4549-001

```
>infoequ
Equipment Information:

  Micro-processor identification : MPC8260ZU200, 200/133/66MHz, rev A.1
  Local bus frequency           : 65.536 MHz
  FLASH size                    : 4 Mbytes
  Main SDRAM size               : 64 Mbytes
  Local SDRAM size              : 8 Mbytes
  CAM size                      : 16 K x 64
  IMA SRAM size                 : 512 Kbytes
  IMA device                    : MT90221 (4 IMA TDMs)
>
```

Example 4-3. Equipment Information Display on a 4539-002 or 4539-003

```
>infoequ
Equipment Information:

  Micro-processor identification : MPC8260ZU200, 200/133/66MHz, rev A.1
  Local bus frequency           : 65.536 MHz
  FLASH size                    : 4 Mbytes
  Main SDRAM size               : 64 Mbytes
  Local SDRAM size              : no memory device
  CAM size                      : no CAM device
  IMA SRAM size                 : no memory device
  IMA device                    : no device
>
```

Example 4-4. Equipment Information Display on a 4539-004 or 4539-005

```
>infoequ
Equipment Information:

  Micro-processor identification : MPC8260ZU133, 133/133/66MHz, rev A.1
  Local bus frequency           : 65.536 MHz
  FLASH size                    : 4 Mbytes
  Main SDRAM size               : 32 Mbytes
  Local SDRAM size              : no memory device
  CAM size                      : no CAM device
  IMA SRAM size                 : no memory device
  IMA device                    : no device
>
```

INFOPCI Command

This command displays PCI general information as shown in [Example 4-5](#). The various items listed are described in [Table 4-2](#).

Example 4-5. PCI General Information Display

```
>infopci

PCI Information:

Vendor / Device Id : 107E/90A0
Revision Id       : 0
PCI EEPROM       : SX0597-A00

Base Registers address/size:

BAR 0 = 0x02010000 / 64 KBytes
BAR 1 = 0x02000000 / 4 KBytes
BAR 2 = 0x00000000 / 2 MBytes
BAR 3 = 0x02400000 / 512 KBytes
BAR 4 = disabled
BAR 5 = disabled

>
```

Table 4-2. INFOPCI Command Field Descriptions

| Field | Description |
|-----------------------------|--|
| Vendor Id | Interphase PCI vendor identifier |
| Device Id | 4539 PCI device identifier |
| Revision Id | 4539 PCI revision identifier. |
| PCI EEPROM | Version number of the file used to program the PowerSpan initial register values in the Serial EEPROM (Addresses 0x00 to 0x3F). This information is the PSP_SX_LOC field in the Serial EEPROM. If the EEPROM was not programmed using standard Interphase manufacturing processes, the date of programming located in PSP_UPDATE_LOC is displayed instead. |
| Base Registers address/size | BAR 0: I ² O registers base address and size in PCI memory space BAR 1: PowerSpan internal registers base address and size in PCI memory space BAR 2: PCI-to-local window 0 base address and size in PCI memory space BAR 3: PCI-to-local window 1 base address and size in PCI memory space BAR 4: PCI-to-local window 2 base address and size in PCI memory space BAR 5: PCI-to-local window 3 base address and size in PCI memory space Note: disabled indicates that the corresponding BAR is not used. |

CHIPREV Command

This command detects the chips and displays their versions as shown in [Example 4-6](#).

- Processor: Motorola MPC8260
- PCI Chip interface: Tundra PowerSpan
- T1/E1/J1 Quad Framer: Infineon QuadFALC™
- Ethernet interface: Level One LXT971
- IMA chip: MITEL MT90220/1 (only the revision of the chip is displayed - the type of the chip (MT90220 or MT90221) is displayed by the INFOEQU command - for boards that are not equipped with an IMA chip, the IMA detection information is not displayed).
- CAM memory (No revision number available - for boards that are not equipped with a CAM, the CAM detection information is not displayed).

Example 4-6. Chip Revision Display

```
>chiprev  
  
Chips detection and revision:  
  
MPC8260 : Revision = A1  
PowerSpan: Revision = 1  
QuadFALC: detected - Revision = 1.3  
Ethernet: detected - Revision = 1  
IMA: detected - Revision = 1  
CAM: detected  
  
>
```


Introduction

The MONITOR test category is divided into sub-categories (Memories, Framers, Ethernet, IMA, and Others). Each test command starts with a letter that matches the first letter of the sub-category it belongs to, except for the Others sub-category commands.

- M for Memories
- F for Framers
- E for Ethernet
- I for IMA
- P for PCI Controller

Some groups of letters also indicate what a command is for:

- IL for Internal Loopback test
- EL for External Loopback test
- INT for Chip Interrupt test
- RST for Chip Reset test

Examples:

- FIL command executes a Framers Internal Loopback test
- EEL command executes a Ethernet External Loopback test
- IINT command executes a IMA Chip Interrupt test
- FRST command executes a Framers Chip Reset test

Depending on the 4539 equipment option, some tests are not available. In [Table 5-1](#), X indicate the tests available for each 4539-XXX version (4539-001, 4539-002, etc.). Note that boards equipped with MITEL MT90220 IMA chip (8TDMs) are engineering versions which explains they have no reference (IMA8)

Table 5-1. BIST Tests

| Title | Command | Version | | | |
|-----------------------|---------|-------------|-------------|-------------|------|
| | | 000/ 001 | 002/ 003 | 004/ 005 | IMA8 |
| Main SDRAM | M60X | X | X | X | X |
| Local SDRAM | MLOC | X | | | X |
| CAM memory | MCAM | X | | | X |
| Serial EEPROM | MSRPROM | X | X | X | X |
| FLASH EEPROM | MFPROM | X | X | X | X |
| CAM chip reset signal | MCAMRST | X | | | X |

Table 5-1. BIST Tests (cont)

| Title | Command | Version | | | |
|-------------------------------------|---------|-------------|-------------|-------------|------|
| | | 000/ 001 | 002/ 003 | 004/ 005 | IMA8 |
| Framers BERT | FB | X | X | X | X |
| Framers in external loopback | FEL | X | X | X | X |
| Framers in internal loopback | FIL | X | X | X | X |
| Framers chip interrupt signal | FINT | X | X | X | X |
| Framers line status | FPHY | X | X | X | X |
| Framers pass-through mode | FPT | X | X | X | X |
| Framers in remote loopback | FRL | X | X | X | X |
| Framers chip reset signal | FRST | X | X | X | X |
| Framers switched mode | FSW | X | X | X | X |
| Ethernet in internal loopback | EIL | X | X | X | X |
| Ethernet in external loopback | EEL | X | X | X | X |
| Ethernet chip interrupt signal | EINT | X | X | X | X |
| Ethernet chip reset signal | ERST | X | X | X | X |
| IMA ports clocks | ICLK | X | | | X |
| IMA optional TDMs clocks | ICLK8 | | | | X |
| IMA ports external loopback | IEL | X | | | X |
| IMA optional TDMs external loopback | IEL8 | | | | X |
| IMA ports internal loopback | IIL | X | | | X |
| IMA chip interrupt signal | IINT | X | | | X |
| IMA chip reset signal | IRST | X | | | X |
| IMA SRAM | ISRAM | X | | | X |
| LED | LED | X | X | X | X |
| Product tests | PROD | X | X | X | X |
| PCI controller interrupt | PINT | X | X | X | X |
| UTOPIA external loopback on P3 | UEL | X | X | X | X |
| Agency | AGENCY | X | X | X | X |

Memories

60x Bus Main Memory (M60X Command)

Test Description

The test fills in the memory with pseudo-random patterns, and then checks the patterns by reading all memory. This sequence is run twice with different pseudo-random sequences.

The pseudo-random sequences are generated from four words that are the first pattern of the four first memory locations and an increment applied on each word. The increment equals $2*3*5*7*11*13*19*23-1$ ($=0x04DC2085$).

The memory start address is 0x80000000 (non-cachable mapping is used - see the *4539 Hardware Reference Manual*).

Table 5-2. Memory Pseudo-Random Pattern 1

| Address | Pattern |
|------------|----------------------------------|
| 0x80000000 | 0x55555555 |
| 0x80000004 | 0xAAAAAAAA |
| 0x80000008 | 0x66666666 |
| 0x8000000C | 0x99999999 |
| 0x80000010 | 0x55555555+0x04DC2085 |
| 0x80000014 | 0xAAAAAAAA+0x04DC2085 |
| 0x80000018 | 0x66666666+0x04DC2085 |
| 0x8000001C | 0x99999999+0x04DC2085 |
| 0x80000020 | 0x55555555+0x04DC2085+0x04DC2085 |
| 0x80000024 | 0xAAAAAAAA+0x04DC2085+0x04DC2085 |
| 0x80000028 | 0x66666666+0x04DC2085+0x04DC2085 |
| 0x8000002C | 0x99999999+0x04DC2085+0x04DC2085 |
| ... | ... |
| ... | ... |

Table 5-3. Memory Pseudo-Random Pattern 2

| Address | Pattern |
|------------|------------|
| 0x80000000 | 0xAAAAAAAA |
| 0x80000004 | 0x55555555 |
| 0x80000008 | 0x99999999 |

Table 5-3. Memory Pseudo-Random Pattern 2 (cont)

| Address | Pattern |
|------------|----------------------------------|
| 0x8000000C | 0x66666666 |
| 0x80000010 | 0xAAAAAAAA+0x04DC2085 |
| 0x80000014 | 0x55555555+0x04DC2085 |
| 0x80000018 | 0x99999999+0x04DC2085 |
| 0x8000001C | 0x66666666+0x04DC2085 |
| 0x80000020 | 0xAAAAAAAA+0x04DC2085+0x04DC2085 |
| 0x80000024 | 0x55555555+0x04DC2085+0x04DC2085 |
| 0x80000028 | 0x99999999+0x04DC2085+0x04DC2085 |
| 0x8000002C | 0x66666666+0x04DC2085+0x04DC2085 |
| ... | ... |
| ... | ... |

When an error occurs, the test reports the error and stops the test as shown in the following example:

```
>M60X
    Access error : address = 0x8000000C ; expected= 0x66666666 ; read= 0x00000000
    Test completed : failed
>
```

M60X Command

- Name: M60X
- Syntax: m60x [-A Addr] [-S Size]
- Description: Tests the 60x bus main SDRAM memory. A global result is displayed with intermediate error messages in case of errors. If erroneous parameters are entered, the whole memory is tested.
- Options:
 - A Addr The block start address where the test is to be performed. Expressed in Hexadecimal (without 0x). Addr has to be 32 bits aligned. If Addr does not fit, a default SDRAM address is used.
 - S Size The block number under test expressed in decimal. Size has to be 4*32 bits aligned. If Size does not fit, a default SDRAM size is used.

Local Memory (MLOC Command)

Test Description

The local memory is checked by using the same algorithm as for the main memory (see [60x Bus Main Memory \(M60X Command\)](#) on page 41). The memory start address is 0x88000000 (non-cachable mapping is used - see the *4539 Hardware Reference Manual*).

MLOC Command

Name: MLOC

Syntax: `mloc [-A Addr] [-S Size]`

Description: Tests the local SDRAM. A global result is displayed with intermediate error messages in case of errors.

Options:

- A *Addr* The block start address where the test is to be performed. Expressed in Hexadecimal (without 0x). *Addr* has to be 32 bits aligned. If *Addr* does not fit, a default SDRAM address is used.
- S *Size* The block number under test expressed in decimal. *Size* has to be 4*32 bits aligned. If *Size* does not fit, a default SDRAM size is used.

CAM (MCAM Command)

Test Description

This test checks CAM memory. The CAM is first filled with entries defined in [Table 5-4](#). If the test cannot insert as many entries as the CAM size, the test fails. Then the test searches the channel that matches each header, by using the control port and the match port. The same search is performed on the headers. At the end, the CAM is emptied, and the test verifies that there are no more entries in the CAM.

Table 5-4. CAM Entries for MCAM Test

| Entry Index | Entry Header | Entry Channel |
|-------------|------------------|--------------------|
| $2*i$ | $0x06000000+2*i$ | $0x02000000+2*i$ |
| $2*i+1$ | $0x09000000+2*i$ | $0x02000000+2*i+1$ |

MCAM Command

Name: MCAM

Syntax: `mcam`

Description: Tests the CAM memory. A global result is displayed with intermediate error messages in case of errors.

Serial EEPROM (MSRPROM Command)

Test Description

This test checks the Serial EEPROM checksum stored in its last 2 bytes (see MONP_CHECKSUM in [Serial EEPROM Mapping on page 109](#)).

If the test failed, the computed checksum and the checksum in the Serial EEPROM are displayed. If the test is successful, just `Test completed : OK` is displayed.

The following 'C' code shows the checksum computation algorithm:

```
/* Checksum as computed */
unsigned short Checksum;

/* Buffer containing a copy of EEPROM content*/
unsigned char EEPROMContent [256];

/* Cumulative 32 bits sum initialize to 0 */
unsigned int Sum=0;

/* Loop counter */
word Loop;

for(Loop = 0; Loop < 254; Loop +=2 ) {
Sum += ((unsigned short) (EEPROMContent [Loop]))<<16;
Sum += (unsigned short) (EEPROMContent [Loop+1]);
}
/* Add back carry out from 32 bits to low 16 bits */
Sum = (Sum >> 16) + (Sum &0xFFFF);

/* Add carry */
Sum += (Sum>>16);

/* Ones-complement, then truncate to 16 bits to have checksum */
Checksum = (unsigned short) (~Sum);

/* Checksum update */
EEPROMContent [254]=(unsigned char) (Checksum>>16);
EEPROMContent [255]=(unsigned char) (Checksum&0x00FF);
```

MSRPROM Command

| | |
|--------------|---|
| Name: | MSRPROM |
| Syntax: | msrprom |
| Description: | This command computes the checksum of the serial EEPROM and compares it with the one stored in it. In case of error, the stored checksum and the computed checksum are displayed, else OK is displayed. |

FLASH EEPROM (MFPROM Command)

Test Description

FLASH EEPROM is divided into 64 kByte blocks that are numbered from 0. Test is performing on one block or all of them, using the same algorithm as SDRAM memories, but only one access is done to avoid saturating the allowed erase/read/write access of the EEPROM.

To write a byte at a specified address in FLASH EEPROM, the old value has to be 0xFF. Otherwise, the whole corresponding sector has to be erased (0xFF is written in the sector when it is erased). This is a very destructive test, and there is no way to recover memory if power fails during it.

MFPROM Command

| | |
|--------------|---|
| Name: | MFPROM |
| Syntax: | mfprom [-S Sector] |
| Description: | This command tests the Flash EEPROM. Flash sectors that contain Boot Firmware code are not tested, though test completed result is displayed. |
| Option: | -S Sector Block number to test. If no parameter or 0 is entered, the whole FLASH is tested. |



WARNING

MFPROM is a very destructive test, if a sector that contains a part of needed information is under test, it is erased at the beginning of the test, and restored at the end. If power is lost during the test, or if an external event stops the test before its completion, there is no way to recover the sector.

Chip Detection (CHIPREV Command)

Test Description

This command detects the chips and displays their versions.

The chips are:

- Processor: Motorola MPC8260
- PCI Chip interface: Tundra PowerSpan
- T1/E1/J1 Quad Framer: Infineon QuadFALC™
- Ethernet interface: Level One LXT971
- IMA chip: MITEL MT90220 or MT90221
- CAM memory (No revision number available)

(For IMA only, the revision of the chip is displayed—the type of the chip (MT90220 or MT90221) is displayed by the INFOEQU command—for boards that are not equipped with an IMA chip, the IMA detection information is not displayed.)

Processor Detection Description

The processor is not really detected: the code itself would not run if the processor was not there. In fact, just the version from the IMMR register is displayed in ACSII format for known processor versions at the time the 4539 Boot Firmware version was released, otherwise the revision is displayed as a numerical value.

PCI Chip Interface Detection Description

The PCI chip interface is not detected. Just its version located in register CSR is displayed.

Framers Detection Description

The test writes the values 0xAA and 0x55 in registers PCD and PCR respectively and then the registers are read to verify that the values were correctly written. The version is read from the VSTR register.

Ethernet Interface Detection Description

The test writes the value 0xA5A5 in the LED Configuration register (0x14) and checks that the register content is what was written. The revision is read from the PHY identification registers (0x02 and 0x03).

IMA Chip Detection Description

The test writes 0x55 in Tx Links 1 and 2 FIFO register (0x14A) and reads the register to check the value. The same operation is performed with value 0xAA. The revision is read from the General Status register (0x206).

CAM Detection Description

The test writes values 0x5555, 0xAAAA, 0x6666, and 0x9999, in registers IO0 (0x0000), IO1 (0x0002), IO2 (0x0004), IO3 (0x0006) respectively and checks the values by reading the registers. No revision register is available.

CHIPREV Command

| | |
|--------------|--|
| Name: | CHIPREV |
| Syntax: | chiprev |
| Description: | <p>This command detects and displays the revision numbers of the following chips:</p> <ul style="list-style-type: none"> - Processor: Motorola MPC8260 - PCI Chip interface: Tundra PowerSpan - T1/E1/J1 Quad Framer: Infineon QuadFALC™ - Ethernet interface: Level One LXT971 - IMA chip: MITEL MT90220 or MT90221 - CAM memory (No revision number available) |

Chip Reset

Overview

Chips on the 4539 board have a reset pin connected to a PowerQUICC II Parallel Port pin. The purpose of a chip reset test is to test the connectivity between the chip reset pin and the corresponding PowerQUICC II parallel port pin. This is done by setting the chip to a state different from the reset state and by verifying that after reset (assertion/deassertion of the corresponding Parallel Port pin), the chip is in reset state.

T1/E1/J1 Framers Reset (FRST Command)

Test Description

The reset test of a quad framers is performed in two steps:

1. The quad-framers are first detected – if this test succeeds, PCD and PCR register contents are 0xAA and 0x55 respectively (see [Chip Detection \(CHIPREV Command\)](#) on page 46).
2. The reset signal is asserted and deasserted (this is done by toggling the corresponding MPC8260 Port C data pin) and the test verifies that PCD and PCR registers have their reset value: 0x00. Note that the quad-framers have their individual corresponding reset pins on MPC8260 Port C.

FRST Command

Name: FRST

Syntax: `frst`

Description: Tests the framers chips reset signals. One QuadFALC chip handles four framers.

Ethernet Interface Reset (ERST Command)

Test Description

This test is performed in two steps.

1. A value different from the reset value is written in a register, and the test verifies it by reading the register.
2. The test resets the Ethernet interface with corresponding MPC8260 port D pin and verifies that the content of the register read in step 1 is the reset value.

The register is the LED Configuration register (0x14) (written value is 0xA5A4 and reset value is 0x0422).

ERST Command

Name: ERST

Syntax: `erst`

Description: Tests the Ethernet chip reset signal.

IMA Reset (IRST Command)

Test Description

This test is performed in two steps.

1. A value different from the reset value is written in a register, and the test verifies it by reading the register.
2. The test resets the IMA chip with corresponding MPC8260 port C pin and verifies that the content of the register read in step 1 is the reset value.

The register used is Tx Links 1 and 2 FIFO register (0x14A) (written value is 0x55 and reset value is 0x33).

IRST Command

Name: IRST

Syntax: `irst`

Description: Test the IMA chip reset signal.

CAM Reset (MCAMRST Command)

Test Description

The 4539 boards have two CAM memories. The CAM chips have a common reset signal from a pin of MPC8260 port D. The test is performed on both CAM memories.

The test of a CAM is performed in three steps:

1. The CAM is reset and the test verifies that no entry is in the CAM.
2. An entry is inserted in the CAM and the test verifies that there is one entry in the CAM.
3. The CAM is reset and the test verifies that no entry is in the CAM.

MCAMRST Command

Name: MCAMRST
 Syntax: `mcamrst`
 Description: Test CAM chips reset signals.

Chip Interrupt

Overview

The chips on the 4539 communication controller have an interrupt pin connected to either an MPC8260 Port C pin or IRQx pin. The purpose of a chip interrupt test is to test the connectivity between the chip interrupt pin and the corresponding MPC8260 pin. This is done by setting the chip in a state so that it generates an interrupt, and by verifying that the corresponding external interrupt event is generated on the MPC8260, we also verify that when the interrupt on the chip is served, the corresponding event on the MPC8260 disappears.

T1/E1/J1 Framers Interrupt (FINT Command)

Test Description

This test verifies the connectivity between the INT pin of a QuadFALC and the corresponding IRQ3 pin of the MPC8260.

The test is performed by using the QuadFALC alarm simulation feature. The alarm simulation state machine goes through eight states by setting/resetting the FRM0.SIM bit. At each state, some alarms are generated. For this test, only the LOS alarm (Loss Of Signal) which is unmasked (others are masked), is checked.

In a state for which LOS is generated (state 1 for example), the following operations are performed:

1. Verify that the QuadFALC INT pin is asserted by reading the GIS.INT bit that must be set.
2. Verify that the IRQ3 MPC8260 pin is asserted by reading the corresponding bit in SIPNR_H, that must be set.
3. Read the QuadFALC ISR2 register and check that the LOS bit is set, signalling LOS alarm – this read operation cancels the LOS alarm.
4. Read the QuadFALC ISR2 register again and check that the LOS bit is reset, signalling that the LOS alarm is not present.
5. Verify the QuadFALC INT pin is not asserted by reading the GIS.INT bit that must be set to 0.
6. Clear pending IRQ3 interrupts by setting the corresponding bit in MPC8260 SIPNR_H register.
7. Verify that the IRQ3 MPC8260 pin is deasserted by reading the corresponding bit in SIPNR_H that must be set to 0.

On a state for which LOS is not generated (state 3 for example), the following operations are performed:

1. Read QuadFALC ISR2 register and check that the LOS bit is not set.
2. Verify the QuadFALC INT pin is not asserted by reading the GIS.INT bit, that must be set to 0.

FINT Command

| | |
|--------------|---|
| Name: | FINT |
| Syntax: | fint |
| Description: | Tests the T1/E1/J1 framer chip interrupt signals. |

Ethernet Interface Interrupt (EINT Command)

Test Description

This test verifies the connectivity between the INT pin of a Level One LXT971A chip MDINT and the corresponding IRQ2 pin of MPC8260. The chip is programmed to generate an interrupt and the test checks that an interrupt is received.

The following steps are performed:

1. Enable interrupts by setting INTEN bit of Interrupt Enable Register.
2. Force interrupt on MDINT pin by setting TINT bit of Interrupt Enable Register.
3. Wait for MDINT corresponding pin on IRQ2 to be set to 0 (MDINT asserted)—return test failure on time-out.
4. Clear MDINT interrupt (Clear TINT bit in Interrupt Enable register, read Status register, read Interrupt Status register.).
5. Wait for MDINT corresponding pin on IRQ2 to be set to 1 (MDINT deasserted)—return test failure on time-out.

EINT Command

| | |
|--------------|---|
| Name: | EINT |
| Syntax: | eint |
| Description: | Tests the Ethernet chip interrupt signal. |

IMA Interrupt (IINT Command)

Test Description

This test verifies the connectivity between the UP_IRQ pin of the IMA chip and the IRQ7 pin on MPC8260. For this test, PCM Link 1 of IMA chip is configured in UNI mode: that automatically generates a Loss of IMA Frame interrupt (LIF) on UP_IRQ pin (when this interrupt is unmasked).

The following steps are performed:

1. Configure PCM Link 1 of IMA chip in UNI mode (a link can be in UNI or IMA mode).
2. Clear any pending IRQ7 interrupts by setting the corresponding bit in MPC8260 SIPNR_H register.
3. Verify that IRQ7 MPC8260 pin is deasserted by reading corresponding bit in SIPNR_H that must be set to 0.
4. Verify that PCM link 1 IRQ Link Status Register (0x222) has its LIF bit (0x04) set to 1, indicating a Loss of IMA frame.
5. Globally unmask UP_IRQ interrupts for PCM link 1 by the setting corresponding bit (0x01) in IRQ Master Enable register (0x218).
6. Unmask UP_IRQ interrupt for LIF event on PCM link 1 by setting the corresponding bit (0x04) in PCM Link 1 IRQ Link Enable register (0x219).
7. Wait 2 ms for IRQ7 assertion.

8. Verify that IRQ7 MPC8260 pin is asserted by reading corresponding bit in SIPNR_H that must be set.

IINT Command

| | |
|--------------|--------------------------------------|
| Name: | IINT |
| Syntax: | <code>iint</code> |
| Description: | Tests the IMA chip interrupt signal. |

PCI Controller Interrupt (PINT Command)

Test Description

This test verifies the connectivity between INT0 pin of the PCI controller chip (PowerSpan) and IRQ1 pin on MPC8260. For this test, Doorbell 3 (DB3) of the PowerSpan is mapped to pin INT0 and INT0 is asserted (and therefore IRQ1 of MPC8260) by writing 1 to DB3.

The following steps are performed:

1. DB3 is mapped on INT0 by writing 0x00004000 to PowerSpan IMR_DB register (0x424).
2. Clear any pending INT0 interrupt by writing a 1 to DB3 bit of PowerSpan ISR0 register (0x410).
3. Clear any pending IRQ1 interrupt by setting corresponding bit in MPC8260 SIPNR_H register.
4. Verify that DB3 bit in PowerSpan ISR0 register is 0.
5. Verify that IRQ1 MPC8260 pin is deasserted by reading corresponding bit in SIPNR_H that must be set to 0.
6. Generate a PowerSpan INT0 interrupt by writing 1 to DB3 bit of PowerSpan IER0 register (0x418).
7. Verify that DB3 bit in ISR0 is set, indicating that an interrupt is generated on PowerSpan INT0 pin.
8. Verify that IRQ1 MPC8260 pin is asserted by reading the corresponding bit in SIPNR_H, that must be set.
9. Clear INT0 interrupt by writing a 1 to DB3 bit of PowerSpan ISR0 register.
10. Verify that DB3 bit in PowerSpan ISR0 register is 0.
11. Clear IRQ1 interrupt by setting corresponding bit in MPC8260 SIPNR_H register.
12. Verify that IRQ1 MPC8260 pin is deasserted by reading corresponding bit in SIPNR_H that must be 0.

PINT Command

| | |
|--------------|---|
| Name: | PINT |
| Syntax: | <code>pint</code> |
| Description: | Tests the PowerSpan (PCI chip interface) interrupt signal to the PowerQUICC II processor. |

Loopback Tests Using The PowerQUICC II MCCs

Overview

Framers Operating Modes

The 4539 board can be programmed to operate the framers in the following modes:

Multiplexed Direct Mode

In this mode, the QuadFALC system interface is in multiplexed mode and the four framers have the same rhythm. The first QuadFALC TDM bus is directly tied to the CPM TDM bus TDMA1.

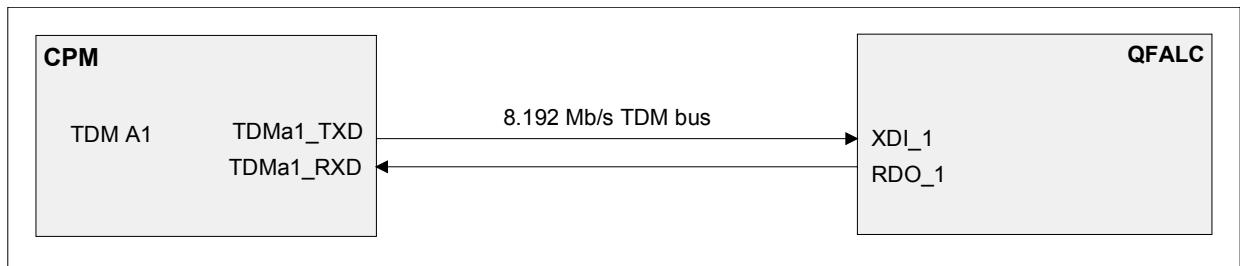


Figure 5-1. Multiplexed Direct Mode Configuration

On the quad-framers system bus, the time slots are interleaved.

This mode is activated with FIL and FEL tests by using the MUL option.

Independent Direct Mode

In this mode, each framer can have its own rhythm. Each QuadFALC TDM bus is directly tied to a CPM TDM bus and has its own clock and frame synchronization signal provided by the QuadFALC.

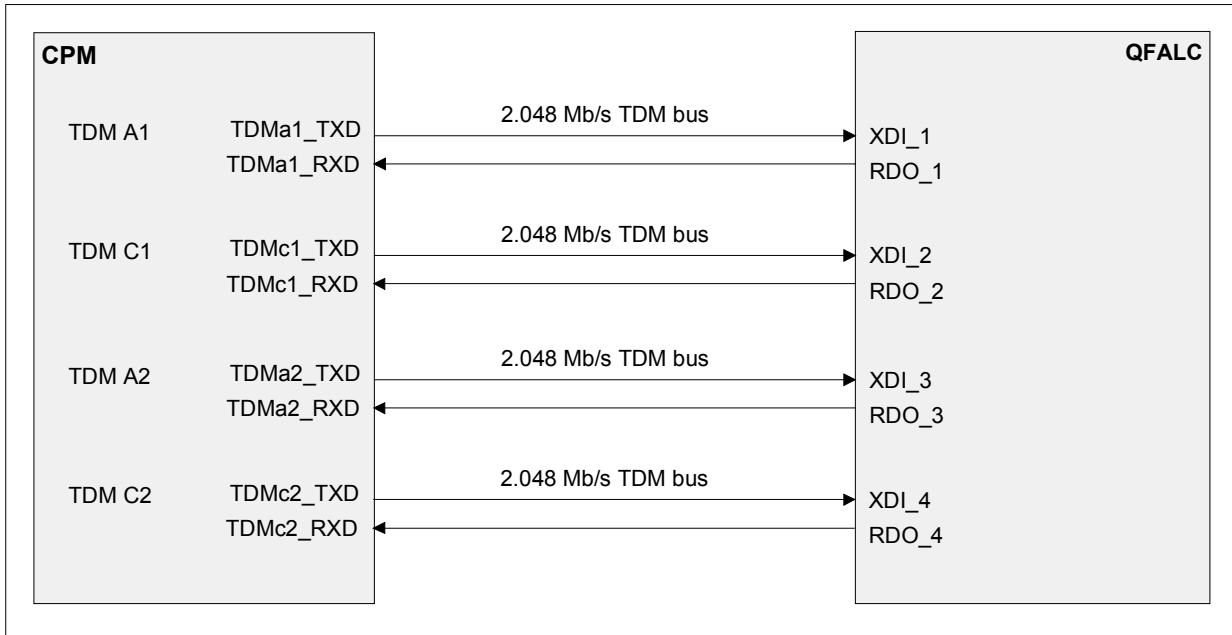


Figure 5-2. Independent Direct Mode Configuration

This mode is activated with FIL and FEL tests by using the IND option.

