# *i*SPAN® 4576 PMC ATM Over OC-3/STM-1 Communications Interface Users Guide

Document No. UG04576-10-00-B01

Release Date: March 2005

## NOTE

**See Appendix D for Regulatory Statements/Conditions that affect the operation of this product.**
**The CE Declaration of Conformity can be found at www.interphase.com**

## Copyright Notice

© 2002–2005 by Interphase Corporation. All rights reserved.

Printed in the United States of America, 2005.

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
Parkway Centre 1
2901 North Dallas Parkway, Suite 200
Dallas, Texas 75093

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

All other trademarks are the property of their respective manufacturers.

# 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 Service

| | | |
|---|---|---|
| United States: | Telephone: | (214) 654-5666 |
| | Fax: | (214) 654-5624 |
| | E-Mail: | intouch@interphase.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

# 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


**Using This Guide** ..... v

Purpose ..... v
Audience ..... v
Icon Conventions ..... v
Text Conventions ..... vi
Documentation Updates ..... vi
Driver Updates ..... vii

**CHAPTER 1 Introduction**

Overview ..... 1
Product Features ..... 2
PCI ..... 2
ATM ..... 2
Platforms ..... 2
Optional Features ..... 3
System Requirements ..... 3
Hardware ..... 3
Memory ..... 3
Hard Disk ..... 3
Random Access Memory (RAM) ..... 3
Block Diagram ..... 5
Mindspeed RS8234 ATM SAR ..... 5
Mindspeed CX28250 PHY ..... 6
Optical Transceiver ..... 6
Controller Functions ..... 6
PCI Bus ..... 6
UTOPIA Interface ..... 6
Control Memory ..... 6
iSAR Application Programing Interface (API) ..... 6
Automatic Protection Switching ..... 8
Software Package ..... 8

**CHAPTER 2 Hardware Installation**

Overview ..... 9
Inspecting the Card ..... 9
Installing the Hardware ..... 10
Connecting to the Network ..... 11

**CHAPTER 3 Installing the Solaris Driver**

Overview ..... 13

Before You Start ..... 13
Package Contents ..... 13
Installing the Driver ..... 15
Removing the Driver ..... 17
Testing the Card ..... 18
VBR-CBR Support ..... 18

**CHAPTER 4 Installing the VxWorks Driver**

Overview ..... 19
Before You Start ..... 19
Package Contents ..... 19
Install Driver ..... 21
Download Driver ..... 21
Unload Driver ..... 21
Testing the Card ..... 21
VBR-CBR Support ..... 21

**CHAPTER 5 Installing the Linux Driver**

Overview ..... 23
Before You Start ..... 23
Package Contents ..... 23
/usr/src/<redhat,packages,montavista>/BUILD/isaratm/ ..... 23
/usr/src/<redhat,packages,montavista>/BUILD/isaratm/src/ ..... 24
Header Files ..... 24
/usr/src/iphase/isaratm/src/ ..... 25
Header Files ..... 26
Installing the ATM/API Driver on a PowerPC Platform ..... 27
Installing the ATM/API Driver on Intel Equipment ..... 29
Command-Line Options ..... 32
VBR-CBR Support ..... 33

**CHAPTER 6 IPOA Operations**

Overview ..... 35
Linux ..... 35
Solaris ..... 35
Introduction ..... 35
Adapter Initialization ..... 36
Flags ..... 36
Addr[8] ..... 36
Connection Open ..... 36
Enable ..... 36
Addr[8] ..... 36