Switched Mode

In switched mode, the QuadFALC multiplexed TDM bus is tied to the first TDM bus on P4. The second TDM bus on P4 is tied to the MPC8260.

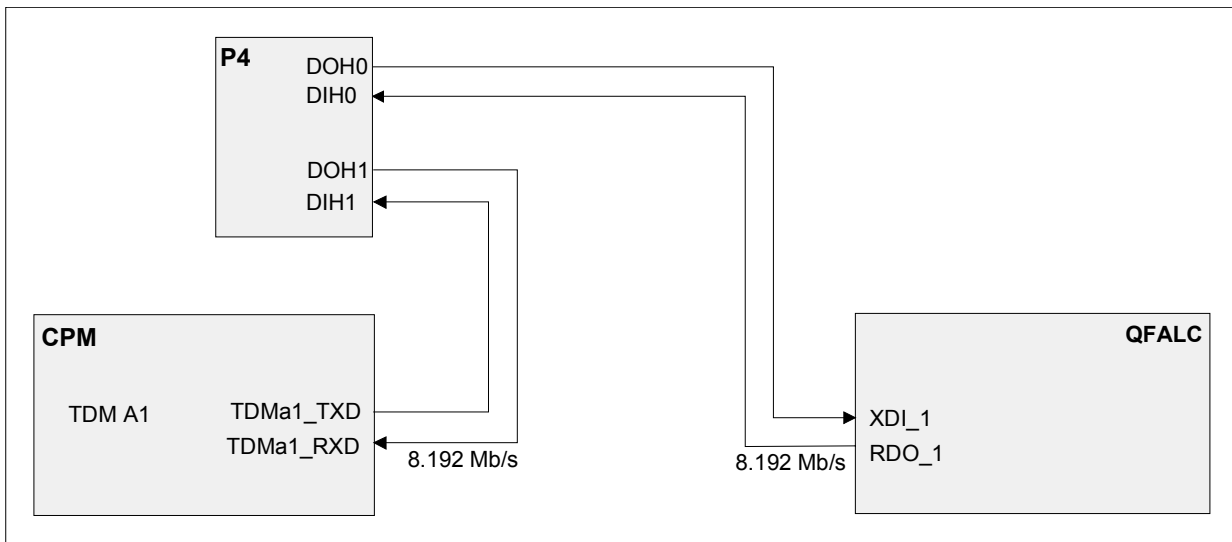


Figure 5-3. Switched Mode Configuration

The specific FSW command tests this configuration.

Pass-Through Mode

Pass-through is possible from port 0 to port 1 and vice versa and from port 2 to port 3 and vice versa. The specific FPT command tests this configuration.

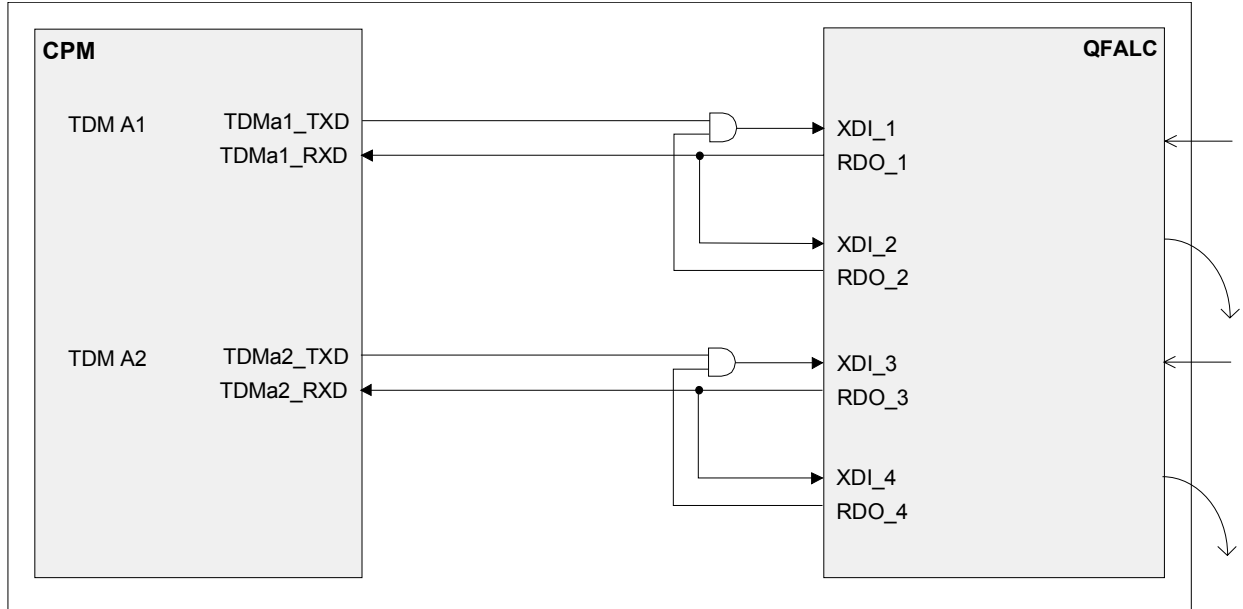


Figure 5-4. Pass-Through Mode configuration (0->1 & 2->3)

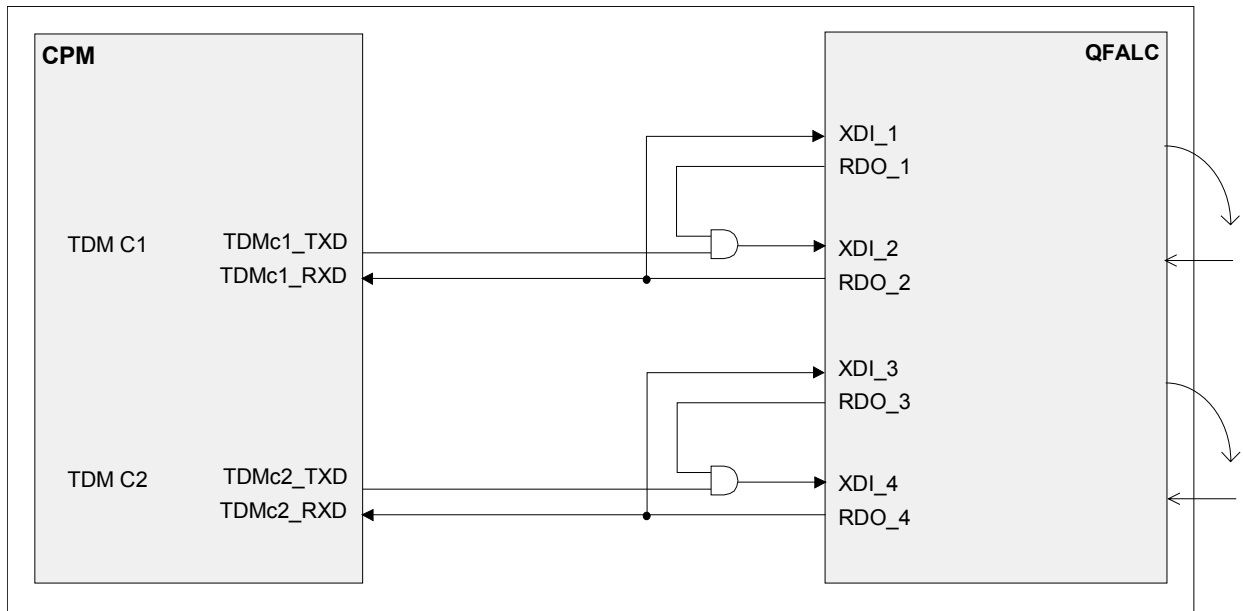


Figure 5-5. Pass-Through Mode Configuration (1->0 & 3->2)

HDLC Frames Format

The PowerQUICC-II MCCs are used to generate HDLC frames. An MCC loopback test consists of sending HDLC frames from an MCC channel and checking that they are returned. The following is a list of possible errors on an MCC channel loopback test.

Error Report:

- Abort received
- Received a non octet aligned frame
- Received a too long frame
- Received a frame with wrong CRC
- Received a frame different from expected
- Error on transmission
- Time-out

Several tests run MCC channel loopbacks:

- Framers in internal loopback (FIL)
- Framers in external loopback (FEL)
- Framers in pass-through mode (FPT)
- Framers in switched mode (FSW)
- Agency (AGENCY)

The frames are 256 bytes long with a 16-bit CRC included. [Table 5-5](#) shows the content of the frames sent on MCC channel 1.

Table 5-5. Frame Content

| Frame Byte | 0 | 1 | 2 | ... | N | ... |
|------------|--------|--------|--------|-----|--------|-----|
| 1 | 0x30 † | 0x30 † | 0x30 † | ... | 0x30 † | ... |
| 2 | 0x00 | 0x01 | 0x02 | ... | N | ... |
| 3 | 0x01 | 0x02 | 0x03 | ... | N+1 | ... |
| 4 | 0x02 | 0x03 | 0x04 | ... | N+2 | ... |
| ... | ... | ... | ... | ... | ... | ... |
| 252 | 0xFC | 0xFD | 0xFE | ... | ... | ... |
| 253 | 0xFD | 0xFE | 0xFF | ... | ... | ... |
| 254 | 0xFE | 0xFF | 0x00 | ... | ... | ... |
| 255,256 | CRC | CRC | CRC | ... | CRC | ... |

† The first byte of the frame depends on the MCC channel number.

The checking process consists of verifying that the second character of a frame received matches the one previously sent. Since CRC checking is automatically performed by the MCC controller, it is presumed that this checking is enough. A loopback test can be performed on an MCC channel with, for example, an external loopback connector on a T1/E1/J1 port, but a loopback test can also be performed between two MCC channels with an external cross-connect connector on two T1/E1/J1 ports: taking into account this possibility, no checking is performed on the first byte of the received frames.

Framers In Internal Loopback (FIL Command)

Test Description

This test verifies the connectivity between the PowerQUICC II TDMs and the framers' system bus. [Figure 5-6 on page 58](#) shows the configuration of this loopback test. In that configuration each QuadFALC TDM bus is tied directly to a CPM TDM bus and has its own clock and frame synchronization signal provided by the QuadFALC. The frames are returned by the framers configured in internal loopback. A loopback test is performed on each framers' port (from P0 to P3).

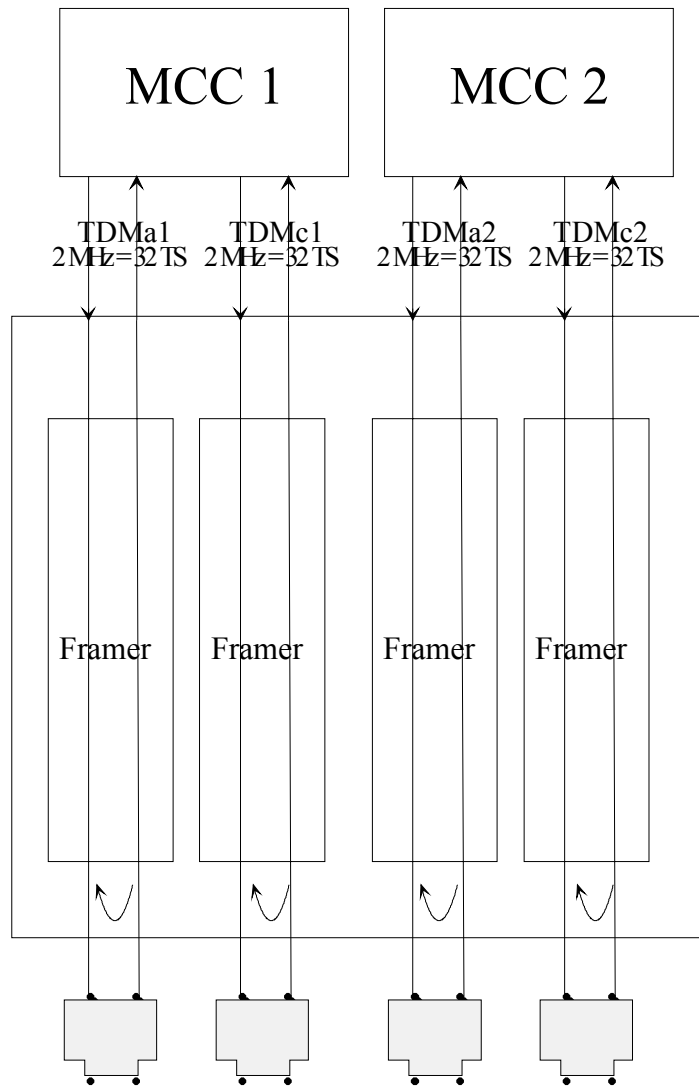


Figure 5-6. Framers Internal Loopback Test Configuration

FIL Command

Name: FIL
Syntax: fil [bus] [ports] [nx]

| | |
|--------------|---|
| Description: | This command tests the framers ports in internal loopback configuration. The test verifies that the frames sent from the PQII MCCs and looped back within the framers are correctly returned. In multiplexed mode (option bus=MUL), frames on ports 0 through 3 are sent from MCC1/TDMa1 TDM; in independent mode, (option bus=IND), frames on ports 0, 1, 2, 3 are sent form TDMa1, TDMc1, TDMa2, TDMc2 respectively. Frame length is 256 characters (2-byte CRC included) and one time slot (64 kbps) is used for each port. The framing format is E1 and the clock mode is MASTER. |
| Options: | <p><code>bus</code> Defines the direct mode type: { IND MUL }; IND is for independent mode. MUL is for multiplexed mode. Default is IND.</p> <p><code>ports</code> Defines the ports on which the test is performed. Syntax is Pxyz..., where s, y, z ... are port numbers. Default is all ports.</p> <p><code>nx</code> Used to specify the number of 32-bit frames to send on each port. Default is 1. Use x=0 to run the test indefinitely.</p> |
| Examples: | <p><code>fil mul p10 n0</code> Indefinitely exchange frames on ports 1 and 0 in multiplexed direct mode.</p> <p><code>fil ind n50</code> Exchanges 50 frames on each port in independent direct mode.</p> |

Framers In External Loopback (FEL Command)

Test Description

[Figure 5-7 on page 60](#) shows the configuration of this loopback test. In that configuration, each QuadFALC TDM bus is tied directly to a CPM TDM bus and has its own clock and frame synchronization signal provided by the QuadFALC. The frames are returned by external loopback connectors. A loopback test is performed on each port.

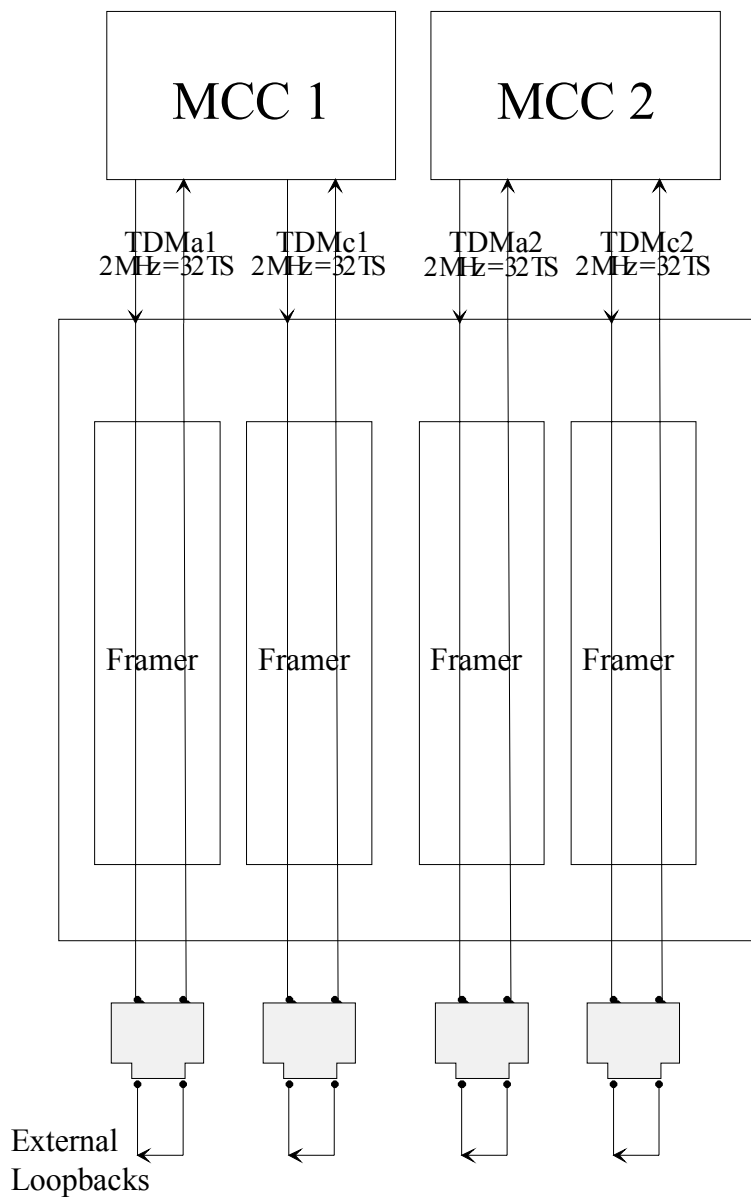


Figure 5-7. Framers External Loopback Test Configuration

FEL Command

Name: FEL

Syntax: fel [format] [mode] [bus] [ports] [nx]

| | |
|--------------|--|
| Description: | This command tests the T1/E1/J1 ports in external loopback configuration. If the board is plugged into a CompactPCI carrier card, the RTM is necessary with loopback connectors plugged on the T1/E1/J1 ports – if the board is plugged into a PCI carrier card, the special P4LBTstConn loopback connector on P4 can be used, but in that case, the RTM itself is not tested (see P3LBTstConn and P4LBTstConn Loopback Connectors on page 150 for a description of the P4LBTstConn connector). The test verifies that the frames sent from the PQII MCCs through T1/E1/J1 framers are correctly returned. In multiplexed mode (option bus=MUL), frames on ports 0 through 3 are sent from MCC1/TDMa1 TDM; in independent mode (option bus=IND), frames on ports 0, 1, 2, 3 are sent from TDMa1, TDMc1, TDMa2, TDMc2 respectively. The length of the frames is 256 characters (2-byte CRC included) and one time slot (64 kbps) is used for each port. |
| Options: | <p><code>format</code> Defines the framing format: {T1 E1 E1-CRC4 J1}; default value is E1.</p> <p><code>mode</code> Defines the clock configuration: {MASTER SLAVE}; default value is MASTER.</p> <p><code>bus</code> Defines the direct mode type : {IND MUL}; IND is for independent mode. MUL is for multiplexed mode. Default is IND.</p> <p><code>ports</code> Defines the ports on which the test is performed. Syntax is Pxyz..., where s, y, z ... are port numbers. Default is all ports.</p> <p><code>nx</code> Used to specify the number of 32-bit frames to send on each port. Default is 1. Use x=0 to run the test indefinitely.</p> |
| Example: | <p>fel e1 slave mul p10 n100 Exchanges 100 frames on ports 1 and 0 in multiplexed direct mode using E1 framing, slave clocks. Incoming signal on port 1 provides the clocks (bit and frame clocks) for TDMa1.</p> <p>fel t1 master ind p23 n0 Indefinitely exchanges frames on ports 2 and 3 in independent direct mode using T1 framing, master clocks.</p> |

Framers In Pass-Through Mode (FPT Command)

Test Description

This test verifies the board connectivity for the pass-through mode. Pass-through mode from port 0 to port 1 is verified by configuring the two ports in internal loopback: MCC1 channel 0 is mapped on time slot 1 (time-slot numbering starts from 0) of TDMA1 TX SIRAM bank and on time slot 2 of TDMA1 in RX SIRAM bank; framer of port 1 is configured to shift time slot 'i' to time slot 'i+1' on its TDM output (RDO2) and for other time slots than time slot 2, 0xFF is output; that configuration has the following implication: a frame sent from TDMA1_TXD on time slot 1 is received on both time slots 1 and 2 of TDMA1_RXD; with MCC1 channel 0 mapping in SIRAM as defined earlier, a frame sent on that channel should be received on the same channel. The same principles are used for the other pass-through configurations.

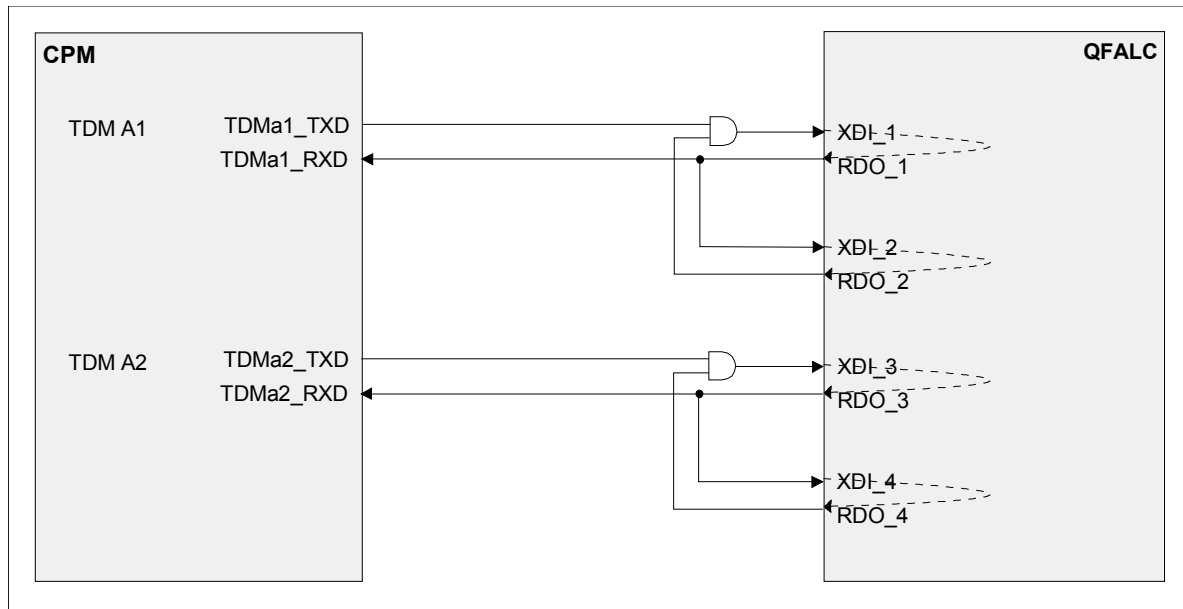


Figure 5-8. Configuration for Pass-Through Port 0 to Port 1 and Port 2 to Port 3 Test

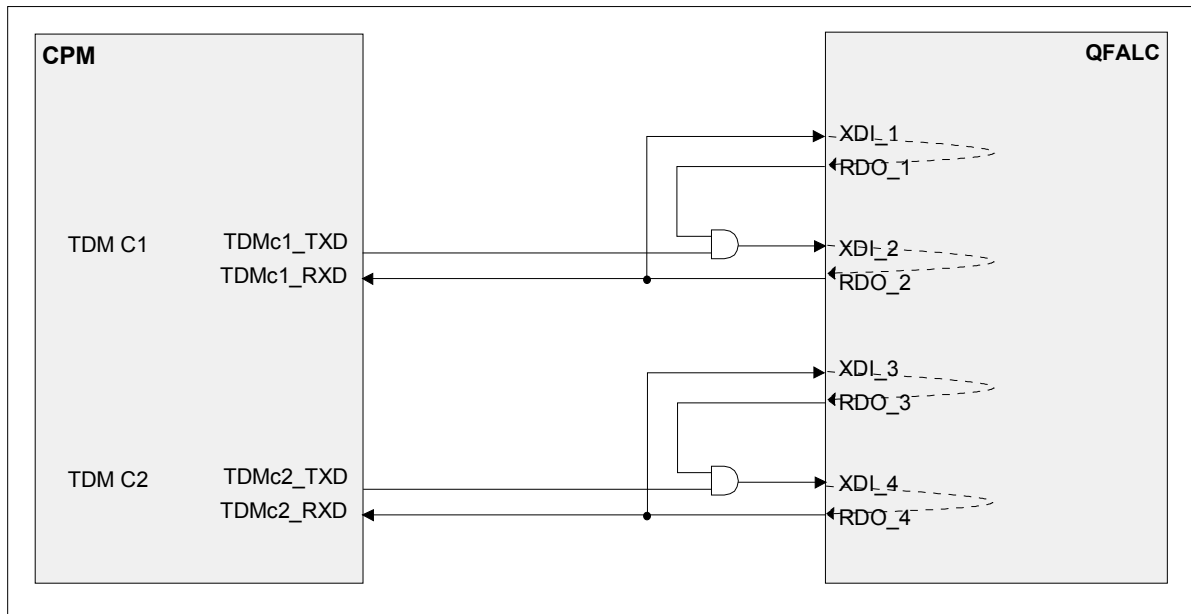


Figure 5-9. Configuration for Pass-Through Port 1 to Port 0 and Port 3 to Port 2 Test

FPT Command

| | |
|--------------|--|
| Name: | FPT |
| Syntax: | <code>fpt [ports] [nx]</code> |
| Description: | This command tests the T1/E1/J1 framers in internal loopback with pass-through mode configuration. The test verifies that the frames sent from the PQII MCCs and looped back within the framers are correctly returned. |
| Options: | <p><code>ports</code> Defines the pass-through mode: { P0123 P0132 P1023 P1032 } P0123 to test port 0 to 1 and 2 to 3 P0132 to test port 0 to 1 and 3 to 2 P1023 to test port 1 to 0 and 2 to 3 P1032 to test port 1 to 0 and 3 to 2 Default is P0123</p> <p><code>nx</code> Used to specify the number of 32-bit frames to send on each port. Default is 1. Use x=0 to run the test indefinitely.</p> |
| Example: | <p><code>fpt p0132 n100</code> Test port 0 to 1 and 3 to 2 pass-through mode configurations. 100 frames are sent on framers 0 and 3 and returned by framers 1 and 2 respectively.</p> |

Framers In Switched Mode (FSW Command)

Test Description

This test should be run with the 4539 board plugged in a PCI carrier board with the special P4LBTstConn connector (see [P3LBTstConn and P4LBTstConn Loopback Connectors on page 150](#)). The following figure shows how the QuadFALC and CPM TDMA1 data and clock signals are tied together using the P4LBTstConn connector. On TDMA1, loopback tests are performed in the same way as for the FIL (Framers Internal Loopback) test in multiplexed direct mode (MUL option).

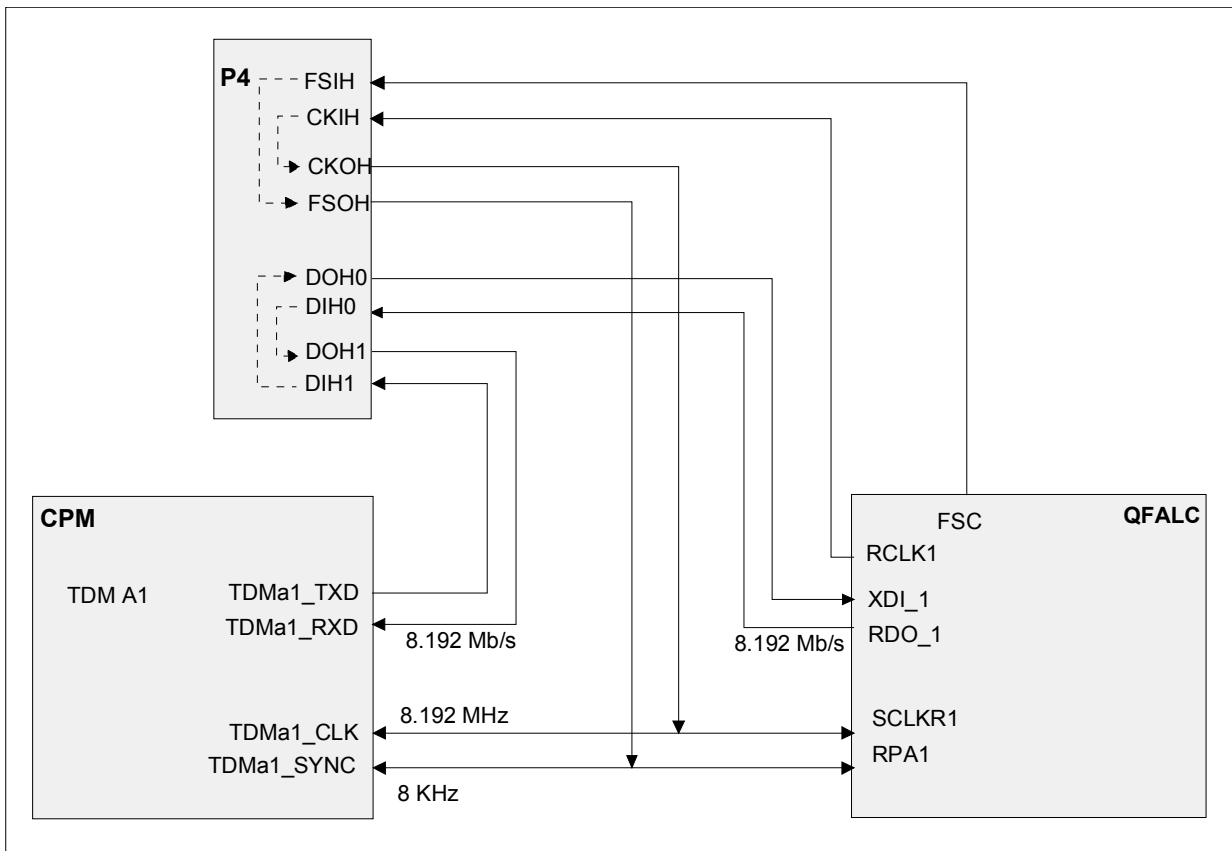


Figure 5-10. Framers in Switched Mode Test Configuration

FSW Command

Name: FSW

Syntax: `fsw [ports] [nx]`

Description: This command tests the T1/E1/J1 framers in internal loopback with switched mode configuration. The test verifies that the frames sent from PQII TDMA1 are correctly returned by the framers. Clocks and Data stream connections between TDMA1 and QuadFALC are

performed using the special P4LBTstConn connector on P4. This special connector is described in [P3LBTstConn and P4LBTstConn Loopback Connectors](#) on page 150.

| | |
|----------|--|
| Options: | <p><code>ports</code> Defines the ports on which the test is performed. Syntax is Pxyz..., where x, y z are port numbers in range [0,3]. Default is all ports: P0123.</p> <p><code>nx</code> Used to specify the number of 32-bit frames to send on each port. Default is 1. Use x=0 to run the test indefinitely.</p> |
| Example: | <p><code>fsw p01 n0</code> Indefinitely exchange frames on ports 0 and 1.</p> <p><code>fsw n50</code> Exchange 50 frames on each port.</p> |

Loopback Tests For Ethernet

Overview

Ethernet loopback tests consist of sending frames from the MPC8260 FCC3 controller and verifying that they are returned. [Figure 5-11](#) shows the Ethernet frame structure.

| Preamble | Start Frame Delimiter | Destination Address | Source Address | Type/Length | Data | Frame Check Sequence |
|----------|-----------------------|---------------------|----------------|-------------|-----------------|----------------------|
| 7 Bytes | 1 Byte | 6 Bytes | 6 Bytes | 2 Bytes | 46 - 1500 Bytes | 4 Bytes |

Figure 5-11. Ethernet Frame Structure

The frames sent from the FCC3 controller have the following content:

- Destination and Source Addresses = 55 48 33 22 19 00 (hexadecimal).
- Length = 512 bytes.
- Data = All the bytes have the same 0xA5 content. When several frames are sent, the content is incremented from one frame to the other. Once 0xFF is reached, the content is put to 0x00.

Ethernet In Internal Loopback (EIL Command)

Test Description

This test configures the Ethernet interface chip in 10 Mbps internal loopback. Frames are sent from the PowerQUICC II FCC to the Ethernet chip through the MII bus; the test checks that the frames are returned without errors.

Figure 5-12 shows that loopback is performed between the MII and the digital block of LXT971.

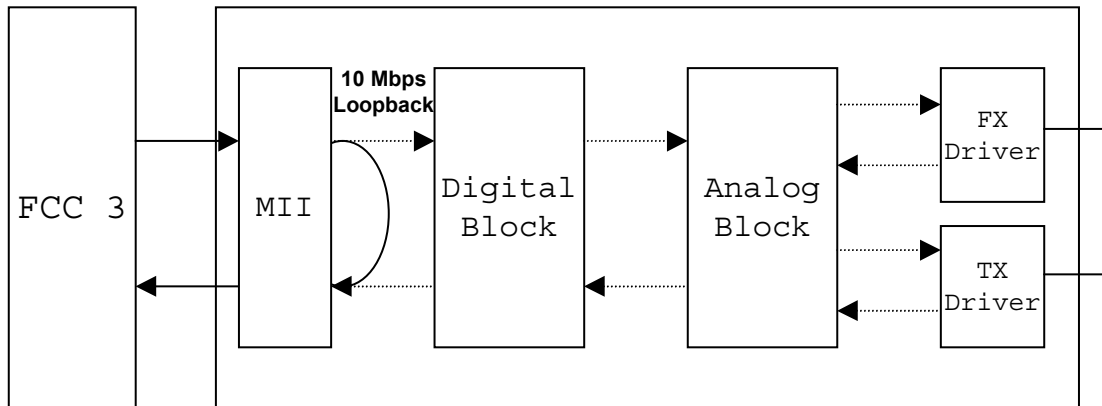


Figure 5-12. Ethernet Internal Loopback Between MII and Digital Block

To configure the Ethernet interface chip in internal loopback, the following LXT971 register operations are performed:

1. Loopback mode is enabled by setting bit 14 of the Control register (0x00).
2. Full duplex mode is enabled by setting bit 8 of the Control register.
3. Auto-negotiation is disabled by resetting bit 12 of the Control register.
4. 10 Mbps speed is selected by resetting bits 6 and 13 of the Control register.
5. Operational loopback is disabled by resetting bit 8 of the LXT971 Configuration register (0x10) .

EIL Command

| | |
|--------------|---|
| Name: | EIL |
| Syntax: | eil [nx] |
| Description: | Tests the Ethernet port in internal loopback configuration. The Ethernet interface chip is configured to echo the frames sent from the PQII FCC3. This test verifies that the frames sent are returned correctly. The test stops when all the frames are correctly returned or when an error occurs (time out or an incorrect frame is returned). |

Options: nx
 Used to specify the number of 32-bit frames to send on each port.
 Default is 1. Use x=0 to run the test indefinitely.

Examples: **ei1 n100** (Runs the test for 100 frames)
 ei1 n0 (Runs the test indefinitely)

Ethernet In External Loopback (EEL Command)

Test Description

This test is similar to the internal loopback test, except that the loopback is performed externally by a loopback connector (see [Ethernet Loopback Connector on page 149](#)). The Ethernet interface chip is configured in 10 Mbps full duplex mode.

EEL Command

| | | | | | | | | | | | |
|------------------------|--|-----------------------|-------------------|-----------------------|-------------------|------------------------|--------------------|------------------------|--------------------|---------------------|------------------|
| Name: | EEL | | | | | | | | | | |
| Syntax: | <code>eel [nx] [mode]</code> | | | | | | | | | | |
| Description: | <p>This command tests the Ethernet port in external loopback configuration. A loopback connector must be plugged into the Ethernet interface for 10 or 100 BT full duplex tests. To test 10 or 100 BT half duplex, a switch is required. The test verifies that frames sent from PQII FCC3 are correctly returned. The test stops when all the frames are correctly returned or when an error occurs (time out or incorrect frame is returned).</p> | | | | | | | | | | |
| Option: | <p><code>nx</code> Used to specify the number of 32-bit frames to send. Default is 1. Use <code>x=0</code> to run the test indefinitely.</p> <p><code>mode</code> Specifies the connection mode (default is <code>ETH_10FD</code>).</p> <table> <tr> <td><code>ETH_10HD</code></td> <td>10 BT half duplex</td> </tr> <tr> <td><code>ETH_10FD</code></td> <td>10 BT full duplex</td> </tr> <tr> <td><code>ETH_100HD</code></td> <td>100 BT half duplex</td> </tr> <tr> <td><code>ETH_100FD</code></td> <td>100 BT full duplex</td> </tr> <tr> <td><code>ETH_AN</code></td> <td>Auto-negotiation</td> </tr> </table> | <code>ETH_10HD</code> | 10 BT half duplex | <code>ETH_10FD</code> | 10 BT full duplex | <code>ETH_100HD</code> | 100 BT half duplex | <code>ETH_100FD</code> | 100 BT full duplex | <code>ETH_AN</code> | Auto-negotiation |
| <code>ETH_10HD</code> | 10 BT half duplex | | | | | | | | | | |
| <code>ETH_10FD</code> | 10 BT full duplex | | | | | | | | | | |
| <code>ETH_100HD</code> | 100 BT half duplex | | | | | | | | | | |
| <code>ETH_100FD</code> | 100 BT full duplex | | | | | | | | | | |
| <code>ETH_AN</code> | Auto-negotiation | | | | | | | | | | |
| Examples: | <p><code>ee1 n100</code> (Runs the test for 100 frames)</p> <p><code>ee1 n0</code> (Runs the test indefinitely)</p> | | | | | | | | | | |

IMA Internal Loopback (IIL Command)

Test Description

[Figure 5-13 on page 70](#) shows how the IMA chip is interfaced with the PowerQUICC II FCC 2 controller and the QuadFALC. The IMA is connected to FCC2 through a UTOPIA bus, and to the QuadFALC through four TDMs. To test the IMA chip, the framers are internally loop-backed. On the UTOPIA side, the MPHY feature is used to associate each IMA TDM to one UTOPIA PHY address. On FCC2, only one loopback test can be run at a time (see [IMA Internal Loopback \(IIL Command\) on page 68](#)). Cells are sent from FCC2 and the test checks that they are returned with no error.



NOTE

Boards equipped with a MITEL MT90221 have only four TDMs, whereas boards with a MITEL MT90220 have four additional TDMs: the IIL command does not test the additional TDMs (4 through 7).

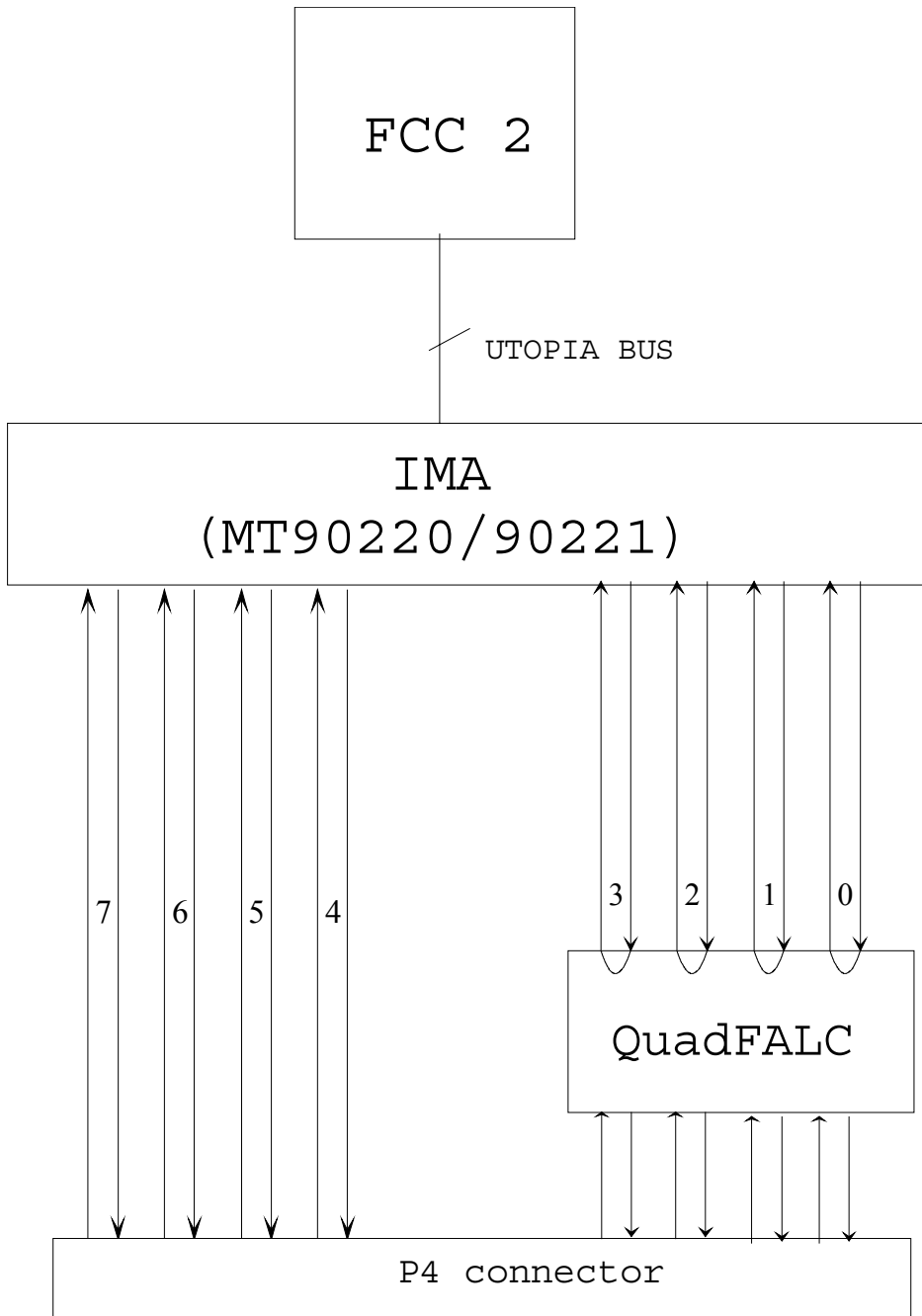


Figure 5-13. IMA internal Loopback Test Configuration

IIL Command

Name: IIL
Syntax: iil port [Nx]

| | |
|--------------|---|
| Description: | This command tests IMA with QuadFALC framers in internal loopback. AAL5 frames are sent from PQII/FCC2 through the IMA chip to the QuadFALC and the test verifies that they are correctly returned. |
| | <p>port</p> <p>Defines the IMA TDM on which the test is performed. Syntax is Px, where x is an IMA TDM port number in range [0,3].</p> |
| Options: | <p>nx</p> <p>Used to specify the number of 32-bit frames to send. Default is 1. Use x=0 to run the test indefinitely</p> |
| Examples: | <p><code>i11 p0 n100</code> (Runs the test for 100 frames on IMA TDM0 - loopback is performed within T1/E1/J1 port 0 framer)</p> <p><code>i11 p3 n0</code> (Runs the test indefinitely on IMA TDM3 - loopback is performed within T1/E1/J1 port 3 framer)</p> |

IMA External Loopback (IEL Command)

Test Description

[Figure 5-14 on page 72](#) shows how the IMA chip is interfaced with the PowerQUICC II FCC 2 controller and the QuadFALC. The IMA is connected to FCC2 through a UTOPIA bus, and to the QuadFALC through four TDMS. To test the IMA chip, the framers are externally looped-back. Cells are sent from FCC2 and the test checks that they are returned with no error. Note that boards equipped with a MITEL MT90221 have only four TDMS whereas boards with a MITEL MT90220 have four additional TDMS: the IEL command does not test the additional TDMS (4 through 7).

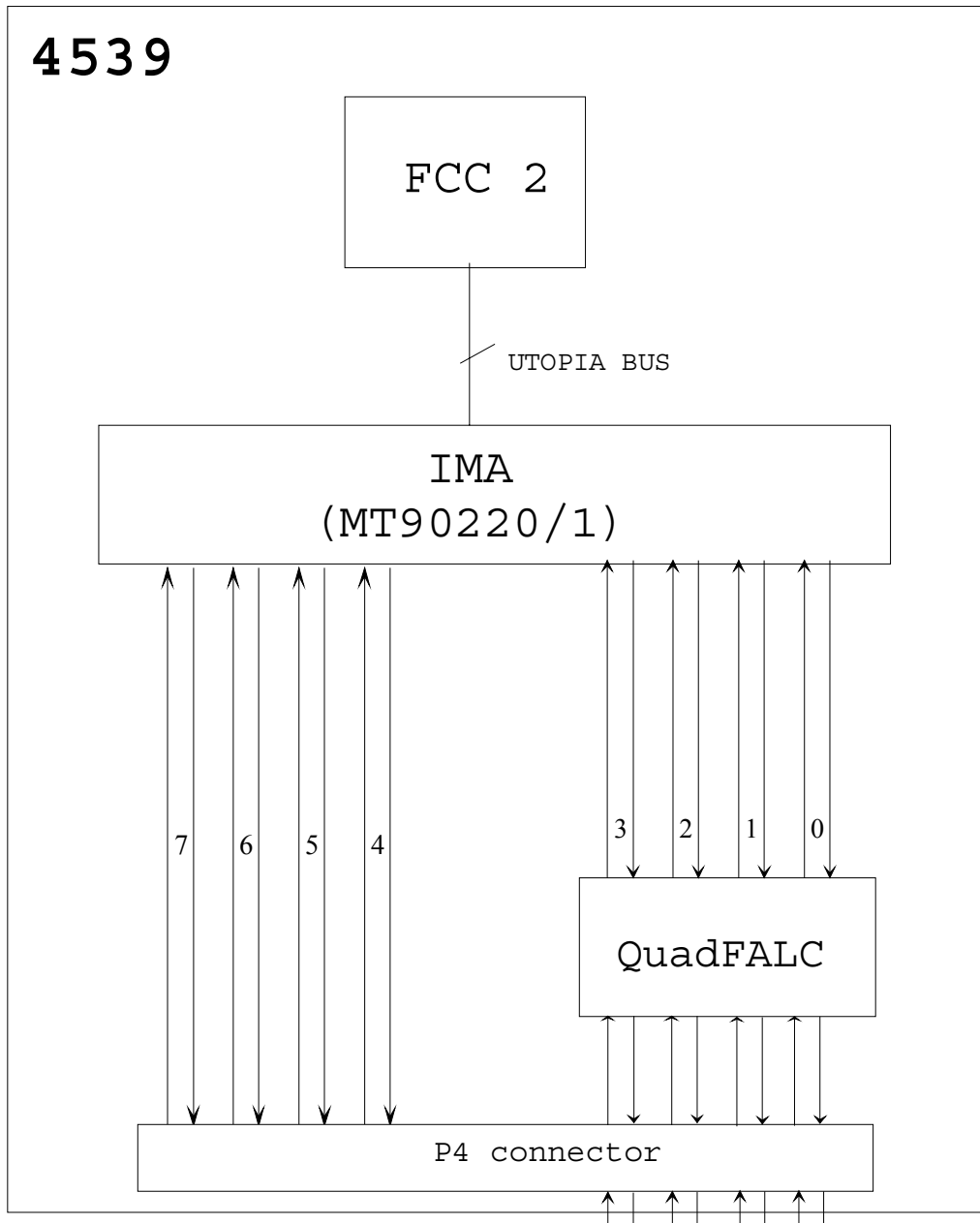


Figure 5-14. IMA External Loopback Test Configuration

IEL Command

Name: IEL
Syntax: iel [format] [mode] port [Nx]

| | |
|--------------|---|
| Description: | This command tests IMA with QuadFALC framers externally looped-back. AAL5 frames are sent from PQII/FCC2 through the IMA chip to the QuadFALC and the test verifies that they are correctly returned. |
| | <p>port Defines the IMA TDM on which the test is performed. Syntax is Px, where x is an IMA TDM port number in range [0,3].</p> |
| Options: | <p>format Defines the framing format: {T1 E1 E1-CRC4 J1}; default value is E1</p> <p>mode Defines the clock configuration: {MASTER SLAVE}; default is MASTER.</p> <p>nx Used to specify the number of 32-bit frames to send. Default is 1. Use x=0 to run the test indefinitely</p> |
| Examples: | <p><code>ie1 e1 slave p0 n100</code> (Runs the test for 100 frames on port 0 using E1 framing and slave clock)</p> <p><code>ie1 t1 master p3 n0</code> (Runs the test indefinitely on port 3 using T1 framing and master clock)</p> |

BERT (FB, FRL Commands)

Test Description

This test is used to qualify a link between a 4539 T1/E1/J1 port and a remote device.

The 4539 framers have the ability to generate and monitor a $2^{20}-1$ Pseudo-Random Bit Sequence (PRBS). The PRBS generator handles an unframed data stream. The FB command generates the PRBS and monitors the returned pattern. If the remote device is a 4539, the port connected to the link must be configured in remote loopback by using the FRL command.

[Figure 5-15](#) shows a scenario between two 4539 boards, one generating the PRBS on port 2 and the other looping back the signal.

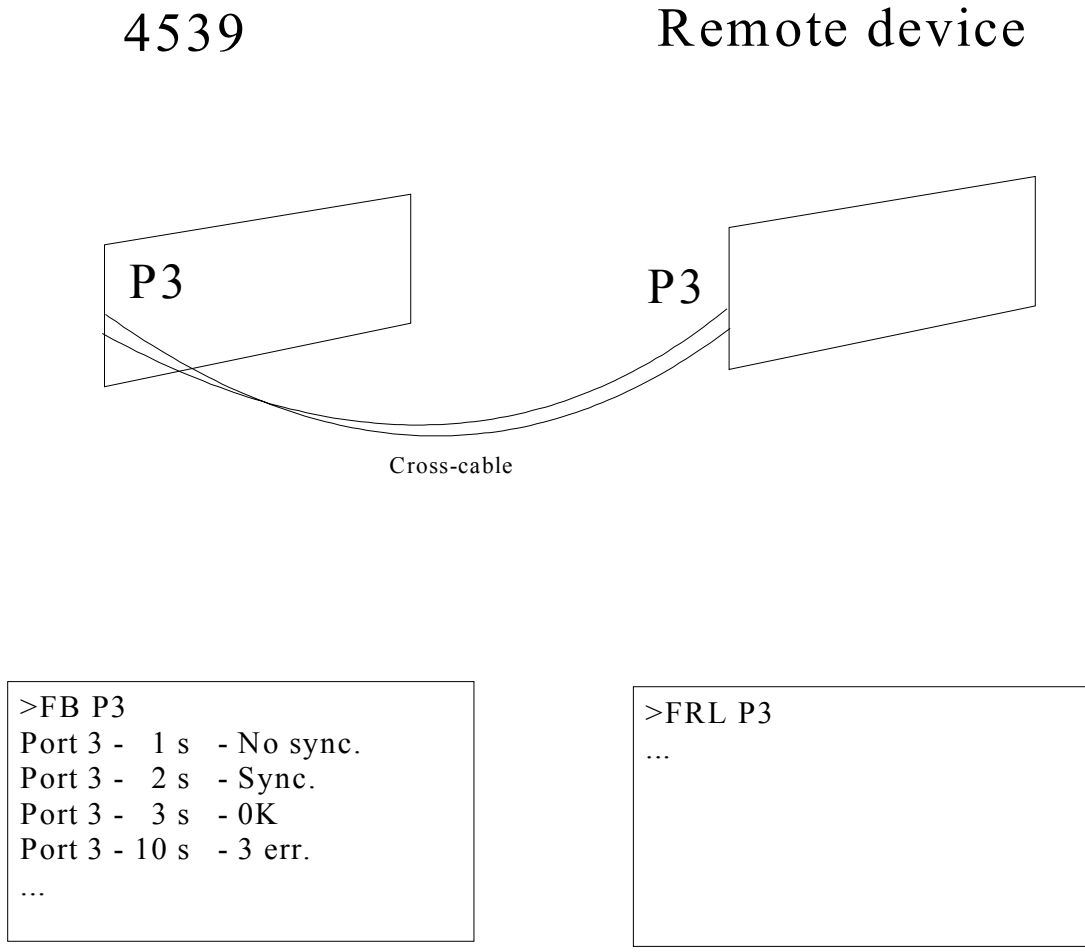


Figure 5-15. BERT Scenario

The board that generates the PRBS, polls the status of the PRBS monitor every second. Synchronization is reached within 400 ms with a probability of 99.9% and a bit error rate of up to 10^{-1} . The PRBS monitor bit error counter is polled and reset every second: only the changes are displayed.

On [Figure 5-15](#), a BERT test is performed on port 3 (P3), with a second 4539 as the remote device on which a ‘FRL P4’ (Framers Remote Loopback) test is run. The BERT result is polled every 1 second and a message is displayed when a change occurs. Note that initially, 1 second after BERT starts, BERT monitor on Port 3 is not synchronized *Port 3 - 1 s - No sync.*, then one second later, the BERT monitor synchronizes *Port 3 - 2 s - Sync.* and finally another second later, the BERT monitor detects full correct pseudo-random sequences *Port 3 - 3 s - OK.* Within seconds 4 to 9, the BERT monitor still recognizes pseudo-random bit sequences with no errors: no special information is displayed since it is still *OK*, but on

second 10 (between second 9 and 10), the monitor detects 3 false bits (but the monitor is still synchronized and still recognizes pseudo-random bit sequences) and a message informs the user: *P3 - 10s - 3err.*

FB Command

| | |
|--------------|---|
| Name: | FB |
| Syntax: | <code>fb [format] [mode] [bus] [ports]</code> |
| Description: | <p>Generates Pseudo-Random Bit Sequences (PRBS) on T1/E1/J1 ports and monitors signals that are supposed to be returned by an external device. The return signal status is polled every second and displayed upon changes. The format of the report can be one of the following:</p> <pre> Port X - Ys - no sync. Port X - Ys - sync Port X - Ys - zzz err. Port X - Ys - OK </pre> <p>x is the port number Y is the number of elapsed seconds zzz is the number of bit errors for the preceding second OK indicates that the returned PRBS is correct</p> |
| Options: | <p><code>format</code> Defines the framing format: {T1 E1 E1-CRC4 J1}; Default is E1</p> <p><code>mode</code> Defines the clock configuration: MASTER or SLAVE Default is MASTER</p> <p><code>bus</code> Defines the direct mode type: {IND MUL} IND for independent mode. MUL for multiplexed mode. Default is IND.</p> <p><code>ports</code> Defines the ports on which the test is done. Syntax is Pxyz..., where x, y, z ... are port numbers. Default is all ports.</p> |
| Example: | <p><code>fb e1 slave ind p02</code> (performs BERT on ports 0 and 2 in independent direct mode, using E1 framing and slave clocks)</p> <pre> 1. FB E1 SLAVE IND P02: 2. Port 0 - 1s - no sync 3. Port 2 - 1s - no sync 4. Port 2 - 4s - sync. 5. Port 2 - 5s - OK 6. Port 0 - 6s - sync. 7. Port 0 - 7s - 3 err. 8. Port 0 - 8s - 5 err. 9. ... </pre> |

FRL Command

- Name:** FRL
- Syntax:** `frl [format] [bus] [ports]`
- Description:** Configures the framers in remote loopback: the input signal is returned through a transmit jitter attenuator – the clock used to transmit is the one recovered from the input signal: a port in remote loopback behaves like a Slave.
- It can be useful when doing tests on framers such as external loopback (see FEL command), or physical alarm polling (see FPHY command), to use another 4539 board in place of external loopback connectors and configured in remote loopback. This commands runs indefinitely until it is stopped with the `s` command.
- Options:**
- `format` Defines the framing format: T1, E1, E1-CRC4 OR J1
Default is E1.
 - `bus` Defines the direct mode type: {IND|MUL}
IND for independent mode.
MUL for multiplexed mode.
Default is IND.
 - `ports` Defines the ports on which the test is done. Syntax is Pxyz..., where x, y, z ... are port numbers.
Default is all ports.
- Example:** `frl t1` (configures all framers in remote loopback with T1 framing)

Framer Line Status (FPHY Command)

Test Description

This test allows monitoring the alarms of a T1/E1/J1 link. [Table 5-6](#) shows the possible alarms.

Table 5-6. Framer Alarms

| Alarm | Abbrev. | Other Designation | Comment |
|-------------------------------|---------|-------------------|-------------------------------|
| Loss of signal | LOS | Red alarm | |
| Alarm indication signal | AIS | Blue alarm | |
| Loss of frame alignment | LFA | LOF | |
| Receive Remote alarm | RRA | RAI, Yellow alarm | |
| No Multiframe Alignment Found | NMF | | Valid in E1-CRC4 format only. |

The test can be run in either E1 (Double frame format), E1-CRC4 (Multi-frame format), T1, or J1 framing mode.

FPHY Command

Name: FPHY

Syntax: `fphy [format] [mode] [bus] [ports]`

Description: Configures the framers according to the parameters of the command and performs a continuous pooling of their alarm status and displays them when one changes. The T1/E1/J1 ports are assumed to be connected to external devices.

The following alarms are monitored:

LOS: Loss of Signal
 AIS: Alarm Indication Signal
 LFA: Loss of Frame Alignment
 RRA: Receive Remote Alarm
 NMF: No Multi-frame alignment Found

A “-” character before the alarm mnemonic indicates that the alarm is not active. For example: -LOS

A “+” character before the alarm mnemonic indicates that the alarm is active. For example: +LOS

Note: The NMF alarm is valid in E1-CRC4 format only and always displays -NMF in other formats.

Options:

- `format` Defines the framing format: {T1 | E1 | E1-CRC4 | J1};
Default is E1.
- `mode` Defines the clock configuration: MASTER or SLAVE
Default is MASTER.
- `bus` Defines the direct mode type: {IND | MUL}
IND for independent mode.
MUL for multiplexed mode.
Default is IND.
- `ports` Defines the ports on which the test is done. Syntax is Pxyz..., where x, y, z ... are port numbers.
Default is all ports.

Example: `fphy p21`

```

Port 2: -LOS -AIS -LFA -RRA -NMF
Port 1: -LOS +AIS -LFA -RRA -NMF
Port 1: -LOS -AIS -LFA -RRA -NMF
Port 2: -LOS -AIS +LFA -RRA -NMF
...
...

```

IMA TDM Clocks (ICLK Command)

Test Description

The IMA chip qualifies the clock inputs on its TDMs.

There are two types of clocks:

- The bit clock at 2 MHz
- The frame clock at 8 KHz

The test verifies that the frame clocks are at 8 KHz and that the bit clocks toggle (no check on bit clock rate is performed). The test is only run on the TDMs connected to the framers.

ICLK Command

| | |
|--------------|--|
| Name: | ICLK |
| Syntax: | <code>iclk</code> |
| Description: | This command tests the clocks of the IMA 4 TDMs connected to the framers. Two types of clocks are tested: 2 MHz bit clock and 8 KHz frame clock. |

IMA Optional TDMs Clocks (ICLK8 Command)

Test Description

This test is available on boards equipped with the optional IMA MT90220 chip: this chip provides four additional PCM links directly connected to the P4 connector. It is similar to ICLK test but is run on IMA TDMs in range [4,7] instead of [0,3].

To run this test, the 4539 must be plugged on a PCI carrier card. The 4539 must have a special loopback connector on P4 (see [P3LBTstConn and P4LBTstConn Loopback Connectors on page 150](#)). Note that the purpose of this test is to check the connectivity of the IMA additional TDMs clocks traces: its purpose is not to test the IMA additional TDM clocks provided by an external system. [Figure 5-16 on page 79](#) shows how the clocks are generated from the 4539 QuadFALC frame (FSC) and bit (RCLK1) clock outputs.

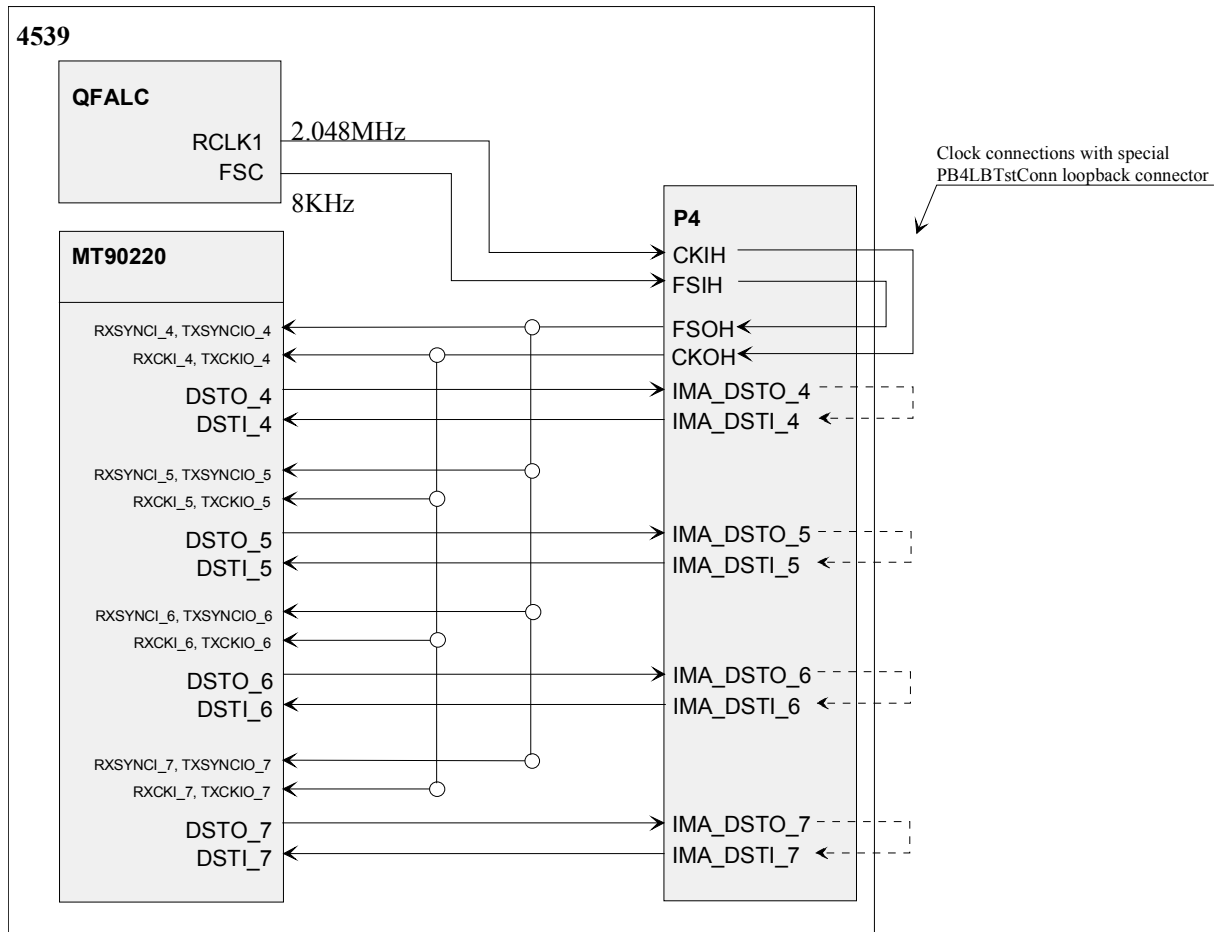


Figure 5-16. ICLK8 Test – Clock Configuration with Loopback Connector on P4

ICLK8 Command

Name: ICLK8

Syntax: `iclk`

Description: This command tests the clock connectivity of the optional IMA 4 TDMs available with the MITEL MT90220. These TDMs are directly connected to P4. To run the test, the board must be plugged in a PCI carrier card with a special loopback connector on P4 as shown in [P3LBTstConn and P4LBTstConn Loopback Connectors on page 150](#) (this loopback connector connects FSOH to FSIH, CKOH to CKIH, and DSTIx to DSTOx with x in range [4,7]).

IMA Optional TDMs External Loopback (IEL8 Command)

Test Description

This test is available on boards equipped with the optional IMA MT90220 chip: this chip provides four additional PCM links directly connected to the P4 connector. It is similar to IEL test but is run on IMA TDMs in range [4,7] instead of [0,3].

To run this test, the 4539 must be plugged in a PCI carrier card. The 4539 must have a special loopback connector on P4 (see [P3LBTstConn and P4LBTstConn Loopback Connectors on page 150](#)). Note that in comparison with ICLK8 that tests only the clock inputs (for TDMs 4 through 7), the purpose of the IEL8 test is to check the global connectivity of the TDMs.

Figure 5-17 shows how the clocks are generated from the 4539 QuadFALC frame (FSC) and bit (RCLK1) clock outputs. To test the IMA chip TDM 4 through 7, these are externally looped back. On FCC2, only one loopback test can be run at a time. Cells are sent from FCC2 and the test checks that they are returned with no errors.

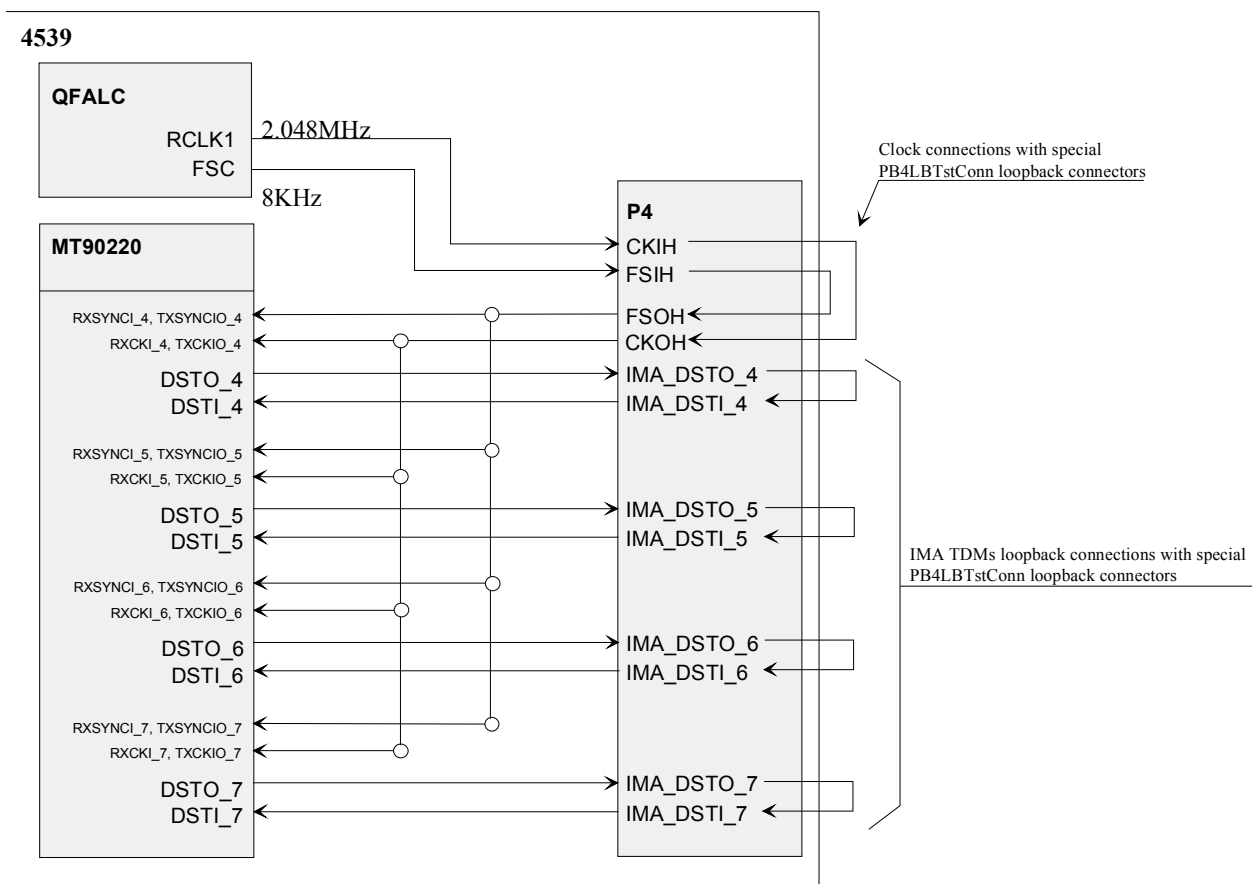


Figure 5-17. IEL8 Test – IMA TDMs 4 to 7 Configuration with Loopback Connectors on P4

IEL8 Command

| | |
|--------------|--|
| Name: | IEL8 |
| Syntax: | <code>iel8 port [Nx]</code> |
| Description: | <p>This command tests the global connectivity of the optional IMA 4 TDMs available with the MITEL MT90220. These TDMs are directly connected to P4. To run the test, the board must be plugged in a PCI carrier card with a special loopback connector on P4 as shown in P3LBTstConn and P4LBTstConn Loopback Connectors on page 150 (this loopback connector connects FSOH to FSIH, CKOH to CKIH, DSTIx to DSTOx with x in range [4,7]). To test a TDM, AAL5 frames are sent from PQII/FCC2 through IMA chip on that TDM and the test verifies that they are correctly returned.</p> <p><code>port</code> Defines the IMA TDM on which the test is performed. Syntax is Px, where x is an IMA TDM port number in range [4,7].</p> |
| Options: | <p><code>nx</code> Used to specify the number of 32-bit frames to send. Default is 1. Use x=0 to run the test indefinitely</p> |
| Examples: | <p><code>iel8 p4 n100</code> (Runs the test for 100 frames on IMA TDM4)</p> <p><code>iel8 p7 n0</code> (Runs the test indefinitely on IMA TDM7)</p> |

IMA SRAM (ISRAM Command)

Test Description

The IMA SRAM is accessed through the IMA chip. It is checked by using the same algorithm as for the main memory (see [60x Bus Main Memory \(M60X Command\) on page 41](#)). This test is long and lasts approximately 50 seconds.

ISRAM Command

| | |
|--------------|---|
| Name: | ISRAM |
| Syntax: | <code>isram</code> |
| Description: | Tests the SRAM used by the IMA chip. A global result is displayed with intermediate error messages in case of faults. |

LEDs (LED Command)

Test Description

The only way to check that the LEDs work correctly is to check that they switch on when requested. When turned on, bi-color LEDs are first lighted in red for half a second and then in green.

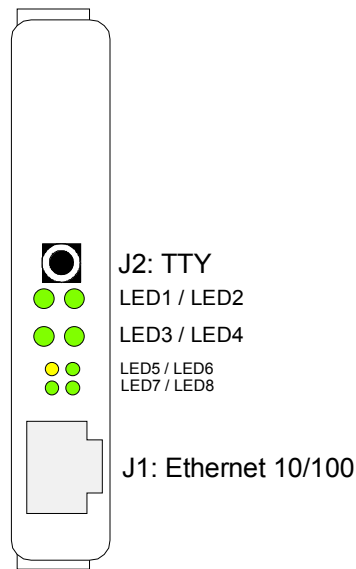


Figure 5-18. LED Configuration

Table 5-7. LED Definitions

| LED | Function |
|-----|--|
| 0 | User-programmable red LED (on the board) |
| 1 | Bi-color red-green LED controlled by the first framer |
| 2 | Bi-color red-green LED controlled by the second framer |
| 3 | Bi-color red-green LED controlled by the third framer |
| 4 | Bi-color red-green LED controlled by the fourth framer |
| 5 | User-programmable yellow LED |
| 6 | LXT971 LED driver 3 (Collision status) |
| 7 | LXT971 LED driver 1 (Link Status) |
| 8 | LXT971 LED driver 2 (Duplex Status) |

LED Command

| | |
|--------------|---|
| Name: | LED |
| Syntax: | led |
| Description: | The LEDs are all toggled (ON if OFF, OFF is ON) at each call. When turned on, bi-color LEDs first appear red for half a second and then switch to green. Note that at the end of a test (except LED), Ethernet and framer chips are reset causing their LEDs to be switched off (bi-color LEDs are driven by QuadFALC framers). |

UTOPIA External Loopback (UEL Command)

Test Description

This test verifies the FCC1 UTOPIA bus connectivity on P3 connector. To run this test, the 4539 must be plugged in a PCI carrier card. The 4539 must have a special loopback connector on P3 referenced P3LBTstConn (see [P3LBTstConn and P4LBTstConn Loopback Connectors on page 150](#)). The transmit section of the bus is configured as master whereas the receive section as slave. The transmit section is loop backed to the receive section with the P3LBTstConn. Cells are sent from FCC1 and the test checks that they are returned correctly. Note that since the test does not implement MULTI-PHY on FCC1, the addressing pins are configured as general purpose outputs TXADR(i) pins and general purpose inputs for RXADR(i) pin: the test verifies that the different values 0x0, 0xF, 0x5, 0x9 output on TXADR(i) pins can be read on RXADR(i) pins.

UEL Command

| | |
|--------------|--|
| Name: | UEL |
| Syntax | uel [nx] |
| Description: | This command tests the FCC1/UTOPIA bus connectivity on PMC P3 connector. To run the test, the board must be plugged in a PCI carrier card with a special loopback connector on P3 as shown in P3LBTstConn and P4LBTstConn Loopback Connectors on page 150 . The test first checks that the signal PTENB# on P3 is asserted and then tests the connectivity of the UTOPIA addressing part by configuring FCC1 TXADDR(i) pins as general purpose outputs, RXADDR(i) pins as general purpose inputs and by checking that different values (0x0, 0xF, 0x5, 0xA) output on TXADDR(i) can be read on RXADD(i). Then for success, FCC1 is configured for mono-PHY operation to send AAL5 frames and check that they are correctly returned (transmit section of UTOPIA is configured as master and receive as slave). |

Options: nx
 Used to specify the number of 32-bit frames to send.
 Default is 1. Use x=0 to run the test indefinitely.

Examples: **uel n1000** (exchanges 1000 frames).
 uel n0 (indefinitely exchanges frames).

AGENCY (AGENCY Command)

Test Description

This test performs Framers and Ethernet external loopback tests in parallel to generate traffic and CPU load. The purpose of this test is to perform Electromagnetic Compatibility (EMC) measurements.

AGENCY Command

Name: AGENCY

Syntax agency [nx] [mode]

Description: This command runs simultaneous Ethernet and framers external loopback tests to generate traffic for EMC measurements. External loopback connectors must be plugged in the T1/E1/J1 and Ethernet ports. Framing mode used for T1/E1/J1 ports is E1. HDLC frames are generated from MCCs on ports 0,1, and 2 whereas ATM AAL5 frames are generated from FCC2 through IMA chip on port 3. Note that if the board is not equipped with an IMA chip, HDLC frames are generated on port 3 as for other ports.

Options: nx
 Used to specify the number of 32-bit frames to send on each port.
 Default is 1. Use x=0 to run the test indefinitely.

mode
 Specifies the connection mode (default is ETH_10FD).
 ETH_10HD 10 BT half duplex
 ETH_10FD 10 BT full duplex
 ETH_100HD 100 BT half duplex
 ETH_100FD 100 BT full duplex
 ETH_AN Auto-negotiation

Examples: **agency n10000 eth_100fd**
 Exchanges 10000 frames on each port.
 On T1/E1/J1 ports 0, 1 and 2, HDLC frames (256 bytes) are exchanged on 64 kbps channels.
 On T1/E1/J1 port 3, ALL5 frames (288 bytes) are exchanged on a 500 kbps VCI.

On the Ethernet port, frames (512 bytes) are exchanged with 100 BT Full Duplex configuration (Note that the Ethernet driver is not optimized for speed and the real data speed is a little lower than the ATM AAL5 VCI).

agency n0

Indefinitely exchanges frames on each port.

On the Ethernet port, frames are exchanged with default 10 BT Full Duplex configuration.

Testing the Product (PROD Command)

Test Description

During the manufacturing process or during the life of a 4539 board, a special command named **PROD** is available to run a series of tests whose purpose is to test as many features as possible. This command runs more tests than the POST, since it assumes that the board has special connectors connected to it allowing it to run external loopback tests on T1/E1/J1 ports, the Ethernet Port, P3 UTOPIA port, and optional IMA TDMs on P4.

[Table 5-8](#) provides a list of tests for the **PROD** command in order of execution. The tests that are not run in POST have their '**Test Title**' name displayed in italics: they are the tests that need special cables or connectors.

Table 5-8. PROD Command Test List

| Test Title | BIST Command | Reference |
|------------------------------------|---------------------|--|
| 1 - Test Local SDRAM memory | MLOC | <i>Local Memory (MLOC Command) on page 43</i> |
| 2 - Test CAM memory | MCAM | <i>CAM (MCAM Command) on page 43</i> |
| 3 - Test Serial EEPROM | MSRPROM | <i>Serial EEPROM (MSRPROM Command) on page 44</i> |
| 4 - Test CAM chip detection | No Command | Part of <i>Chip Detection (CHIPREV Command) on page 46</i> |
| 5 - Test Framers chip detection | No Command | |
| 6 - Test Ethernet chip detection | No Command | |
| 7 - Test IMA chip detection | No Command | |
| 8 - Test Framers chip reset signal | FRST | <i>T1/E1/J1 Framers Reset (FRST Command) on page 47</i> |

Table 5-8. PROD Command Test List (cont)

| Test Title | BIST Command | Reference |
|--|---------------------|---|
| 9 - Test Ethernet chip reset signal | ERST | <i>Ethernet Interface Reset (ERST Command) on page 48</i> |
| 10 - Test IMA chip reset signal | IRST | <i>IMA Reset (IRST Command) on page 48</i> |
| 11 - Test CAM chip reset signal | MCAMRST | <i>CAM Reset (MCAMRST Command) on page 49</i> |
| 12 - Test PCI chip local interrupt signal | PINT | <i>PCI Controller Interrupt (PINT Command) on page 52</i> |
| 13 - Test Framers chip interrupt signal | FINT | <i>T1/E1/J1 Framers Interrupt (FINT Command) on page 49</i> |
| 14 - Test Ethernet chip interrupt signal | EINT | <i>Ethernet Interface Interrupt (EINT Command) on page 50</i> |
| 15 - Test IMA chip interrupt signal | IINT | <i>IMA Interrupt (IINT Command) on page 51</i> |
| 16 - Test Framers internal loopback - independent mode | FIL IND | <i>Framers In Internal Loopback (FIL Command) on page 57</i> |
| 17 - Test Framers internal loopback - independent mode | FIL MUL | <i>Framers In Internal Loopback (FIL Command) on page 57</i> |
| 18 - Test Framers pass-through mode (0->1, 2->3) | FPT P0123 | <i>Framers In Pass-Through Mode (FPT Command) on page 62</i> |
| 19 - Test Framers pass-through mode (1->0, 3->2) | FPT P1032 | <i>Framers In Pass-Through Mode (FPT Command) on page 62</i> |
| 20 - Test Ethernet internal loopback | EIL | <i>Ethernet In Internal Loopback (EIL Command) on page 66</i> |
| 21 - Test IMA port clocks | ICLK | <i>IMA TDM Clocks (ICLK Command) on page 78</i> |
| 22 - Test IMA port 0 internal loopback | IIL P0 | <i>IMA Internal Loopback (IIL Command) on page 68</i> |
| 23 - Test IMA port 1 internal loopback | IIL P1 | |
| 24 - Test IMA port 2 internal loopback | IIL P2 | |
| 25 - Test IMA port 3 internal loopback | IIL P3 | |
| 26 - Test IMA SRAM | ISRAM | <i>IMA SRAM (ISRAM Command) on page 81</i> |
| 27 - Test Ethernet external loopback | EEL | <i>Ethernet In External Loopback (EEL Command) on page 68</i> |
| 28 - Test Framers external loopback | FEL | <i>Framers In External Loopback (FEL Command) on page 59</i> |
| 29 - Test Framers switched mode | FSW | <i>Framers In Switched Mode (FSW Command) on page 64</i> |
| 30 - Test FCC1/UTOPIA on P3 in external loopback | UEL | <i>UTOPIA External Loopback (UEL Command) on page 83</i> |
| 31 - Test Clocks of optional P4 IMA TDMs | ICLK8 | <i>IMA Optional TDMs Clocks (ICLK8 Command) on page 78</i> |

Table 5-8. PROD Command Test List (cont)

| Test Title | BIST Command | Reference |
|--|--------------|---|
| 32 - Test IMA TDM4 on P4 external loopback | IEL8 P4 | IMA Optional TDMs External Loopback (IEL8 Command) on page 80 |
| 33 - Test IMA TDM5 on P4 external loopback | IEL8 P5 | IMA Optional TDMs External Loopback (IEL8 Command) on page 80 |
| 34 - Test IMA TDM6 on P4 external loopback | IEL8 P6 | IMA Optional TDMs External Loopback (IEL8 Command) on page 80 |
| 35 - Test IMA TDM7 on P4 external loopback | IEL8 P7 | IMA Optional TDMs External Loopback (IEL8 Command) on page 80 |

The result of the PROD command for a full equipped board is displayed on the console as follows:

```

***** PROD BEGIN *****
 1 - Test Local SDRAM ..... OK
 2 - Test CAM memory ..... OK
 3 - Test Serial EEPROM ..... OK
 4 - Test CAM chip detection ..... OK
 5 - Test Framers chip detection ..... OK
 6 - Test Ethernet chip detection ..... OK
 7 - Test IMA chip detection ..... OK
 8 - Test Framers chip reset signal ..... OK
 9 - Test Ethernet chip reset signal ..... OK
10 - Test IMA chip reset signal ..... OK
11 - Test CAM chip reset signal ..... OK
12 - Test PCI chip local interrupt signal ..... OK
13 - Test Framers chip interrupt signal ..... OK
14 - Test Ethernet chip interrupt signal ..... OK
15 - Test IMA chip interrupt signal ..... OK
16 - Test Framers internal loop-back - independent mode .. OK
17 - Test Framers internal loop-back - multiplexed mode .. OK
18 - Test Framers pass-through mode (0->1,2->3) ..... OK
19 - Test Framers pass-through mode (1->0,3->2) ..... OK
20 - Test Ethernet internal loop-back ..... OK
21 - Test IMA ports clocks ..... OK
22 - Test IMA port 0 internal loop-back ..... OK
23 - Test IMA port 1 internal loop-back ..... OK
24 - Test IMA port 2 internal loop-back ..... OK
25 - Test IMA port 3 internal loop-back ..... OK
26 - Test IMA SRAM ..... OK
27 - Test Ethernet external loop-back ..... OK
28 - Test Framers external loop-back ..... OK
29 - Test Framers switched mode ..... OK
30 - Test FCC1/UTOPIA on P3 in external loop-back ..... OK
31 - Test Clocks of optional P4 IMA TDMs . ..... OK
32 - Test IMA TDM4 on P4 external loop-back ..... OK
33 - Test IMA TDM5 on P4 external loop-back ..... OK
34 - Test IMA TDM6 on P4 external loop-back ..... OK
35 - Test IMA TDM7 on P4 external loop-back ..... OK
** PROD OK ** PROD OK ** PROD OK ** PROD OK ** PROD OK ** PROD OK ** PROD OK **

```

At the end of the PROD command, a repeated text pattern on one line indicates the global result; if it is successful, the pattern is ****PROD OK** else **** PROD FAILED**.

When a `PROD` test fails, details are displayed.

The Main SDRAM test (see [60x Bus Main Memory \(M60X Command\) on page 41](#)) is not run within the `PROD` command: this test cannot be integrated to the `PROD` command and must be run alone to fully test the 4539 board.

PROD Command

Name: `PROD`

Syntax: `prod`

Description: Runs a set of elementary tests for product qualification. The command for each test is displayed with individual result reports. a global report gives the result of the product test.

Running a Command Several Times (LOOP Command)

Description

The `LOOP` command allows running the same command several times or indefinitely. Some commands cannot be looped: see [Appendix C Monitor Commands on page 117](#).

LOOP Command

Name: `LOOP`

Syntax: `loop n cmd`

Description: Used to run a test several times. If the looped command is a test, looping stops on the first failure.

Options: `n` Specify the number of loops, range 0-65535
`0=` infinite loop
`cmd` A command

Example: `loop 10 led`
Executes the LED test 10 times.

Stopping a Command (S Command)

Some commands such as loopback tests can last a long time or run indefinitely. To show the user that the command is in progress, the `MONITOR` displays a spinning top. The user can stop a running command by pressing `S` followed by carriage return. When a test is stopped, no report on the global result is displayed: just `Process stopped` is displayed followed by the prompt (`>`).

Example 5-1. Stopping a Command

```
>fil n0 p1  
  
s  
  
Process stopped  
>
```


Flashing Firmware Using the TTY (LOADFC Command)

Description

The **LOADFC** command is a useful utility to flash a firmware into the FLASH EEPROM. It allows programming an Operational Firmware as well as the 4539 Boot Firmware itself. The image of the firmware to flash must be coded with ELF format. The image is first transferred by a remote machine into the 4539 main SDRAM through the TTY port and then analyzed and flashed by the 4539 Boot Firmware itself. Part of the 4539 Boot Firmware code runs in main SDRAM so that it can be reprogrammed in FLASH EEPROM by itself. The size of the image file has to be less than the MONITOR file transfer area size reserved by the MONITOR in main SDRAM (see [SDRAM Mapping on page 111](#)). The image file is transferred by the remote machine using XModem protocol. For example, this protocol is supported by the Microsoft Windows NT HyperTerminal application.

The remote machine and the 4539 board have to be interconnected with an RS232 cable. Before using the **LOADFC** command, the user must select the TTY baud rate at which the transfer will operate. If the cable is not too long and of good quality, the speed can be as high as 115200 baud (as long as the remote machine supports this baud rate). If you need to change the baud rate, use the **BAUDRATE** command (see [Changing the Baud Rate \(BAUDRATE Command\) on page 26](#)), you must reset the 4539 board to have the TTY controller configured at the new baud rate.

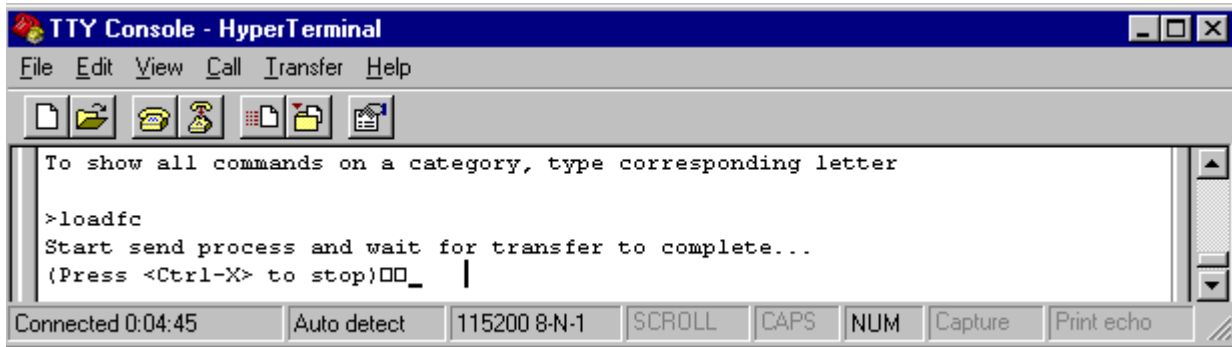
The X-modem protocol uses a packet size of 128 or 1024 bytes: the MONITOR will adapt to the size selected on the remote machine.

LOADFC – Example with Windows NT HyperTerminal Application

[Example 6-1](#) shows how to flash the 4539 boot firmware with the Windows HyperTerminal application. The application is supposed to run on a remote machine connected to the 4539 board with a RS232 cross-connector. In this example, the HyperTerminal session (which is named ‘TTY Console’) uses COM1 port at baud rate 115200, no parity, 1 stop bit: the 4539 board must be configured with the same TTY parameters.

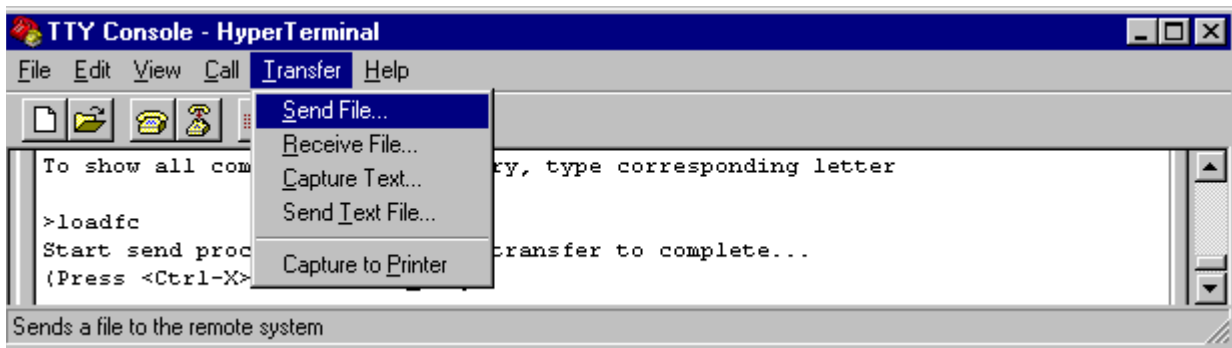
To start the process, input the **LOADFC** command on the TTY console as shown in [Example 6-1](#).

A message instructs the user to start sending the image file to flash or to press <Ctrl-X> to abort the process. The 4539 board starts notifying the remote machine that it is ready to transfer by sending x-modem ‘NAK’ messages which are displayed on the TTY Console as small squares after the message (*Press <Ctrl-X> to stop*).



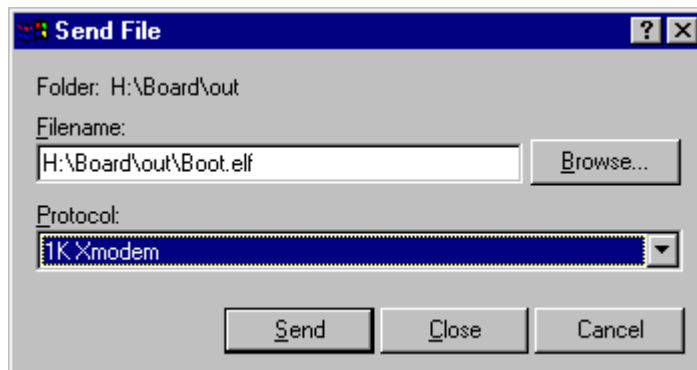
Example 6-1. LOADFC - Step 1

To send the image file to flash, the user must select the *Transfer* and *Send File* options on TTY Console as shown in [Example 6-2](#).



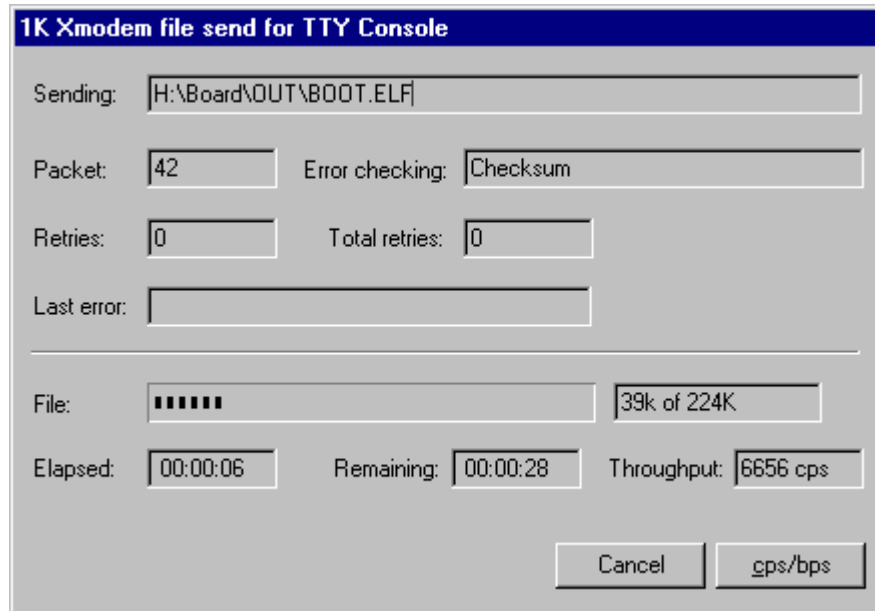
Example 6-2. LOADFC - Step 2

The user is then instructed to select the image file to transfer, the protocol (1K Xmodem or Xmodem) and press Send button.



Example 6-3. LOADFC - Step 3

Next, a window shows the progression of the transfer.



Example 6-4. LOADFC - Step 4

At the end of the transfer, the user has to confirm the flash process (message: `Please confirm beginning of flash process [Y/N]:`). If the firmware to flash overwrites the 4539 Boot Firmware (which is the case in the current example), the user is warned (message: `WARNING: At least 7 section(s) will be flashed over present monitor code.`) and instructed to confirm again (message: `Are you sure to start the flash process [Y/N]:`).

If the 4539 Boot Firmware was overwritten during the flash process, the user has to press a key to reset the board (the case of this example) else the monitor is ready for a new command.

```

>loadfc
Start send process and wait for transfer to complete...
(Press <Ctrl-X> to stop)
File transfer succeeded: flashing ELF sections to memory...
ELF sections will be flashed from address FF800000h
Please confirm beginning of flash process [Y|N]: y
WARNING: At least 7 section(s) will be flashed over present monitor code.
Are you sure to start the flash process [Y|N]: y
Flashing section #1 at FF800000h (32 bytes): . OK
Flashing section #2 at FFF00100h (12032 bytes): .. OK
Flashing section #3 at FFF03000h (2560 bytes): . OK
Flashing section #4 at FFF03A00h (256 bytes): . OK
Flashing section #5 at FFF03B00h (1520 bytes): . OK
Flashing section #7 at FFF040F0h (140728 bytes): ..... OK
Flashing section #8 at FFF266A8h (37936 bytes): ..... OK
Flashing section #9 at FFF2FAD8h (232 bytes): . OK
Press any key to reset the adapter...
  
```

Example 6-5. LOADFC - Step 5

In [Example 6-5](#), the user can see that the start address (FF800000h) where the firmware is flashed is automatically deduced from the image file. The user can force this address by putting it in the `LOADFC` command. In the `LOADFC` command, the user can specify the checksum computation type with option `{cs|crc}`: the HyperTerminal application will adapt to both of them.

LOADFC Command

| | | | | | | | |
|-------------------|--|-------------------|---|-----------------|--|------------------|---------------------------------|
| Name: | LOADFC | | | | | | |
| Syntax: | <code>loadfc lfc [-mode] [-bin] [{+ -}offset address]</code> | | | | | | |
| Description: | Flashes an ELF coded program transferred through the TTY console interface using X-Modem[1K] or X-Modem[1k]-CRC protocol. ELF section addresses are used as default flash memory addresses. However, an offset value (or an absolute address) can be used to fix the real address in the flash memory: | | | | | | |
| Options: | <table border="0"> <tr> <td><code>mode</code></td> <td>Xmodem[1K] mode {<code>crc cs</code>}</td> </tr> <tr> <td><code>cs</code></td> <td>Selects XModem[1K] with checksum control sequence (default protocol)</td> </tr> <tr> <td><code>crc</code></td> <td>Selects XModem[1K]-CRC protocol</td> </tr> </table> | <code>mode</code> | Xmodem[1K] mode { <code>crc cs</code> } | <code>cs</code> | Selects XModem[1K] with checksum control sequence (default protocol) | <code>crc</code> | Selects XModem[1K]-CRC protocol |
| <code>mode</code> | Xmodem[1K] mode { <code>crc cs</code> } | | | | | | |
| <code>cs</code> | Selects XModem[1K] with checksum control sequence (default protocol) | | | | | | |
| <code>crc</code> | Selects XModem[1K]-CRC protocol | | | | | | |

| | |
|----------------------|--|
| <code>bin</code> | Flashes file content without any interpretation. If used, the <code>address</code> parameter is mandatory. |
| <code>offset</code> | Added to each section address to get the flash memory address. Offset is expressed in hexadecimal (without 0x). |
| <code>address</code> | Used as the flash memory address of the program section whose address is the lowest; other sections are flashed accordingly. Address is expressed in hexadecimal (without 0x). |

Ethernet Loading (LOADF/LOADM Commands and Auto-Boot)

Overview

This section describes the process of loading a Firmware through the Ethernet interface of a 4539 board.

Firmware loading process can be performed by one of the following three methods:

1. During the boot initialization sequence (called auto-boot).
2. While running the `LOADF` command.
3. While running the `LOADM` command.

Because each method is doing the same thing (downloading, loading and running a firmware), this section describes the different steps of that process. Although these three methods share the same code, they run this code under their own specific context. So, some of the steps detailed below may not be used by all methods.

In order to configure the firmware server see [Appendix F Server Consideration on page 141](#).

Initial Configuration

The way the firmware will be loading while the auto-boot or the `LOADF/LOADM` command executes depends on this initial configuration. It defines the set of data the module has to know before or during the loading process.

Local Ethernet/MAC Address

This data defines the Ethernet MAC address of the board.

It is read from the Serial EEPROM and managed with the `mac` command (see [Displaying the Board Ethernet MAC Address \(MAC Command\) on page 28](#)). It is read once at boot time and is used to initialize the different network layers.

This data is mandatory.

Local IP Address and Local IP Mask

This data defines the IP address of the board on the local network and the IP mask used to separate the network part from the host part of an IP address.

They are read from the Serial EEPROM and managed with the `localip` and `maskip` commands ([Changing the Board IP Local Address \(LOCALIP Command\) on page 28](#) and [Changing the Board IP Subnet Mask \(MASKIP Command\) on page 28](#)). They are read once at boot time and are used to initialize the different network layers.

If the local IP address of the board is not defined (that is, if the content of the Serial EEPROM object is an invalid IP address), the loading process will first try to determine it.

Once the local IP address is determined, it is stored in the firmware global data but not modified in the Serial EEPROM. This will explain why the execution of the same command twice (`LOADF` for example) will not generate the same result.

The IP mask is used to check if a received request is broadcast on the local sub-network, that is, if the board is (also) the destination of that request. Its value is not checked and is used 'as is'.

This data is not mandatory.

Server IP Address

This data is used to define the host on the local network that will (or has to) reply to the next TFTP network request sent by the board. If it is set to the IP broadcast value (255.255.255.255), the first host that replies positively to the request will be used as the default server for the next step of the loading process.

The initial value of that data is different if the auto-boot process is used or if a `LOADF/LOADM` command is being executed:

- It is read from the Serial EEPROM (and managed with the `serverip` command – see [Changing the Server IP Address \(SERVERIP Command\) on page 28](#)) at boot time and used ONLY by the auto-boot process if it must be engaged.
- It is given on the command line as a mandatory parameter of the `LOADF/LOADM` command.

As stated above, the server IP address can be set to the IP broadcast value. This implies that the Ethernet loading process has to handle the possibility that several hosts may reply to a request sent by the board. The first positive reply determines the server host and its address will be used for the next step of the process (if no other information from the received reply invalidates it).

Firmware Filename

This data defines the name of the file that contains the firmware to load.

The initial value of that data is different if the auto-boot process is used or if a `LOADF/LOADM` command is being executed:

- It is read from a configuration file previously loaded by the auto-boot process. The name of the configuration file is first deduced from the bytes of the local IP address or from a host reply.
- It is given on the command line as a mandatory parameter of the `LOADF/LOADM` command.

Configuration Filename

This data defines the name of the file that contains a configuration that is interpreted by the loading process.

This file is a text file stored on any host of the local network. It is requested by the auto-boot process only, using the TFTP protocol (RFC 783).

At this time, the configuration defines the parameters listed in [Table 6-1](#).

Table 6-1. Configuration Parameters

| Syntax | Status | Description | Comment |
|-------------------------|-----------|---|---|
| TFTPSERVER-n1s.n2.n3.n4 | Optional | Defines the IP address of the host that stores the firmware file name | n1, n2, n3, and n4 are decimal values in the range 0–255. If not defined, this is the address of the configuration file server that will be used to request to firmware |
| FIRMWARE=filename | Mandatory | Defines the name of an ELF format file that contains the firmware to load and execute | filename string is built with all printable characters that appear after the = sign. The maximum length of the string is 128 characters (including the terminal “\0”). |

The content of the configuration file is loaded into a memory buffer and every parameter is searched from the beginning of that buffer (no parameter sequence is requested). The search of a parameter is made by searching a characters string built from the concatenation of the parameter name and the '=' sign. No space nor any other character is allowed between the parameter name and the '=' sign.

A configuration file may contain any other character since it doesn't contain any sub-string built from the concatenation of a parameter name and the '=' sign.

Configuration Management

Once initial configuration is set, its data is used and/or changed by the firmware loading process according to the context in which it is run.

Auto-Boot Process Configuration Management

This section summarizes the configuration management during the auto-boot process.

Note that the local IP address `gLocalIP` and the server IP address `gServerIP` are updated by the auto-boot process only if they were undefined before. The configuration file name `gCFGFilename` is updated only if the local IP address is updated.

Moreover, the firmware file name `gFWFilename` is deduced from the auto-boot process.

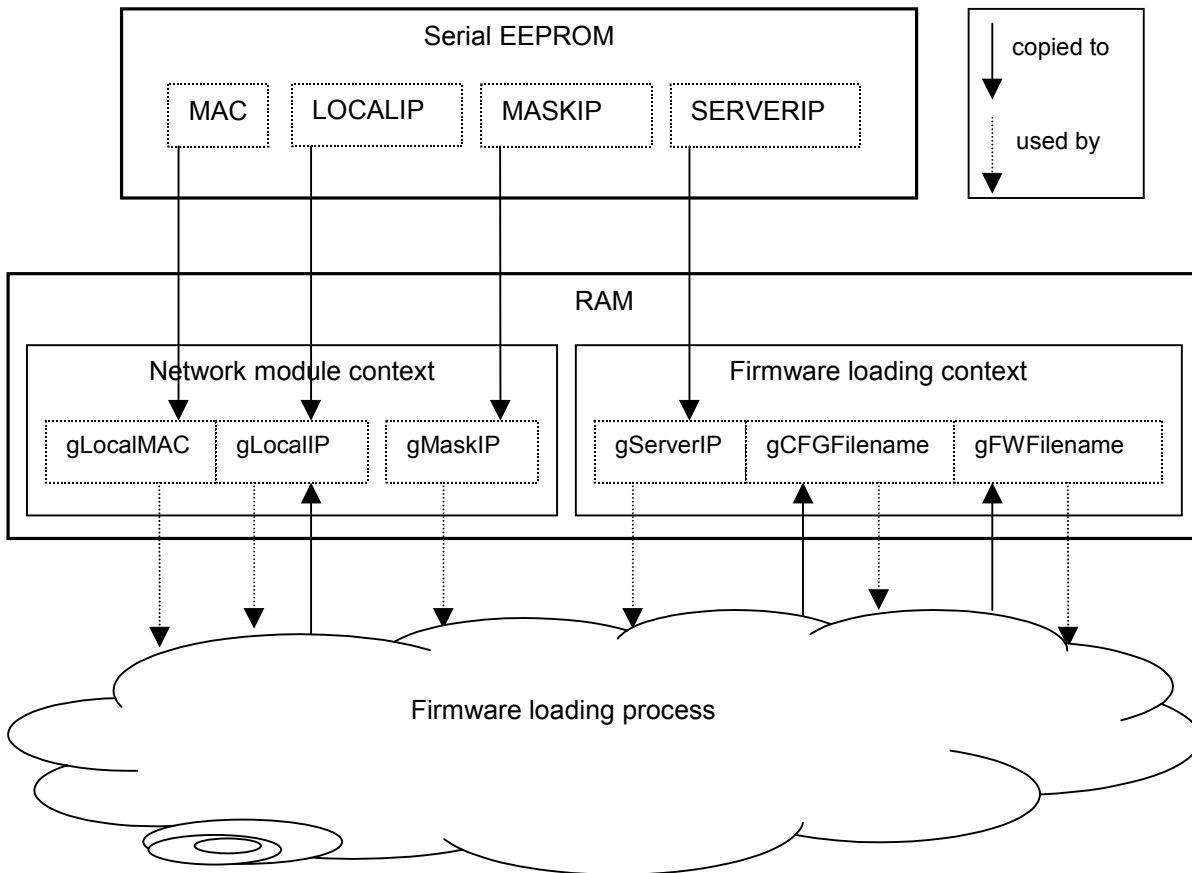


Figure 6-1. Auto-Boot Configuration Management

LOADF/LOADM Configuration Management

This section summarizes the firmware loading process configuration management while executing the `LOADF` and `LOADM` commands.

Note that the `SERVERIP` Serial EEPROM variable and the configuration file `gCFGFilename` are not used in that context.

Moreover, the server IP address `gServerIP` and the firmware filename `gFWFilename` have to be known by the process (these are mandatory parameters of the commands).

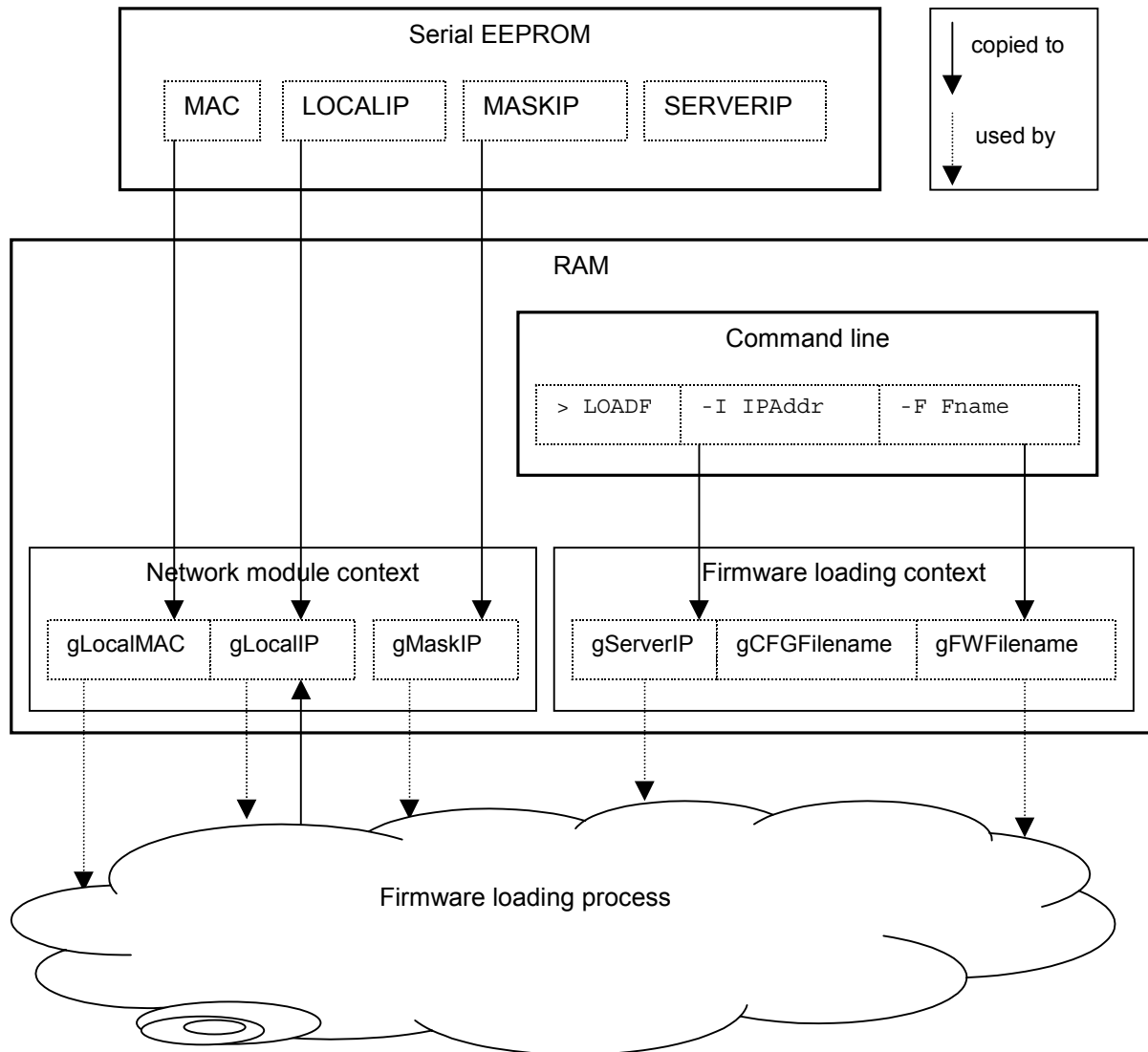


Figure 6-2. LOADF/LOADM Configuration Management

The Firmware Loading Process in Detail

As the initial configuration is known, a firmware can be loaded via the Ethernet interface. According to this configuration, the state machine of the loading process can run all of its states or avoid some of them.

[Figure 6-3 on page 100](#) shows the state machine of the loading process.

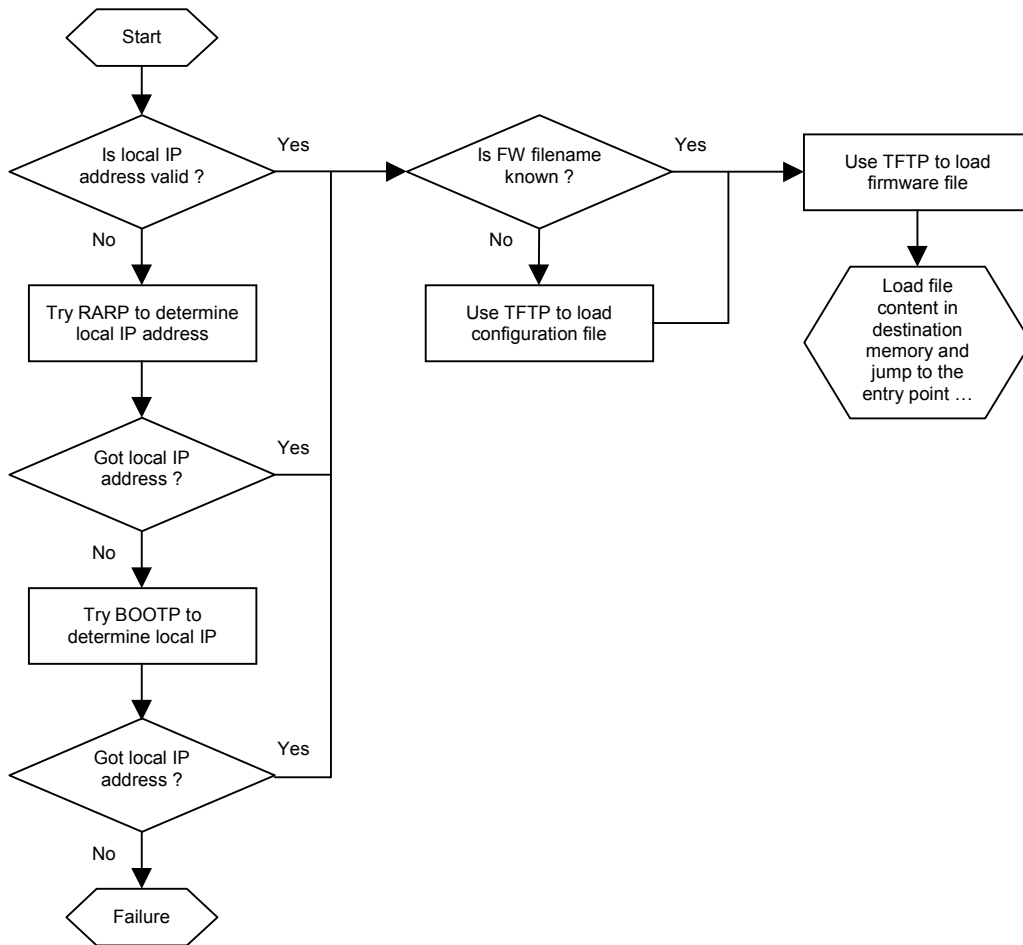


Figure 6-3. Firmware Loading Process

RARP Module

RARP (Reverse Address Resolution Protocol) is used to determine the local IP address of equipment on the local network according to its Ethernet/MAC address. RARP is a data-link level protocol (RFC 903).

The board always sends RARP requests with a broadcast destination address.

The local IP address is read from the first positive reply received by the board.

Conversely, if reply waiting times out, the board sends a RARP request again. The board sends up to three RARP requests before determining that no RARP server is configured on the network. In that case, the board tries the BOOTP protocol.

BOOTP Module

BOOTP is used to determine the local IP address of equipment on the local network according to its Ethernet/MAC address (RFC 951). The main difference between BOOTP and RARP is that BOOTP is a packet-level protocol, that is, the BOOTP primitives are transported by the IP/UDP layer (which offers a few other guarantees of the content delivered).

The board always sends BOOTP requests with a broadcast destination address.

The local IP address is read from the first positive BOOTP reply received by the board.

Finally, BOOTP also defines a field in the reply that can be used by the BOOTP server to store the name of the file to download. If this field is not empty in the BOOTP reply, the board used its value as the name of the configuration file to download (instead of the default one built from the local IP address).

Conversely, if reply waiting times out, the board sends a BOOTP request again. The board sends up to three BOOTP requests before determining that no BOOTP server is configured on the network. In that case, the board determines that firmware loading has failed.

TFTP Transfers

TFTP is used to transfer files on IP/UDP layer (RFC 783).

As its local IP address is known, the board uses TFTP to download one or two files, depending whether the name of the firmware file is known or not.

Configuration File Transfer

A configuration file is only used for the auto-boot process ([Configuration Filename on page 97](#)). Specifically, the state machine requests it if doesn't know the name of the firmware to download.

If the name of the configuration file is not known, the board builds it from its IP address by concatenating the characters strings made of the hexadecimal values of the four bytes of the IP address. [Figure 6-4 on page 101](#) gives an example of this.

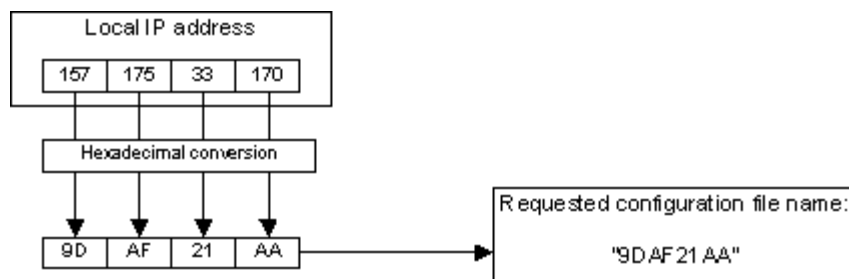


Figure 6-4. Example of Building the Name of the Configuration File

Otherwise, the board already knows the name of the configuration file (if BOOTP was used to determine its IP address, for example).

Once the name of the configuration file is known, the board sends TFTP read requests to the current server IP address or broadcasts them.

The IP address of the sender of the first positive reply received by the board is used as the current server IP address.

Once the configuration file has been received, it is parsed to read the parameters the board needs to continue the loading process.

If the `TFTPSERVER` parameter is found (*Configuration Filename on page 97*), the address it defines is used as the address of the host where the firmware file is stored instead of the configuration file server address.

If the `FIRMWARE` parameter is found (*Configuration Filename on page 97*), the characters string it defines is used as the name of the file to download. If it is not found, the loading process fails.

Conversely, if reply waiting times out, the board sends a TFTP request again. The board sends up to three TFTP requests before determining that no TFTP server is configured on the network. In that case, the board determines that firmware loading has failed.

Firmware File Transfer

Knowing the name of the firmware file, the board sends TFTP Read Requests to the host the server IP address defines or broadcasts them.

The IP address of the sender of the first positive reply received by the board is used as current server IP address.

Once the firmware file has been received, it is parsed as an ELF formatted one. If all checks succeed, the sections of the ELF files are written to the destination memory (that is the RAM for the auto-boot process and the `LOADM` command or the `FLASH` for the `LOADF` command).

Starting Firmware Loading

Auto-Boot Process

The firmware loading is done during the boot initialization sequence only if the configuration bit `MONP_RUN_FW_ETHERNET` is set in Serial EEPROM byte `0xC2`.

LOADF Command

Description

This command is used to download a firmware from a TFTP server to the FLASH memory of the communications controller. The user has to know where and in which file this firmware is stored, that is, the user has to give the IP address of the TFTP server and the filename that contains the firmware.

Once the file has been loaded, it is read and flashed to the addresses specified in the ELF sections of the firmware. An offset can also be specified on the command line to translate these addresses before flashing the sections.

LOADF Command

Name: LOADF

Syntax: `lf [-bin] -I IpAddr -F Fname [Addr]`

Description: Flashes a program through the Ethernet interface to a specified location.

I IpAddr TFTP server IP address, expressed in decimal (for example, 192.134.47.90)

F Fname Firmware file name (use "Fname" if case sensitive).

Options:

addr Flashed firmware address in hexadecimal (without 0x). Specifies the location where the firmware will be flashed.

bin Flashes file content without any ELF interpretation. If `bin` is used, 'Addr' parameter is mandatory.

LOADM Command

Description

This command is used to download a firmware from a TFTP server to the RAM of the communications controller. The user has to know where and in which file this firmware is stored, that is, the user has to provide the IP address of the TFTP server and the file name that contains the firmware.

Once the file has been loaded, it is read and loaded to the RAM addresses specified in the ELF sections of the firmware. An offset can also be specified on the command line to translate these addresses before copying the sections.

LOADM Command

Name: LOADM

Syntax: `lm [-bin] -I IpAddr -F Fname [Addr]`

Description: Loads a program in memory through the Ethernet.

I IpAddr TFTP server IP address, expressed in decimal (for example 192.134.47.90)

F Fname Firmware file name

Options:

addr Flashed firmware address in hexadecimal (without 0x). Specifies the location where the firmware will be flashed.

bin Flashes file content without any ELF interpretation.. If `bin` is used, 'Addr' parameter is mandatory.

Memory Debugging Commands

Overview

The memory debugging commands allow reading or modifying a memory location. A memory location is a processor address, and so the commands are not restricted to the main 60X SDRAM or local SDRAM: these commands also allow access to the chip registers. For local space mapping, refer to [4539 Boot Firmware Mapping on page 111](#). Since these commands allow modification of any memory location; use them with care.

The commands are:

- **MREAD** to read a memory location
- **MWRITE** to write a memory location
- **MDUMP** to dump a memory block
- **MFILL** to fill a memory block with a pattern
- **MMOVE** to move a memory block.

Note that the commands start with the letter **m** for **m**emory

Reading a Memory Location (MREAD Command)

Name: MREAD

Syntax: `mr addr access`

Description: Reads the content of a memory location (`addr`) in the specified format.

| | |
|---------------------|--|
| <code>addr</code> | Memory location to read, in hexadecimal (without 0x) |
| <code>access</code> | Access Type: |
| 8 | Read one byte (8 bits) |
| 16 | Read one word (16 bits) |
| 32 | Read a double word (32 bits) |

Writing a Memory Location (MWRITE Command)

Name: MWRITE

Syntax: `mw addr value access`

Description: Writes a value (`value`) at address (`addr`) memory location in the specified format (`access`). Whatever value size is, a mask is done in the specified format.

| | |
|---------------------|--|
| <code>addr</code> | Memory location to write to, in hexadecimal (without 0x) |
| <code>value</code> | Value to write, in hexadecimal (without 0x). |
| <code>access</code> | Access type: |

| | |
|----|---------------------------------|
| 8 | Write one byte (8 bits) |
| 16 | Write one word (16 bits) |
| 32 | Write one double word (32 bits) |

Dumping a Memory Block (MDUMP Command)

| | | |
|--------------|--|--|
| Name: | MDUMP | |
| Syntax: | md addr length | |
| Description: | Dumps a memory block from addr to addr+length. | |
| | addr | Starting block address in hexadecimal (without 0x) |
| | length | Block length in bytes (in decimal) |

Filling a Memory Block (MFILL Command)

| | | |
|--------------|--------------------------------------|---|
| Name: | MFILL | |
| Syntax: | mf addr length pattern | |
| Description: | Fills a memory block with a pattern. | |
| | addr | Starting block address in hexadecimal (without 0x) |
| | length | Block length in bytes (in decimal) |
| | pattern | Pattern to fill the block, expressed in hexadecimal (without 0x). This pattern is one byte. |

Moving a Memory Block (MMOVE Command)

| | | |
|--------------|--|---|
| Name: | MMOVE | |
| Syntax: | mm AddrSRC AddrDest length | |
| Description: | Moves a length-wide memory block from source address (AddrSRC) to destination address (AddrDest). Avoid overlapping when specifying length, AddrSrc, and AddrDest. | |
| | AddrSrc | Block start address in hexadecimal (without 0x) |
| | AddrDest | Block destination in hexadecimal (without 0x) |
| | length | Block size in bytes |

Resetting the Board (RESET Command)

Description

Resetting the board may be useful for restarting the 4539 Boot Firmware in a new configuration. A reset is achieved by accessing an invalid address that generates a bus monitor exception. The bus monitor exception generates an internal hard reset sequence equivalent to the assertion of the –HRESET pin of MPC8260 processor (the –HRESET pin is bidirectional and is, in this case, driven by the processor itself). Note that the PCI chip (PowerSpan) is not reset and so the PCI configuration is not lost.

RESET Command

Name: RESET

Syntax: rst

Description: Resets the board by performing a hard reset.

Executing a Code From an Address (GO Command)

Description

With the go command, the user can run a code located at an address in FLASH EEPROM or in SDRAM. Before the jump, the 4539 Boot Firmware performs the following operations:

- Gets the PowerSpan SEM3 semaphore to prevent the HOST from accessing the PCI console exchange area (see [Interfacing to the PCI Console on page 115](#)).
- Stops MCCs
- Disables MPC8260 external interrupts
- Resets the CPM
- Resets the chips (CAM, Framers - QuadFALC , IMA - MT9022x, Ethernet interface - LXT971)

GO Command

Name: Go

Syntax: go addr

Description: Jumps to a specified address in order to execute a program. The command cannot be undone.

 addr Beginning address of the program to execute, expressed in hexadecimal (without 0x).

HELP Command

| | |
|--------------|--|
| Name: | HELP |
| Syntax: | h help Command |
| Description: | This command displays a brief description on a <Command> or category. Just pressing <CR> after the prompt (>) will display the available categories. |

Serial EEPROM Mapping

Table A-1. 4539 Serial EEPROM General Mapping

| Add | Len | Mnemonic | Description |
|------|-----|---|---|
| 0x00 | 64 | PSP | PowerSpan registers initial load |
| 0x40 | 4 | BER | Board Equipment register |
| 0x44 | | VPD | VPD and/or custom data – not used on the 4539. |
| 0x90 | 2 | CARD_TIME_LOC | |
| 0x92 | 2 | CARD_DATE_LOC | First serial EEPROM programming date |
| 0x94 | 2 | VPD_SX_LOC | Not used on the 4539. |
| 0x96 | 2 | ARM_SX_LOC | Not used on the 4539. |
| 0x98 | 2 | JSP_SX_LOC | Not used on the 4539. |
| 0x9A | 2 | PSP_SX_LOC | SX number of file used to program PowerSpan area (0x00 to 0x3F) |
| 0x9C | 2 | ISP_SX_LOC | SX number of file used to program the 4539 EPLD |
| 0x9E | 2 | BOO_SX_LOC | SX number of 4539 Boot Firmware |
| 0xA0 | 2 | CARD_SERIAL_L | Less significant word (2 bytes) of the 4539 serial number |
| 0xA2 | 2 | CARD_SERIAL_H | Most significant word (2 bytes) of the 4539 serial number |
| 0xA4 | 2 | VPD_UPDATE_LOC | Not used on the 4539. |
| 0xA6 | 2 | ARM_UPDATE_LOC | Not used on the 4539. |
| 0xA8 | 2 | JSP_UPDATE_LOC | Not used on the 4539. |
| 0xAA | 2 | PSP_UPDATE_LOC | Programming date of PowerSpan area (0x00 to 0x3F) |
| 0xAC | 2 | ISP_UPDATE_LOC | Programming date of EPLD |
| 0xAE | 2 | BOO_UPDATE_LOC | Programming date of the 4539 Boot Firmware |
| 0xB0 | 6 | MONP_MACADD | Board Ethernet MAC address |
| 0xB6 | 4 | MONP_LOCIPADD | Board IP address |
| 0xBA | 4 | MONP_MASKIPADD | Board IP subnet mask |
| 0xBE | 4 | MONP_SERVIPADD | IP server address |
| 0xC2 | 1 | MONP_RUN_FW, MONP_PARITY, MONP_STOPBITS, MONP_BAUDRATE | |
| 0xC3 | 1 | MONP_ECHO, MONP_QUICK_POST | |

Table A-1. 4539 Serial EEPROM General Mapping (cont)

| Add | Len | Mnemonic | Description |
|------|-----|------------------|---|
| 0xC4 | 4 | MONP_FWADD | Address of an operational flashed firmware |
| 0xC8 | 4 | MONP_FWCFGADD | Address of operational flashed firmware configuration |
| 0xCC | 4 | MONP_ROUTERIPADD | IP router address – reserved for a future use. |
| 0xD0 | 28 | Reserved | Unused area |
| 0xFC | 1 | MONP_VMAJOR | Major version of 4539 Boot Firmware software parameter organization |
| 0xFD | 1 | MONP_VMINOR | Minor version of 4539 Boot Firmware software parameter organization |
| 0xFE | 2 | MONP_CHECKSUM | See <i>Serial EEPROM (MSRPROM Command)</i> on page 44. |

Table A-2. Serial EPROM Byte 0xC2 Mapping

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------------|-------|---------------|---------------|-------|-------|
| MONP_RUN_FW | | MONP_PARITY | | MONP_STOPBITS | MONP_BAUDRATE | | |

Table A-3. Serial EPROM Byte 0xC2 Details

| Mnemonic | Value | Description |
|---------------|--------------------------|--|
| MONP_RUN_FW | 0 = MONP_RUN_FW_DISABLE | MONITOR has no firmware to run |
| | 1 = MONP_RUN_FW_FLASH | MONITOR must run a flashed Operational Firmware |
| | 2 = MONP_RUN_FW_ETHERNET | MONITOR must Ethernet-upload and run an Operational Firmware |
| | 3 = reserved value | |
| MONP_PARITY | 0 = MONP_PARITY_NONE | No parity on TTY |
| | 1 = MONP_PARITY_ODD | Odd parity on TTY |
| | 2 = MONP_PARITY_EVEN | Even parity on TTY |
| | 3 = reserved value | |
| MONP_STOPBITS | 0 = MONP_STOPBITS_1 | 1 stop bit on TTY |
| | 1 = MONP_STOPBITS_2 | 2 stop bits on TTY |

Table A-3. Serial EPROM Byte 0xC2 Details (cont)

| Mnemonic | Value | Description |
|-----------------|--------------------------|------------------------|
| MONP_BAUDRATE | 0 = MONP_BAUDRATE_9600 | TTY baud rate = 9600 |
| | 1 = MONP_BAUDRATE_19200 | TTY baud rate = 19200 |
| | 2 = MONP_BAUDRATE_38400 | TTY baud rate = 38400 |
| | 3 = MONP_BAUDRATE_57600 | TTY baud rate = 57600 |
| | 4 = MONP_BAUDRATE_115200 | TTY baud rate = 115200 |
| | 5 to 7 = reserved values | |

Table A-4. Serial EPROM Byte 0xC3 Mapping

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| MONP_ECHO | MONP_POST | | Reserved | | | | |

Table A-5. Serial EPROM Byte 0xC3 Details

| Mnemonic | Value | Description |
|-----------------|-----------------------|--|
| MONP_ECHO | 0 = MONP_ECHO_DISABLE | TTY echo is disabled |
| | 1 = MONP_ECHO_ENABLE | TTY echo is enabled |
| MONP_POST | 0 = MONP_POST_NORMAL | Exhaustive POST |
| | 1 = MONP_POST_QUICK | Boot Firmware compatibility and memory tests |
| | 2 = MONP_POST_NONE | Just Boot Firmware compatibility test |
| | 3 = Reserved value | |

4539 Boot Firmware Mapping

SDRAM Mapping

This section provides information on the 4539 Boot Firmware memory usage in the processor local space for the SDRAM only. 4539 boards are equipped with 64 K SDRAM.

Table A-6. SDRAM Mapping

| From Address | To Address | Size | Description |
|--------------|------------|------------|--|
| 0x00000000 | 0x00003000 | 0x00003000 | Exception vector table |
| 0x00003000 | 0x01000000 | 0x00FFD000 | Unused SDRAM space |
| 0x01000000 | 0x01100000 | 0x00100000 | 4539 Boot Firmware RAM code, data, and stack |
| 0x01100000 | 0x01FF0000 | 0x00EF0000 | MONITOR file transfer area |
| 0x01FF0000 | 0x02000000 | 0x00010000 | MONITOR PCI console exchange area |
| 0x02000000 | 0x04000000 | 0x02000000 | 32 MBytes Unused SDRAM space |

FLASH EEPROM Mapping

The FLASH EEPROM mapping starts at address 0xFF800000 (1st MAP) with the hard reset configuration word. It is redundantly mapped at address 0xFFC00000 (2nd MAP). The reset vector is mapped at address 0xFFB00100 in the last mega-byte of the FLASH. The shaded rows in [Table A-7](#) shows the 4539 Boot Firmware location in the FLASH EEPROM.

Table A-7. 4 MB FLASH EEPROM Mapping

| FLASH Byte | 1st MAP | 2nd MAP | Size | Description |
|------------|------------|------------|------------|--|
| 0x000000 | 0xFF800000 | 0xFFC00000 | 0x0000100 | The first 64-kbyte sector of the FLASH contains the Hardware Configuration word (at addresses 0xFF800000, 0xFF800008, 0xFF800010, 0xFF800018).and remaining space is unused. |
| 0x000100 | 0xFF800100 | 0xFFC00100 | 0x0000FE00 | Unused byte space. |
| 0x010000 | 0xFF810000 | 0xFFC10000 | 0x002F0000 | Free space for Operational Firmware (47* 64 kbytes) |
| 0x300000 | 0xFFB00000 | 0xFFF00000 | 0x00000100 | Unused byte space. |
| 0x300100 | 0xFFB00100 | 0xFFF00100 | 0x000358E0 | 4539 Boot Firmware ROM code |
| 0x3359E0 | 0xFFB359E0 | 0xFFF359E0 | | Unused bytes space. |
| 0x340000 | 0xFFB40000 | 0xFFF40000 | 0x000C0000 | Free space for Operational Firmware. |



NOTE

4539 Boot Firmware ROM code size may change depending on the 4539 Boot Firmware version. Use the MONITOR INFO command to display the actual size (for more information, see [INFO Command on page 33](#)).



NOTE

The Hardware Configuration word is included in the 4539 Boot Firmware. Reprogramming the Boot Firmware will reprogram the Hardware Configuration word.

Interfacing to the PCI Console

B

Introduction

The PCI console is an emulation of a TTY console that enables the user to input commands to the MONITOR through the PCI interface instead of the TTY. A special host application named `iphMonitor` dialogs with the MONITOR through the PCI Console exchange area in the 4539 main SDRAM (see [4539 Boot Firmware Mapping on page 111](#)). The `iphMonitor` application for Solaris and VxWorks systems is provided with the 4539 Board Development Kit. For other systems, the user is advised to change the source of the `iphMonitor` VxWorks application to generate a version that matches the system. This chapter provides details on how the PCI Console exchange area is structured and a 'C' pseudo-code to exchange characters using this interface with the MONITOR.



NOTE

For more information about how to use `iphMonitor`, see the VxWorks and Solaris Maintenance Tool Appendices in the *4539 Board Installation and Maintenance Manual*.

The PCI console exchange area is organized as `MonPCITermIntf` defined as follows:

```
/* Host to board data interface size for terminal emulation */
#define MONPCI_HOSTTOBOARD_DATAINTF_SIZE 80

/* Board to host data interface size for terminal emulation */
#define MONPCI_BOARDTOHOST_DATAINTF_SIZE 32768

/* PCI console exchange area type definition */
typedef struct MonPCITermIntf {

    /* Host to board data exchange area */
    struct {
        word wIn;
        word wOut;
        byte bData[MONPCI_HOSTTOBOARD_DATAINTF_SIZE];
    } HostToBoard;

    /* Board to host data exchange area */
    struct {
        word wIn;
        word wOut;
        byte bData[MONPCI_BOARDTOHOST_DATAINTF_SIZE];
    } BoardToHost;
} MonPCITermIntf_t;
```

In some circumstances, the MONITOR takes the SEM3 semaphore of the PCI chip (PowerSpan) to indicate to the host that the PCI console exchange area is not available (the tag written to SEM3 is 7A): the host must wait for the release of this semaphore before using the PCI console emulation.

The circumstances when the PCI console becomes unavailable are:

- During board reset with RESET command
- During board reset after a flash operation (LOADF or LOADFC command)
- During Main SDRAM test (M60X command)
- On jump to an operational FW after POST (enabled with the AUTORUN F command)
- On jump to a program with the GO command utility

The user can use the following pseudo-codes to put or get a character on the PCI Console exchange area (PciTerm is a pointer to the PCI console area):

To put a character to the board:

```
Temp = PciTerm -> HostToBoard.wIn + 1;
if (Temp = MONPCI_HOSTTOBOARD_DATAINTF_SIZE ) Temp = 0 ;
if (Temp = PciTerm -> HostToBoard.wOut ) return(KO);
HostToBoard.bData[PciTerm -> HostToBoard.wIn]= c;
PciTerm -> HostToBoard . wIn = Temp;
return(OK);
```

To get a character from the board (stored in c variable):

```
Temp = PciTerm -> BoardToHost.wOut;
if ( PciTerm -> BoardToHost . wIn = Temp ) return(KO);
c = PciTerm -> HostToBoard. bData[ Temp ];
Temp ++;
if (Temp = MONPCI_BOARDTOHOST_DATAINTF_SIZE ) Temp = 0;
PciTerm -> HostToBoard . wOut = Temp;
return(OK)
```

Monitor Commands

C

Introduction

Table C-1 provides a list of available commands. The **Loop** column indicates the commands that can be looped with the LOOP command; the **Short** column indicates the commands that have a short-cut.

Table C-1. Monitor Commands

| Command | Short | Description | Reference | Loop |
|----------|-----------|--|--|------|
| AGENCY | | Generates traffic for EMC tests | AGENCY (AGENCY Command) on page 84 | L |
| AUTORUN | AR or RUN | Defines the firmware start mode | AUTORUN Command on page 31 | |
| BAUDRATE | BR | Displays/Changes TTY console baud rate | Changing the Baud Rate (BAUDRATE Command) on page 26 | |
| C | | Displays configuration category commands | Configuration Category Commands on page 11 | |
| CHIPREV | | Detects the chips and displays their version | Chip Detection (CHIPREV Command) on page 46 | L |
| ECHO | E | Displays/changes the console local echo activity | Changing the Echo Mode (ECHO Command) on page 27 | |
| EEL | | Tests the Ethernet port in external loopback | Ethernet In External Loopback (EEL Command) on page 68 | L |
| EIL | | Tests the Ethernet port in internal loopback | Ethernet In Internal Loopback (EIL Command) on page 66 | L |
| EINT | | Tests the Ethernet chip interrupt signal | Ethernet Interface Interrupt (EINT Command) on page 50 | L |
| ERST | | Tests the Ethernet chip reset signal | Ethernet Interface Reset (ERST Command) on page 48 | L |
| FB | | Performs BERT tests on T1/E1/J1 lines | BERT (FB, FRL Commands) on page 73 | |
| FEL | | Tests the framers in external loopback | Framers In External Loopback (FEL Command) on page 59 | L |
| FIL | | Tests the framers in internal loopback | Framers In Internal Loopback (FIL Command) on page 57 | L |
| FINT | | Tests the framers interrupt signals | T1/E1/J1 Framers Interrupt (FINT Command) on page 49 | L |
| FPHY | | Monitors the status of the framers | Framer Line Status (FPHY Command) on page 76 | |

Table C-1. Monitor Commands (cont)

| Command | Short | Description | Reference | Loop |
|----------------|--------------|--|---|-------------|
| FPT | | Test the framers in pass-through mode | <i>Framers In Pass-Through Mode (FPT Command)</i> on page 62 | L |
| FRL | | Configures the framers in remote loopback | <i>BERT (FB, FRL Commands)</i> on page 73 | |
| FRST | | Tests the framers reset signals | <i>T1/E1/J1 Framers Reset (FRST Command)</i> on page 47 | L |
| FSW | | Test the framers in switched mode | <i>Framers In Switched Mode (FSW Command)</i> on page 64 | L |
| FWADDR | FWA | Sets the Firmware address in Serial EEPROM | <i>FWADDR Command</i> on page 31 | |
| FWCFGADDR | FWCA | Sets the Firmware Configuration address in Serial EEPROM | <i>FWCFGADDR Command</i> on page 32 | |
| GO | | Executes a program from an address | <i>Executing a Code From an Address (GO Command)</i> on page 106 | |
| HELP | H | Help | <i>On-line Help</i> on page 13 | |
| I | | Displays information category commands | <i>Information Category Commands</i> on page 10 | |
| ICLK | | Tests the IMA TDMs clocks | <i>IMA TDM Clocks (ICLK Command)</i> on page 78 | L |
| ICLK8 | | Tests the IMA optional TDMs clocks | <i>IMA Optional TDMs Clocks (ICLK8 Command)</i> on page 78 | L |
| IEL | | Tests the IMA with external loopback | <i>IMA External Loopback (IEL Command)</i> on page 71 | L |
| IEL8 | | Tests the IMA Optional TDMs with external loopback | <i>IMA Optional TDMs External Loopback (IEL8 Command)</i> on page 80 | L |
| IIL | | Tests the IMA with internal loopback in framers | <i>IMA Internal Loopback (IIL Command)</i> on page 68 | L |
| IINT | | Tests the IMA chip interrupt signal | <i>IMA Interrupt (IINT Command)</i> on page 51 | L |
| INFO | | Displays general information on the board | <i>INFO Command</i> on page 33 | L |
| INFOEQU | IE | Displays board equipment information | <i>INFOEQU Command</i> on page 35 | L |
| INFOPCI | | Displays PCI information | <i>Equipment Information Display on a 4539-004 or 4539-005</i> on page 35 | L |
| IRST | | Tests the IMA chip reset signal | <i>IMA Reset (IRST Command)</i> on page 48 | L |
| ISRAM | | Tests the SRAM used by the IMA chip | <i>ISRAM Command</i> on page 81 | L |
| LED | | Test LEDs | <i>LEDs (LED Command)</i> on page 82 | L |

Table C-1. Monitor Commands (cont)

| Command | Short | Description | Reference | Loop |
|----------------|--------------|--|---|-------------|
| LOADF | LF | Flashes a program transferred through the Ethernet interface | <i>LOADF Command on page 103</i> | |
| LOADFC | LFC | Flashes a program transferred through a console | <i>LOADFC Command on page 94</i> | |
| LOADM | LM | Loads a program in memory transferred through the Ethernet Interface | <i>LOADM Command on page 103</i> | |
| LOCALIP | LIP | Displays/changes the board local IP Address | <i>Changing the Board IP Local Address (LOCALIP Command) on page 28</i> | |
| LOOP | | Executes the same command several times | <i>Running a Command Several Times (LOOP Command) on page 88</i> | |
| M60X | | Tests the 60x bus main SDRAM | <i>60x Bus Main Memory (M60X Command) on page 41</i> | |
| MAC | | Displays the board MAC address | <i>Displaying the Board Ethernet MAC Address (MAC Command) on page 28</i> | |
| MASKIP | MIP | Displays/Changes the board IP Subnet mask | <i>Changing the Board IP Subnet Mask (MASKIP Command) on page 28</i> | |
| MCAM | | Tests the CAM memories | <i>CAM (MCAM Command) on page 43</i> | L |
| MCAMRST | | Tests the CAM reset signals | <i>CAM Reset (MCAMRST Command) on page 49</i> | L |
| MDUMP | MD | Dumps a memory location | <i>Dumping a Memory Block (MDUMP Command) on page 105</i> | |
| MFILL | MF | Fills a memory block | <i>Filling a Memory Block (MFILL Command) on page 105</i> | |
| MFPROM | | Tests the FLASH EEPROM | <i>FLASH EEPROM (MFPROM Command) on page 45</i> | |
| MLOC | | Tests the local SDRAM | <i>Local Memory (MLOC Command) on page 43</i> | L |
| MMOVE | MM | Moves a memory block | <i>Moving a Memory Block (MMOVE Command) on page 105</i> | |
| MREAD | MR | Reads a memory location | <i>Reading a Memory Location (MREAD Command) on page 104</i> | |
| MSRPROM | | Tests the Serial EEPROM | <i>Serial EEPROM (MSRPROM Command) on page 44</i> | L |
| MWRITE | MW | Modifies a memory location | <i>Writing a Memory Location (MWRITE Command) on page 104</i> | |
| PARITY | P | Displays/Changes the TTY console parity | <i>Changing the Parity (PARITY Command) on page 26</i> | |
| PINT | | Tests the PowerSpan interrupt signal to the PowerQUICC II | <i>PCI Controller Interrupt (PINT Command) on page 52</i> | L |

Table C-1. Monitor Commands (cont)

| Command | Short | Description | Reference | Loop |
|----------------|--------------|---|---|-------------|
| POST | | Configures the Power-On Self Tests mode | <i>Power-On Self Test (POST) on page 15</i> | |
| PROD | | Runs a set of tests for product qualification | <i>Testing the Product (PROD Command) on page 85</i> | |
| RESET | RST | Resets the board | <i>Resetting the Board (RESET Command) on page 106</i> | |
| ROUTERIP | RIP | Displays/changes the router IP address | <i>Changing the Router IP Address (ROUTERIP Command) on page 29</i> | |
| S | | Stops the current command/test | <i>Stopping a Command (S Command) on page 88</i> | |
| SDUMP | SD | Dumps a memory block of Serial EEPROM | <i>Dumping a Serial EEPROM Block (SDUMP Command) on page 30</i> | |
| SERVERIP | SIP | Displays/changes the board server IP address | <i>Changing the Server IP Address (SERVERIP Command) on page 28</i> | |
| SOFTWARE | CFG | Displays the serial EEPROM software configuration | <i>Monitor Configuration (SOFTWARE Command) on page 23</i> | |
| SREAD | SR | Reads a memory location in Serial EEPROM | <i>Reading a Serial EEPROM Location (SREAD Command) on page 29</i> | |
| STOPBIT | SB | Displays/changes the TTY console stop bit number | <i>Changing the Stop Bits (STOPBIT Command) on page 26</i> | |
| SWRITE | SW | Modifies a memory location in the serial EEPROM | <i>Writing a Serial EEPROM Location (SWRITE Command) on page 30</i> | |
| T | | Displays tests category commands | <i>Test Category Commands on page 12</i> | |
| U | | Displays utility category commands | <i>Utility Category Commands on page 13</i> | |
| UEL | | Tests UTOPIA on P3 with external loopback | <i>UTOPIA External Loopback (UEL Command) on page 83</i> | L |

Introduction

The original Boot Firmware code `4539.ELF` is provided for restoration in the `BIN` directory of the CD-ROM.

The 4539 Boot Firmware source files are available in the `SRC\boot.zip` file of the CD-ROM. Developers can unzip the `boot.zip` file and refer to the source files for their own development and/or recompile them to generate their own Boot Firmware.

Files Description

[Table D-1](#) provides the Boot Firmware source file descriptions:

Table D-1. 4539 Boot Firmware Source File Descriptions

| File | Description |
|---------------------|---|
| APP | Directory that contains the 4539 Boot Firmware application (APP) level sources to handle menus, memory tests, PowerSpan tests, Monitor scheduling, and utility commands. |
| APP\ASM | Assembler startup code (STARTUP) directory. |
| APP\ASM\STARTUP.ASM | 4539 Boot Firmware startup code. This code is referred to as STARTUP in the <i>4539 Hardware Reference Manual</i> and performs system initialization as described in that manual. |
| APP\C | Directory that contains APP 'C' language source files. |
| APP\C\AMDFLASH.C | Routines to manage the 4539 FLASH EEPROM. |
| APP\C\BUFFER.C | Buffer object management. |
| APP\C\ELFFLASH.C | Routines to decode a ELF format – used for the <code>LOADFWC</code> command. |
| APP\C\FRAME.C | AMD FLASH management routines. |
| APP\C\LIST2LNK.C | Manages double-linked list objects. |
| APP\C\LOADFW.C | Handles the load of a firmware to memory. |
| APP\C\MAIN.C | 4539 Boot Firmware main initializations performed after system initializations (STARTUP). |
| APP\C\MEM.C | BIST memory tests (MLOC and M60X commands). |
| APP\C\MONABOOT.C | Handles the automatic Operational Firmware Ethernet-uploading process. |
| APP\C\MONARPC | Low-level routines to handle ARP and RARP protocols. Protocols handled are (at least partially): - ARP (RFC 826) - RARP (RFC 903) |

Table D-1. 4539 Boot Firmware Source File Descriptions (cont)

| File | Description |
|------------------|---|
| APP\C\MONBOOTP.C | Handles the BOOTP protocol (RFC 951). |
| APP\C\MONETHER.C | Handles the Ethernet protocol. |
| APP\C\MONIP.C | Handles the IP protocol stack: - IP (RFC 791) - UDP (RFC 768) - ICMP (RFC 792). |
| APP\C\MONMENU.C | Menus for a terminal session (TTY console or PCI console emulation) – tests supervision. |
| APP\C\MONPCI.C | PCI console emulation. |
| APP\C\MONSTRTU.C | Monitor start up configuration commands. |
| APP\C\MONTERM.C | Terminal session management such as TTY console or PCI console emulation. |
| APP\C\MONTFTP.C | Handles the TFTP protocol (RFC 783). |
| APP\C\MONTTY.C | TTY controller (SMC1) management. |
| APP\C\MONUTIL.C | Boot Firmware Monitor Utility commands (MMOVE , LOADFC , etc.). |
| APP\C\PSPANRW.C | PCI controller management (PowerSpan). |
| APP\C\SEEPROM.C | Low-level routines to handle the Serial EEPROM. |
| APP\C\XMODEM.C | XMODEM protocol used by LOAFC utility. |
| APP\H | Directory that contains APP header files. |
| APP\H\AMDFLASH.H | Header file of the APP\C\AMDFLASH.C module. |
| APP\H\APPVERS.H | Header file that contains Boot Firmware and APP versions. |
| APP\H\BUFFER.H | Header file of the APP\C\BUFFER.C module. |
| APP\H\ELF.H | ELF format definitions. |
| APP\H\ELFFLASH.H | Header file of the APP\C\ELFFLASH.C module. |
| APP\H\ELFTYPES.H | Types definitions used for elf records analyses. |
| APP\H\FRAME.H | Header file of the APP\C\FRAME.C module. |
| APP\H\HELPS.H | Includes all monitor on-line help header files in the APP\HLP directory |
| APP\H\LIBC.H | Standard function definitions. |
| APP\H\LIST2LNK.H | Header file of the APP\C\LIST2LNK.C module. |
| APP\H\LOADFW.H | Header file of the APP\C\LOADFW.C module. |
| APP\H\MEM.H | Header file of the APP\C\MEM.C module. |
| APP\H\MONABOOT.H | Header file of the APP\C\MONABOOT.C module. |
| APP\H\MONARP.H | Header file of the APP\C\MONARP.C module |
| APP\H\MONBOOTP.H | Header file of the APP\C\MONBOOTP.C module. |
| APP\H\MONETHER.H | Header file of the APP\C\MONETHER.C module. |

Table D-1. 4539 Boot Firmware Source File Descriptions (cont)

| File | Description |
|---------------------|---|
| APP\H\MONIP.H | Header file of the APP\C\MONIP.C module. |
| APP\H\MONMENU.H | Header file of the APP\C\MONMENU.C module. |
| APP\H\MONPCI.H | Header file of the APP\C\MONPCI.C module. |
| APP\H\MONSTRTU.H | Header file of the APP\C\MONSTRTU.C module. |
| APP\H\MONTERM.H | Header file of the APP\C\MONTERM.C module. |
| APP\H\MONTFTP.H | Header file of the APP\C\MONTFTP.C module. |
| APP\H\MONTTY.H | Header file of the APP\C\MONTTY.C module. |
| APP\H\MONUTIL.H | Header file of the APP\C\MONUTIL.C module. |
| APP\H\PSPANRW.H | Header file of the APP\C\PSPANRW.C module. |
| APP\H\SEEPROM.H | Header file of the APP\C\SEEPROM.C module. |
| APP\H\XMODEM.H | Header file of the APP\C\XMODEM.C module. |
| APP\HLP | Directory that contains monitor commands on-line help files. |
| APP\INC | Header files of the STARTUP module (APP\ASM\STARTUP.ASM). |
| APP\INC\STARTUP.INC | Main header file of the APP\ASM\STARTUP.ASM module. |
| APP\INC\UPM_CAM.INC | UPM table values for the APP\ASM\STARTUP.ASM module. |
| APP\OUT | Directory containing APP compilation output |
| APP\OUT\BIN.DBG | Result (ELF and MAP files) of a DEBUG compilation. |
| APP\OUT\BIN.PRD | Result (ELF and MAP files) of a PRODUCT compilation. |
| APP\OUT\ERR.DBG | APP error files of a DEBUG compilation. |
| APP\OUT\ERR.PRD | APP error files of a PRODUCT compilation. |
| APP\OUT\OBJ.DBG | APP object files of a DEBUG compilation. |
| APP\OUT\OBJ.PRD | APP object files of a PRODUCT compilation. |
| ETH | Low-level routines to handle Ethernet interface tests. |
| ETH\ASM | Empty directory. |
| ETH\C | Directory that contains ETH 'C' language source files. |
| ETH\C\FECIO.C | Level routines to handle MPC8260 parallel port pins reserved to Ethernet chip interface (LXT971). |
| ETH\C\LXTINIT.C | Routines to initialize the Ethernet chip interface (LXT971). |
| ETH\C\FECINIT.C | Routines to initialize the Ethernet framer (FCC3). |
| ETH\C\ETH.C | Routines to handle Ethernet tests. |
| ETH\C\FECINT.C | Routines to handle FCC3 interrupt. |
| ETH\H | Directory that contains ETH header files. |
| ETH\H\ETH.H | Header file of the ETH\H\ETH.C module. |
| ETH\H\ETHVERS.H | Header file that contains ETH version. |

Table D-1. 4539 Boot Firmware Source File Descriptions (cont)

| File | Description |
|------------------|--|
| ETH\H\FECINIT.H | Header file of the ETH\H\FECINIT.C module. |
| ETH\H\FECINT.H | Header file of the ETH\H\FECINT.C module. |
| ETH\H\FECIO.H | Header file of the ETH\H\FECIO.C module. |
| ETH\H\LXTINIT.H | Header file of the ETH\H\LXTINIT.C module. |
| ETH\INC | Empty directory. |
| ETH\OUT\ERR.DBG | ETH error files of a DEBUG compilation. |
| ETH\OUT\ERR.PRD | ETH error files of a PRODUCT compilation. |
| ETH\OUT\OBJ.DBG | ETH object files of a DEBUG compilation. |
| ETH\OUT\OBJ.PRD | ETH object files of a PRODUCT compilation. |
| STD | Directory that contains print utilities. |
| STD\ASM | Empty directory. |
| STD\C | Directory that contains STD 'C' language source files. |
| STD\C\PRINT.C | Print routines. |
| STD\C\STRUTIL.C | Defines some string utilities. |
| STD\H | Directory that contains STD header files. |
| STD\H\STDIO.H | Contains common standard I/O structures, constants, and prototype definitions. |
| STD\H\STDVERS.H | Header file that contains STD version. |
| STD\H\STRUTIL.H | Header file of the STD\C\STRUTIL.C module. |
| STD\INC | Empty directory. |
| STD\OUT\ERR.DBG | STD error files of a DEBUG compilation. |
| STD\OUT\ERR.PRD | STD error files of a PRODUCT compilation. |
| STD\OUT\OBJ.DBG | STD object files of a DEBUG compilation. |
| STD\OUT\OBJ.PRD | STD object files of a PRODUCT compilation. |
| SYS | Directory that contains system routines. |
| SYS\ASM | Empty directory. |
| SYS\C | Directory that contains SYS 'C' language source files. |
| SYS\C\HCFG.C | 4539 hardware configuration management. |
| SYS\C\INTERRUP.C | Low-level interrupt routines. |
| SYS\C\MALLOC.C | Routines to handle dynamic memory allocation. |
| SYS\C\PARPORTS.C | Parallel port initialization and peripheral reset routines. |
| SYS\C\TIME.C | Time routines. |
| SYS\H | Directory that contains SYS header files. |

Table D-1. 4539 Boot Firmware Source File Descriptions (cont)

| File | Description |
|------------------|--|
| SYS\H\4539.H | Contains PowerQUICC II parallel port pin assignments and chip address definitions. |
| SYS\H\HCFG.H | Header file of the SYS\C\HCFG.C module. |
| SYS\H\MALLOC.H | Header file of the SYS\C\MALLOC.C module. |
| SYS\H\PARPORTS.H | Chips reset/un-reset routine prototypes. |
| SYS\H\PQII.H | Power QUICC II definitions. |
| SYS\H\PSPDEF.H | PowerSpan definitions. |
| SYS\H\QFALC55.H | Contains QFALC register definition. |
| SYS\H\STD.H | Header file that contains common definitions. |
| SYS\H\SYSVERS.H | Header file that contains SYS version. |
| SYS\H\TIME.H | Header file of the SYS\C\TIME.C module. |
| SYS\H\TIMER.H | Header file of the SYS\C\TIME.C module. |
| SYS\INC | Empty directory. |
| SYS\OUT\ERR.DBG | SYS error files of a DEBUG compilation. |
| SYS\OUT\ERR.PRD | SYS error files of a PRODUCT compilation. |
| SYS\OUT\OBJ.DBG | SYS object files of a DEBUG compilation. |
| SYS\OUT\OBJ.PRD | SYS object files of a PRODUCT compilation. |
| TST | Low-level routines to handle QuadFALC, CAM, IMA tests. |
| TST\ASM | Empty directory. |
| TST\C | Directory that contains TST 'C' language source files. |
| TST\C\CAM.C | CAM routines. |
| TST\C\IMA.C | IMA chip initialization and test routines. |
| TST\C\PQATM.C | ATM loopback test management. |
| TST\C\PQTDM.C | HDLC over MCCs loopback test management. |
| TST\C\QFALC.C | QuadFALCs initialization and test routines. |
| TST\H\CAM.H | Header file of the TST\C\CAM.C module. |
| TST\H\IMA.H | Header file of the TST\C\IMA.C module. |
| TST\H\MCC.H | MCC constants. |
| TST\H\PQATM.H | Header file of the TST\C\PQATM.C module. |
| TST\H\PQTDM.H | Header file of the TST\C\PQTDM.C module. |
| TST\H\TESTS.H | Constants to handle tests. |
| TST\H\TSTVERS.H | Header file that contains TST version. |
| TST\INC | Empty directory. |
| TST\OUT\ERR.DBG | TST error files of a DEBUG compilation. |

Table D-1. 4539 Boot Firmware Source File Descriptions (cont)

| File | Description |
|-----------------|--|
| TST\OUT\ERR.PRD | TST error files of a PRODUCT compilation. |
| TST\OUT\OBJ.DBG | TST object files of a DEBUG compilation. |
| TST\OUT\OBJ.PRD | TST object files of a PRODUCT compilation. |
| 4539DBG.LNK | File that contains link directives for a debug generation of the 4539 Boot Firmware. This file is invoked from the GEN4539.BAT file. |
| 4539PRD.LNK | File that contains link directives for a product generation of the 4539 Boot Firmware. This file is invoked from the GEN4539.BAT file. |
| GEN4539.BAT | DOS batch file to generate the 4539 Boot Firmware. This batch relies on the DIAB 3.7A compiler and must be modified by the user to adapt to its own development environment. |

Recompiling the 4539 Boot Firmware

To recompile the 4539 Boot Firmware, run 'GEN4539.BAT' batch file.

The procedure to recompile the 4539 Boot Firmware assumes that the development environment is DOS\DIAB3.7 and that the compiler is in a G:\DEVEL\DIAB3.7A directory. For a different environment, developers should edit GEN4539.BAT, 4539DBG.LNK, and 4539PRD.LNK files and adapt them.

Re-Flashing Code into the 4539

E

Overview

Before flashing code into the 4539, you must distinguish between the two types of embedded firmware:

- All 4539 boards are provided with an Interphase-supplied boot code, called Boot Firmware. This Boot Firmware is already flashed on each communications controller. The original Boot Firmware is provided for restoration in the `bin` directory of the Board Development Kit CD-ROM shipped with the Interphase 4539.
- On Interphase boards, you can also flash Interphase embedded firmware or your own firmware code. This firmware code is referred to as Operational Firmware.



WARNING

You must know which space in the FLASH EEPROM is reserved for the Boot Firmware and which area is available for the user Operational Firmware.

If the Operational Firmware erases the Boot Firmware, the 4639 board may not boot up. We recommend the board be returned to Interphase RMA for repair.

Re-flashing may be performed in the field by a qualified technician using one of the methods in this appendix.

Flash EEPROM Memory Mapping

The Flash EEPROM is divided into 64-KByte blocks, some of which are reserved for Interphase Boot Firmware use, others are available to the user.

The following blocks are reserved and are not available to the user:

- The first block (offset 0 to 0xFFFF) of the Flash; this block contains the hardware configuration words of the PowerQUICC II.
- The last Mbyte is reserved for Interphase Boot Firmware code. Since the Boot Firmware code size is less than 1Mbyte, some free space may be available for the user at the end of the last Mbyte.

Figure E-1 shows the Flash EEPROM mapping for a 4 MByte memory and shows a Boot Firmware code size of 0xF0000.

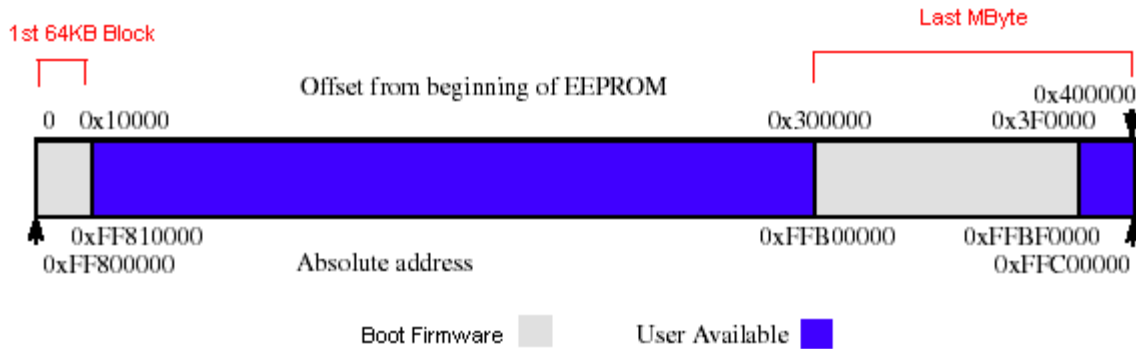


Figure E-1. Flash EEPROM Mapping

Methods for Flashing Firmware into the Board

The user can flash any firmware (Boot Firmware or Operational Firmware) using one of the methods defined below:

- **Through the TTY:** Interphase Boot Firmware provides an utility command, `LOADFC`, that allows the user to flash any firmware onto the board using a TTY Terminal connected to the TTY board connector. Default parameters of the TTY Connection are 9600 baud, 8 data bits, 1 stop bit, no parity and no flow control.
- **Through the PCI Bus:** In chassis systems equipped with a PCI System Board (such as the MCP750 or the CP1500) running VxWorks or Solaris OS, the user can flash the Interphase board using a PCI flash utility running on the System Board. Interphase provides tools for VxWorks and Solaris environments in the tools directory of the BDK CD-ROM.

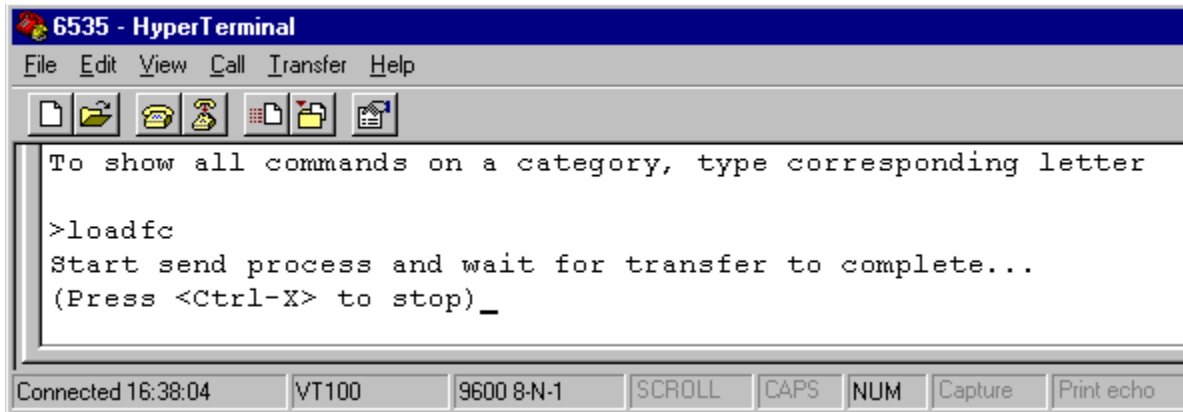
Flashing the Boot Firmware

Since the Boot Firmware is already flashed by Interphase prior to shipping the board, the user must remember that a specific FLASH EEPROM space is reserved for the Boot firmware. The Flash address location where to store the Boot Firmware is already defined inside the Boot Firmware binary.

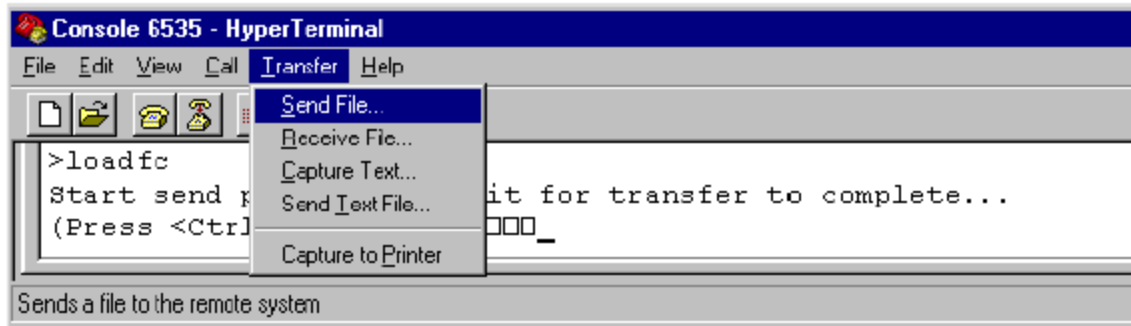
Flashing Through the TTY Port

To access the Boot Firmware and the `LOADFC` command, the user must connect a TTY cable between the TTY console and the board TTY port, using the default parameters specified above. As soon as the Boot Firmware Monitor is accessed, the user must do the following steps to flash the Boot Firmware. The following example is for a Windows Hyper Terminal connected to a 6535 board but it can be applied for any other TTY console emulator and for any Interphase board:

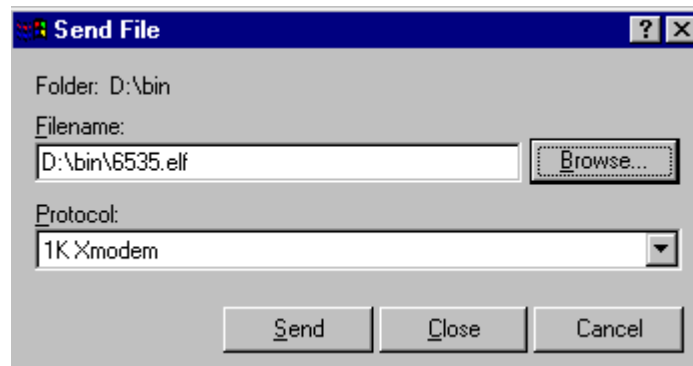
1. Enter `LOADFC` on the Boot Firmware Monitor:



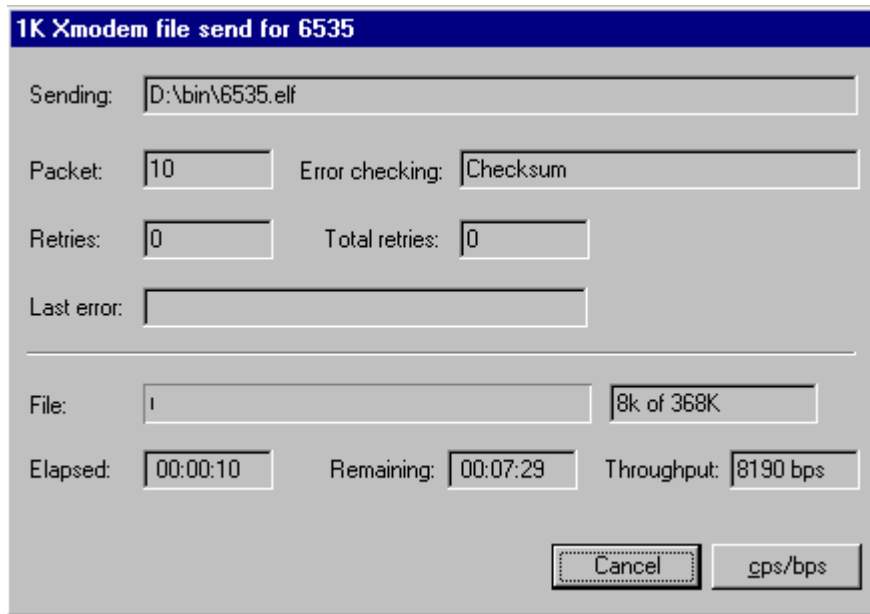
2. To send the Boot Firmware binary (`xxxx.elf`) to FLASH select the **Transfer** and the **Send File** options:



3. Select the Boot Firmware binary to transfer, the protocol (1K Xmodem or Xmodem) and press the **Send** button:



The transfer progress is displayed in another window:



4. At the end of the transfer, confirm the Flash process. A warning is displayed to inform user that he will overwrite the previous Boot Firmware code:

```

>lfc
Start send process and wait for transfer to complete...
(Press <Ctrl-X> to stop)[][]
Receive status=[
Bytes sent: 255 received: 257J01
Frames received: 250 Data bytes count: 256000
NAK sent: 0 Bad FCS: 0 Invalid header: 0 Ignored frames: 0
Timeout: 4
Transfer succeeded.
File sections will be flashed from address FC000000h
Please confirm beginning of flash process [Y|N]: y
WARNING: At least 5 section(s) will be flashed over present memory
Are you sure to start the flash process [Y|N]: y
Flashing section #1 at FC000010h (32 bytes): . OK
Flashing section #2 at FFF00110h (12032 bytes): .. OK
Flashing section #3 at FFF03010h (5576 bytes): . OK
Flashing section #4 at FFF045C8h (512 bytes): . OK
Flashing section #5 at FFF047C8h (1520 bytes): . OK
Flashing section #7 at FFF04DB8h (149552 bytes): ..... OK
Flashing section #8 at FFF295E8h (50264 bytes): ..... OK
Flashing section #9 at FFF35A40h (160 bytes): . OK
Press any key to reset the adapter...

```

5. Press any key to restart the board. The communication controller will reboot from the new flashed Boot Firmware.

Flashing Through the PCI bus

The PCI flashing process is allowed only if the chassis has a System Board (an MCP750 or CP1500 for example). Interphase provides tools that will be loaded on this System Board. The System Board must be running VxWorks or Solaris OS. The PCI Flash utility is `iphSetup`.

Under VxWorks

First, the user needs to install and build the tools before using `iphSetup`:

1. Create an `iphSetupTools` directory in your Tornado project directory.
2. Copy the VxWorks Maintenance tools located in the `\tools\vxworks` directory of the BDK CD-ROM to the `iphSetupTools` directory (for example: `c:\tornado\proj\iphSetupTools`).

3. Open `iphSetupTools` project (`target/proj/ iphSetupTools.wpj`). Verify that the build properties are according to the selected system board in the `board.c` file. Update dependencies and rebuild the project.
4. Open a VxWorks target shell on the System board (MCP750 for example).
Enter `ld < target/proj/iphSetupTools/out/iphSetupTools.out` on the target shell to load the tools.
The user can now flash the Boot Firmware on the Interphase board through the VxWorks target shell already connected onto the system board:
5. Display the list of communication controllers present in the chassis by entering `iphShowDev` on the target shell. Following example shows the displayed result. Note the index number of the board to flash. This index number is used by the `iphSetup` flash tool:

```
-> iphShowDev
Board(s) found with Interphase vendor ID (0x107E):
-----
Index Bus Slot  Function DevId  SerialNum Card Name
-----
0      01  08      00  0x9060  1540120  6535 cPCI 4P/8P T1/E1/J1
1      02  04      00  0x90A0  1428129  4539 PMC 4P T1/E1/J1

value = 1 = 0x1
->
```



NOTE

The following is the syntax and usage for the flash utility:

Syntax:

```
iphSetup (deviceIndex, filename, flashLoadOffset, compare)
```

Usage:

- `deviceIndex` is the index number displayed by the `iphShowDev` tool.
- `filename` is the Boot Firmware name (`xxxx.elf`) with the complete path of the file to be flashed. Do not forget to copy Boot Firmware from the BDK CD-ROM into a directory of your VxWorks environment (for example `c:/tornado`).
- `flashLoadOffset` is the FLASH offset where the code will be flashed. Not significant for the Boot Firmware since offset already defined inside binary.
- `compare` is a Boolean (0 or 1). If it is set to 0, the specified file is flashed on the board. If it is set to 1, the file content is compared with the memory contents.

6. Enter the following command on the target shell connected to the System board.
Only `deviceIndex` and `filename` change because they are associated to a specific

board. The following example shows how to flash a board with `deviceIndex=0` and with a Boot Firmware named `bootfirm.elf`:

```
-> iphSetup(0, "bootfirm.elf", 0, 0)

-----
          Setup tool for Interphase Communications Controllers
          Version 1.17
          Copyright (c) 2001 - Interphase Corp.
-----

PCI regs base address : 0xC1000000
Memory base address : 0xC1200000
DeviceId = 0x9060
SerialNum = 1540120

Program FLASH :
Erasing FLASH : sectors to erase = 0. 48. 49. 50. 51.
  You are going to load in boot firmware area and the current boot firmware
  may be lost
  The board will not boot again, if the file bootfirm.elf is not a correct
  boot firmware
  Do you really want to erase these flash sectors ('y' or 'n')?
  ....
  Erase ok
  Loading flash : successfully programmed

Revisions :
  Card Date       : 06/03/2001
  PSPAN EEPROM   : SX556A00
  EPLD (1rst)    : SX557A00
  Boot Firmware  : SX558A00
value = 0 = 0x0
->
```



WARNING

While flashing any code on Boot Firmware Flash reserved space, a warning message is displayed. A user confirmation is required.

Under Solaris

The system must be equipped with a PCI Master system board running Solaris 7 or Solaris 8.

1. Copy the Solaris package file `iphdrvtoo.tar` from the `/tools/Solaris` directory of the CD-ROM to the target Solaris machine. Then untar the package file by executing the following command: `tar xvf iphdrvtoo.tar`
2. To install this software package, use the `pkgadd` command or the software installation program `Admintool`. For both programs, you have to specify the complete path name where the package can be found, that is the directory name where you copied the package file to your local disk. The following line gives an

example of use of the pkgadd command:

```
pkgadd -d <pathname> iphdrvtoo
```

Where <pathname> is the full path name of the directory where the package is located. The package installation program copies all the component files to their destination directory and dynamically loads the device driver. It does not modify any system files. Then the PCI Flash utility is located in the **/opt/iphdrvtoo/tools** directory.



NOTE

The following is the syntax and usage for the flash utility:

Syntax:

```
iphSetup devicenode [options] [filename]
```

Usage:

- **devicenode** is the name of the device node used to access the card to be configured (format `/dev/iph_wan_x` where `x` represents a card index).
- **option:** Offset to specify the offset from the beginning of the memory space, where to load the code. It is an integer used for flash memory update. It can be in hexadecimal (`0x0123ABCD`) or decimal format (`12345`).
- **filename** is the name of the file to load. If not specified, the `iphSetup` command displays information about the boot firmware that is currently flashed.

Accepted file extensions by iphSetup:

- **.boo** or **.elf** is for Boot Firmware or WAN Firmware to load in a device's Flash EEPROM. It is a binary ELF format file.
- **.bin** is for a non-ELF binary code to load in a device's Flash EEPROM.

-
3. To flash the Interphase Boot Firmware, copy it first to the target Solaris machine. Then execute following command:

```
iphSetup /dev/iph_wan_0 firmware.elf
```

You will be asked “Do you really want to erase these flash sectors? (‘y - ‘n’):**y**”

This will store the Boot Firmware file binary (`firmware.elf`) in the Flash EEPROM of the card identified by the `/dev/iph_wan_0` device node. You do not need to specify the offset where to flash the code since `iphSetup` gets the address directly from the ELF file.

Upon successful completion you will see the following information.

```
Board infos:  
Serial Number: 1234567  
Revisions:  
Card Date: dd/mm/yy  
PSPAN EEPROM: SXnnnnnn
```

```
ELPD (1rst): SXnnnnnn  
Boot Firmware: dd/mm/yy
```



WARNING

While flashing any code on Boot Firmware Flash reserved space, a warning message is displayed. A user confirmation is required.

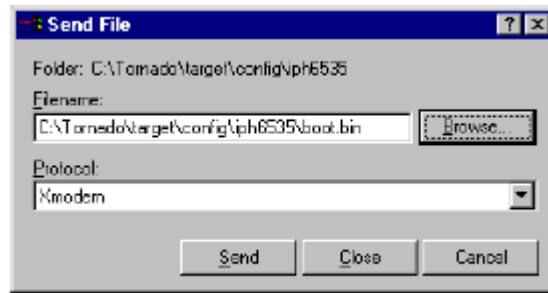
Flashing the Operational Firmware

Before flashing the Operational Firmware on the Flash Memory, the user must check that its code location on the Flash **doesn't erase** any piece of code of the Boot Firmware. If Boot Firmware is erased the board may not boot again and may need to be sent to Interphase Manufacturing.

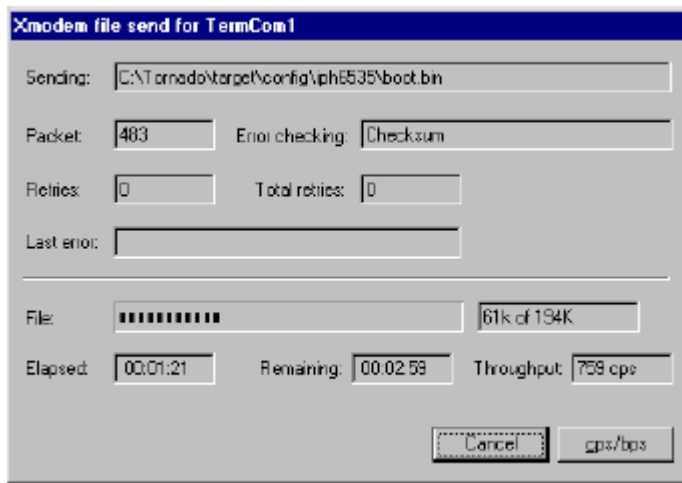
Flashing Through the TTY Port

To access the Boot Firmware, and then the `LOADFC` command, user must connect a TTY cable between the TTY console and the board TTY port, using the default parameters shown on page 128. As soon as the Boot Firmware Monitor is accessed, do the following steps to flash the Operational Firmware. The following example is for a Windows Hyper Terminal connected to a 6535 board but it can be applied for any other TTY console emulator and for any Interphase board:

1. Enter `LOADFC -BIN FF81000` on the Boot Firmware Monitor, assuming that address `0xFF81000` is the starting address location in Flash memory of the Operational Firmware. Check that Operational Firmware starting address + size of Operational Firmware does not overwrite any Boot Firmware area.
2. To send the Operational Firmware binary (`xxxx.bin` or any binary format) to flash, select the `Transfer` and the `Send File` options as described in the Flashing Boot Firmware Section on page 129.
3. Select the Boot Firmware binary to transfer, the protocol (1K Xmodem or Xmodem) and press the **Send** button:



The progression of the transfer is displayed in another window:



- At the end of the transfer, confirm the Flash process. A warning will be displayed to inform the user if he is overwriting the Boot Firmware code:

```
File sections will be flashed to range [FF810000h..FF875480h]
Please confirm beginning of flash process [Y|N] :
    - Confirm the overwrite, the Boot Firmware displays the following:
..... OK
Firmware successfully flashed
```

Flashing Through the PCI bus

The PCI flashing process is allowed only if customer has got a System Board on his chassis (MCP750 or CP1500 for example). Interphase provides tools that will be loaded on this System Board. The System Board must be running VxWorks or Solaris OS. The PCI Flash Utility is `iphSetup`.

Under VxWorks

1. Create an `iphSetupTools` directory in your Tornado project directory.
2. Copy the VxWorks Maintenance tools located in the `\tools\vxworks` directory of the BDK CD-ROM to the `iphSetupTools` directory (for example: `c:\tornado\proj \iphSetupTools`).
3. Open `iphSetupTools` project (`target/proj/ iphSetupTools.wpj`). Check that the build properties are according to the selected system board in the `board.c` file. Update dependencies and rebuild the project.
4. Open a VxWorks target shell on the System board (MCP750 for example).
Enter `ld < target/proj/iphSetupTools/out/iphSetupTools.out` on the target shell to load the tools.
5. Display the list of communication controllers present in the chassis machine by entering `iphShowDev` on the target shell. Following example shows the displayed result. Note the index number of the board to flash. This index number is used by `iphSetup` flash tool:

```
-> iphShowDev
Board(s) found with Interphase vendor ID (0x107E):
-----
Index Bus Slot  Function DevId  SerialNum Card Name
-----
0      01  08      00  0x9060  1540120  6535 cPCI 4P/8P T1/E1/J1
1      02  04      00  0x90A0  1428129  4539 PMC 4P T1/E1/J1

value = 1 = 0x1
->
```



NOTE

The following is the syntax and usage for the flash utility

Syntax:

```
iphSetup (deviceIndex, filename, flashLoadOffset, compare)
```

Usage:

- `deviceIndex` is the index number displayed by `iphShowDev` tool.
- `filename` is the Operational Firmware name (`xxxx.boo`) with complete path of the file to be flashed. Do not forget to copy the Operational Firmware into a directory of your VxWorks environment (for example `c:/tornado`).
- `flashLoadOffset` is the FLASH offset where the code will be flashed. It can be expressed in hexadecimal (`0x125ABC`) or decimal (`12345`). `iphSetup` will ignore the most significant bits, based on the Flash EEPROM size (for example, on a 4 MB Flash EEPROM, only the 22 least significant bits are taken into account, i.e. mask `0x003FFFFFF`). For Operational Codes (`.elf` or `.boo` extension files), the effective address is computed by adding this offset to the address in the ELF format file. For a binary data file (`.bin` extension file), `flashLoadOffset` is considered as an absolute address in the flash memory.
- `compare` is a Boolean (0 or 1). If it is set to 0, the specified file is flashed on the board. If it is set to 1, the file content is compared with the memory contents.

6. Enter following command on the target shell connected to the System board.
`deviceIndex` depends on the board.

The following example shows how to flash Operational firmware named `OpFirmw.boo` at offset `0x10000` (equivalent to `0xFF810000` Flash address) on the board with `deviceIndex=0` and with a Boot Firmware named `bootfirm.elf`:

```
-> iphSetup(0, "OpFirm.boo", 0x10000, 0)
```

```
-----
Setup tool for Interphase Communications Controllers
Version 1.17
Copyright (c) 2001 - Interphase Corp.
-----
PCI regs base address : 0xC1000000
Memory base address : 0xC1200000
DeviceId = 0x9060
SerialNum = 1540120
Program FLASH :
```

```
Erasing FLASH : sectors to erase = 1. 2. 3. 4. 5. 6. ....
Erase ok
Loading flash : successfully programmed
Revisions :
Card Date : 06/03/2001
PSPAN EEPROM : SX556A00
EPLD (1rst) : SX557A00
Boot Firmware : SX558A00
value = 0 = 0x0
->
```



WARNING

While flashing any code on Boot Firmware Flash reserved space, a warning message is displayed. A user confirmation is proposed.

Under Solaris

The system must be equipped with a PCI Master system board running Solaris 7 or Solaris 8.

1. Copy the Solaris package file `iphdrvtoo.tar` from the `/tools/Solaris` directory of the CD-ROM to the target Solaris machine. Then untar the package file by executing the following command: `tar xvf iphdrvtoo.tar`
2. To install this software package, use the `pkgadd` command or the software installation program `Admintool`. For both programs, you have to specify the complete path name where the package can be found, that is the directory name where you copied the package file to your local disk. The following line gives an example of use of the `pkgadd` command:

```
pkgadd -d <pathname> iphdrvtoo
```

Where `<pathname>` is the full path name of the directory where the package is located. The package installation program copies all the component files to their destination directory and dynamically loads the device driver. It does not modify any system files. Then the PCI Flash utility is located in the `/opt/iphdrvtoo/tools` directory.



NOTE

The following is the syntax and usage for the flash utility

Syntax:

```
iphSetup devicenode [options] [filename]
```

Usage:

- **devicenode** is the name of the device node used to access the card to be configured (format `/dev/iph_wan_x` where `x` represents a card index).
- **options:** Offset to specify the offset from the beginning of the memory space, where to load the code. It is an integer used for flash memory update. It can be in hexadecimal (`0x0123ABCD`) or decimal format (`12345`).
- **filename** is the name of the file to load. If not specified, the `iphSetup` command displays information about the boot firmware that is currently flashed.

Accepted file extensions by `iphSetup`:

- **.boo** or **.elf** is for Boot Firmware or WAN Firmware to load in a device's Flash EEPROM. It is a binary ELF format file.
- **.bin** is for a non-ELF binary code to load in a device's Flash EEPROM.

-
3. Copy the Operational Firmware to the target Solaris machine. Then execute following command:

```
iphSetup /dev/iph_wan_0 -o 0xFF8B0000 OpFirm.bin
```

This command will store the Operational binary code contained in the `.bin` binary file `OpFirm.bin` in the Flash EEPROM of the card identified by the `/dev/iph_wan_0` device node at offset `0xFF8B0000`.



WARNING

While flashing any code on Boot Firmware Flash reserved space, a warning message is displayed. A user confirmation is required.

Overview

Before running the firmware loading process, one or several hosts must be configured on the local network in order to reply to the requests sent by the board.

The following servers have to be configured on the local network:

- At least one RARP or BOOTP server
- At least one TFTP server

Following is a non-exhaustive list of different system configurations to apply in order to successfully load a firmware through the Ethernet.

Solaris Operating System

RARP

The RARP server is implemented by the `/usr/sbin/in.rarpd` daemon (see `man in.rarpd`).

Configuration

Logged as root, the network administrator has to create (or edit) the text file `/etc/ethers` which contains the address association database. Each line defines the link between a Ethernet/MAC address and an IP address. For example:

```
# /etc/ethers
# That is a comment
00:00:77:95:91:bd 157.175.33.170
```

Defines IP address 157.175.33.170 of an equipment connected on the local network whose Ethernet interface has the address 00:00:77:95:91:bd.

Moreover, the IP address can be superseded by a host name defined in `/etc/hosts`. For example:

```
# /etc/ethers
# That is a comment
00:00:77:95:91:bd Interphase4539Board1
```

```
# /etc/hosts
Interphase6535Board1 157.175.33.170
```

Execution

The RARP server is started by executing (as root) the `/usr/sbin/in.rarpd` daemon:

```
% /usr/sbin/in.rarpd -a
```

Use the `kill` command to stop the RARP daemon:

```
% ps -ef | grep rarpd
root 242 1 0 12:59:45 ? 0:00 /usr/sbin/in.rarpd -a
% kill 242
```

Option `-a` tells the RARP server to poll all the network interfaces of the host.

Option `-d` runs the RARP server as a 'normal' process (not as a daemon) in a “debug” mode (all RARP requests and replies received and/or sent are displayed on its control terminal). User has to SIGINT the process to stop it (generally `<Ctrl-C>`).

BOOTP

The BOOTP server is implemented by the DHCP server `/usr/lib/inet/in.dhcpd` (see `man in.dhcpd`).

BOOTP can be configured with the `dhcpconfig` shell-script by following the steps described below:



WARNING

The BOOTP configuration steps described below allow configuring a BOOTP server in such a way that a board can retrieve its IP address from this server ONLY IN THE CONTEXT OF A TEST. You should check with your network administrator if you want to integrate the BOOTP configuration into an existing DHCP configuration.

1. Unconfigure DHCP or Relay Service and verify that all the warning prompts display.
2. Configure DHCP Service, then answer `yes` to stop the DHCP service.
3. Enter the datastore type (`files` for example) and keep the proposed directory (`/var/dhcp`).

4. Answer **Yes** to specify non-default daemon options (default is `BOOTP not active`).
5. Keep the proposed values of the different times the script offers next.
6. Answer **Yes** to enable BOOTP compatibility.
7. Answer **Yes** to enable the server to allocate IP addresses to new BOOTP clients.
8. Keep the proposed default DHCP lease policy (in days).
9. Answer **No** to allow clients to renegotiate their leases.
10. Answer **Yes** to enable DHCP/BOOTP support of networks you select.
11. Answer **Yes** to configure BOOTP/DHCP on local LAN network.
12. Answer **No** (BOOTP doesn't matter) to generate and insert host names in the hosts table.
13. Keep the proposed starting IP address.
14. Enter the number of clients you want to add, and then answer **Yes** to specify any of the addresses to be BOOTP specific and finally, enter the number of BOOTP specific addresses.
15. Answer **Yes** to disable ping verification (at this moment, the board does not handle ICMP).
16. Answer **No** to not configure BOOTP/DHCP services on remote networks.
17. Answer **Yes** to restart DHCP services.
18. Finally, choose `Exit` from the main menu to quit the configuration.

You can verify if the DHCP daemon is effectively running by executing:

```
% ps -ef | grep dhcpd
root  930  1  0 15:18:34 ?        0:00 /usr/lib/inet/in.dhcpd -b automatic
```

The script has created the configuration files `/var/dhcp/dhcptab` and `/var/dhcp/n1_n2_n3_n4` (`n1`, `n2`, `n3`, and `n4` are the decimal values of the four bytes of the starting IP address). See `'man dhcptab'` and `'man dhcp_network'` for more details about their content.

TFTP

The TFTP server is implemented by the `/usr/sbin/in.tftpd` daemon (see `man in.tftpd`). This daemon is started (or stopped) as a service by the super daemon `inetd` (see `man inetd`).

Configuration

The TFTP server is configured by the text file `/etc/inetd.conf` (see `'man inetd.conf'`).

Search that file for the line that describes the TFTP service (it begins with `tftpd`). If the line is present, check if it is commented (by a `#`) or not. If it is commented, delete the `#`.

If the TFTP service is not present, add the following line:

```
# /etc/inetd.conf
tftpd  dgram  udp  wait  root  /usr/sbin/in.tftpd  in.tftpd -s /tftpboot
```

Last column of the line describes the arguments of the daemon as it is started. One of these argument is the name of the directory of the sent and/or received files (default directory is `/tftpboot`).

Change this directory if required.

Execution

Sending the `SIGHUP` signal to the `inetd` daemon request it to read the file `/etc/inetd.conf` again:

```
% ps -ef | grep inetd
root 159 1 0 12:59:34 ?    0:00 /usr/sbin/inetd -s
% kill -HUP 159
```

Red Hat Linux Release 6.x

RARP

For this release, no RARP server is available and/or was found.

BOOTP

Configuration



NOTE

The package `dhcp-2.0xxx.rpm` must be installed (this package is not installed in some configurations). See the RedHat CD-ROM (`/mnt/cdrom/RedHat/RPMS`) for an example).

The BOOTP server is implemented by the DHCP server `/usr/sbin/dhcpd` (see `man dhcpd`).

BOOTP is configured by the text file `/etc/dhcpd.conf`. An example of such a file follows:



WARNING

The BOOTP configuration steps described below allow configuring a BOOTP server in such a way that a board can retrieve its IP address from this server **ONLY IN THE CONTEXT OF A TEST**. You should check with your network administrator if you want to integrate the BOOTP configuration into an existing DHCP configuration.

```
# /etc/dhcpd.conf
#
# DESCRIPTION
#
#     DHCP/BOOTP database for Linux RedHat 6.x
#
# USAGE
#
#     - 'dhcp-2.0xxx.rpm' must be installed from RedHat Linux 6.x CD-ROM
#       (/mnt/cdrom/RedHat/RPMS)
#     - % touch /var/state/dhcp/dhcp.leases
#     - /etc/rc.d/init.d/dhcpd start
#
# AUTHOR
#
#     Stephane Grosjean
#
# DATE
#
#     01/19/2001
deny unknown-clients;
allow bootp;

subnet 157.175.33.0 netmask 255.255.255.0 {
    host iph4539Adapter {
        hardware ethernet 00:00:77:95:CA:A2;    # MAC address
        fixed-address 157.175.33.170;          # Fixed IP address
# if 'filename' below not present, adapter will try to load
# '9DAF21AA' as configuration filename (built from IP address above)
        filename "bootp4539.cfg";             # Configuration file
    }
}
```

This configuration file defines the following:

- BOOTP is allowed.

- A board named `iph4539Adapter` whose Ethernet interface address is `00:00:77:95:CA:A2`.
- `157.175.33.170` is the fixed IP address of the board.
- The name of a file that will be interpreted by the board as the name of the configuration file to download.

The DHCP/BOOTP server needs the presence of text file `/var/state/dhcp/dhcp.leases` to run correctly. If it doesn't exist, touch it in order to create it:

```
% touch /var/state/dhcp/dhcp.leases
```

Execution

Once the `/etc/dhcpd.conf` has been modified, the DHCP/BOOTP daemon must be stopped and restarted in order to read the modifications:

```
% /etc/rc.d/init.d/dhcpd stop  
% /etc/rc.d/init.d/dhcpd start
```

TFTP

The TFTP server is implemented by the `/usr/sbin/in.tftpd` daemon (see `man in.tftpd`). This daemon is started (or stopped) as a service by the super daemon `inetd` run at boot time by `/etc/rc.local` (see `man inetd`).

Configuration

The TFTP server is configured by the text file `/etc/inetd.conf` (see `man inetd.conf`).

Search that file for the line that describes the TFTP service (it begins with `tftpd`). If the line is present, check if it is commented (by a `#`) or not. If it is commented, delete the `#`.

If the TFTP service is not present, add the following line:

```
# /etc/inetd.conf  
tftpd dgram udp wait root /usr/sbin/tcpd in.tftpd /tftpboot
```

The last column of the line describes the arguments of the daemon as it is started. One of these argument is the name of the directory of the sent and/or received files (default directory is `/tftpboot`).

Change this directory if required.

Execution

Sending the SIGHUP signal to the `inetd` daemon requests it to read the `/etc/inetd.conf` file again:

```
% ps -ef | grep inetd
root  510  1  0 16:55 ?        00:00:00 inetd
% kill -HUP 510
```

Or

```
% killall -HUP inetd
```

The `inetd` daemon also creates a file `/var/run/inetd.pid` that contains its process identifier.

Windows NT/Win32

RARP

No RARP server is available and/or was found.

BOOTP

No BOOTP server is available and/or was found.

TFTP

By using the freeware `TFTP32.EXE`, the board can download its firmware from a Win32 system.

Execute that program and look at the default directory it uses. Change it if required.

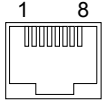
External Loopback Connectors

G

T1/E1/J1 Loopback Connector

The T1/E1/J1 loopback connector must connect signal 1 (IN1) to signal 4 (OUT1) and signal 2 (IN2) to Signal 5 (OUT2).

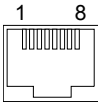
Table G-1. T1/E1/J1 RJ48C Connector

| | Signal |
|---|--------|
|  | |
| 1 | IN1 |
| 2 | IN2 |
| 3 | N/U |
| 4 | OUT1 |
| 5 | OUT2 |
| 6 | N/U |
| 7 | N/U |
| 8 | N/U |

Ethernet Loopback Connector

The Ethernet loopback connector must connect signal 1 (OUT+) to signal 3 (IN+) and signal 2 (OUT-) to Signal 6 (IN-).

Table G-2. Ethernet 10/100 RJ45 Connector

| | Signal |
|---|--------|
|  | |
| 1 | OUT+ |
| 2 | OUT- |
| 3 | IN+ |
| 4 | N/U |
| 5 | N/U |
| 6 | IN- |
| 7 | N/U |
| 8 | N/U |

P3LBTstConn and P4LBTstConn Loopback Connectors

Table G-3 lists the connections established by a P3LBTstConn loopback connector on P3.

Table G-3. P3LBTstConn - Connections

| | | | | |
|-----------|----|---|----|-----------|
| TXSOC | 1 | → | 64 | RXSOC |
| TXCLAV | 5 | ← | 18 | RXCLAV |
| RXADDR(3) | 7 | ← | 17 | TXADDR(3) |
| TXADDR(2) | 19 | → | 24 | RXADDR(2) |
| TXADDR(0) | 29 | → | 34 | RXADDR(0) |
| TXD(7) | 35 | → | 40 | RXD(7) |
| TXD(6) | 37 | → | 42 | RXD(6) |
| TXD(5) | 47 | → | 46 | RXD(5) |
| TXD(4) | 49 | → | 48 | RXD(4) |
| TXD(3) | 53 | → | 52 | RXD(3) |
| TXD(2) | 55 | → | 54 | RXD(2) |
| TXD(1) | 59 | → | 58 | RXD(1) |
| TXD(0) | 61 | → | 60 | RXD(0) |
| RXENB_N | 16 | ← | 22 | TXENB |
| TXADDR(1) | 28 | → | 30 | RXADDR(1) |
| GND | 44 | → | 39 | PTENB_N |

Figure G-1 shows a P3LBTstConn loopback connector.

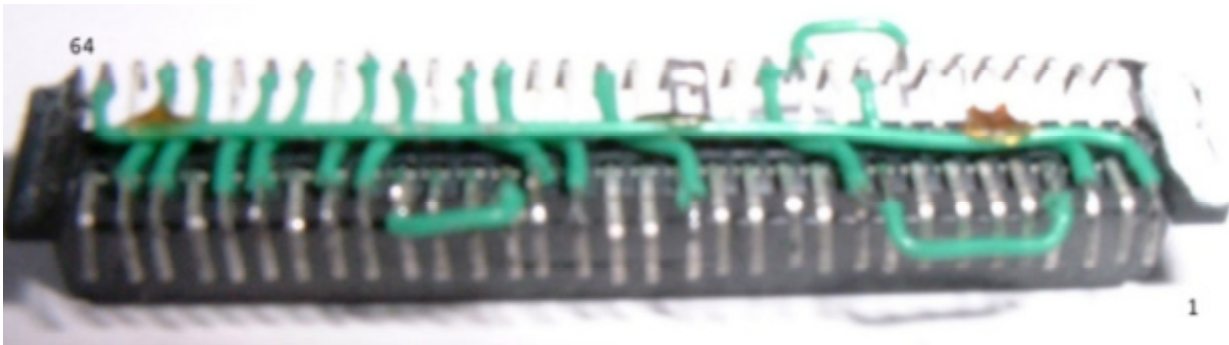


Figure G-1. P3LBTstConn - Example

Table G-4 lists the connections established by a P4LBTstConn loopback connector on P4.

Table G-4. P4LBTstConn - Connections

| | | | | |
|-------|----|---|----|-------|
| DOH0 | 1 | ← | 4 | DIH1 |
| DIH0 | 2 | → | 3 | DOH1 |
| FSOH | 5 | ← | 6 | FSIH |
| CKOH | 7 | ← | 8 | CKIH |
| DSTI4 | 11 | ← | 12 | DSTO4 |
| DSTI5 | 13 | ← | 14 | DSTO5 |
| DSTI6 | 15 | ← | 16 | DSTO6 |
| DSTI7 | 17 | ← | 18 | DSTO7 |
| XL1_0 | 34 | → | 33 | RL1_0 |
| XL2_0 | 36 | → | 35 | RL2_0 |
| XL1_1 | 38 | → | 37 | RL1_1 |
| XL2_1 | 40 | → | 39 | RL2_1 |
| XL1_2 | 42 | → | 41 | RL1_2 |
| XL2_2 | 44 | → | 43 | RL2_2 |
| XL1_3 | 46 | → | 45 | RL1_3 |
| XL2_3 | 48 | → | 47 | RL2_3 |

Figure G-2 shows a P4LBTstConn loopback connector.

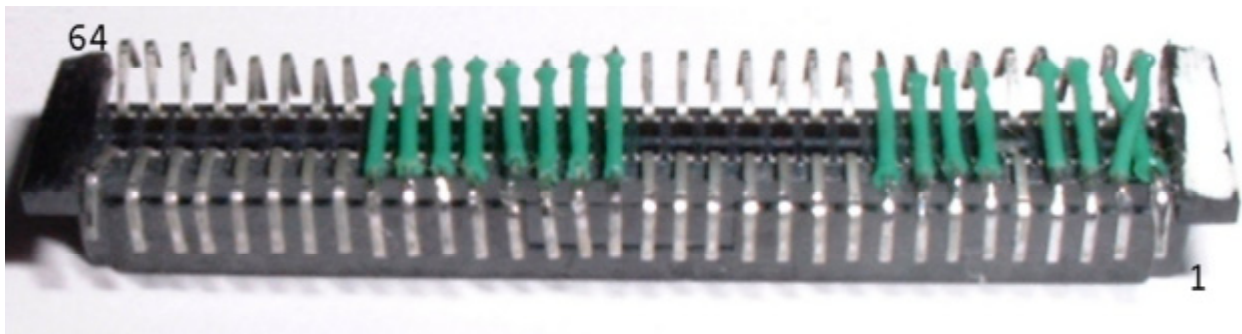


Figure G-2. P4LBTstConn - Example

Figure G-3 shows P3LBTstConn and P4LBTstConn loopback connectors mounted on a 4539.

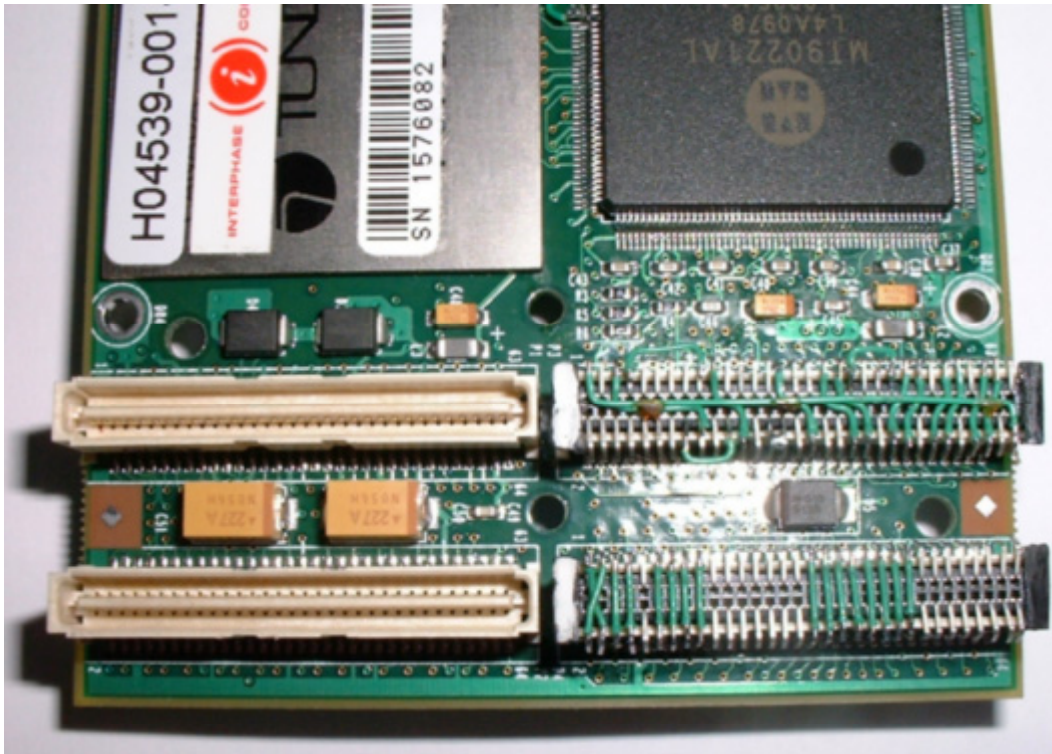


Figure G-3. P3LBTstConn and P4LBTstConn Mounted on a 4539

Figure G-4 shows a 4539 on a PCI carrier with the P3/4LBTstConn connectors.



NOTE

On the PCI carrier board, at the place where the P3LBTstConn and P4LBTstConn will be, there should be no components that would prevent plugging the 4539 on the carrier board—to prevent short circuits, place insulating tape at that place on the carrier.

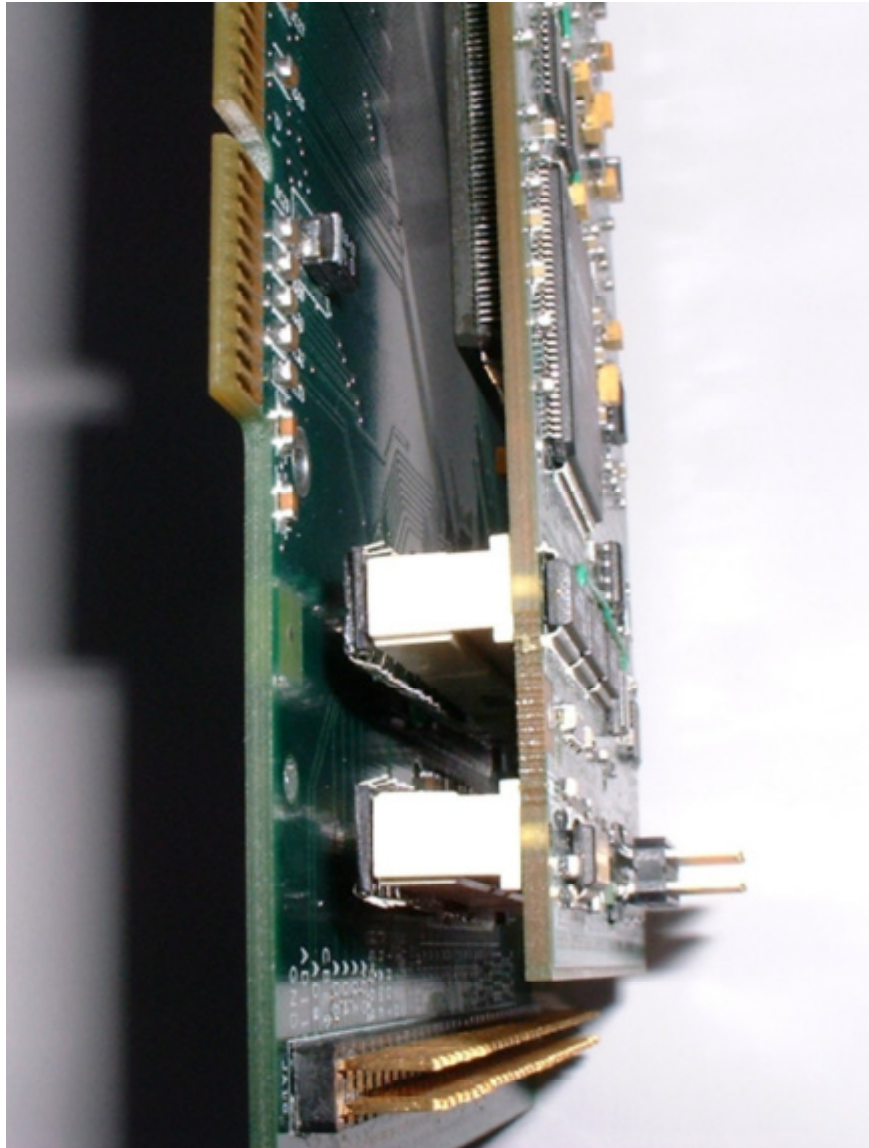


Figure G-4. A 4539 Equipped with P3/4LBTstConn on a PCI Carrier

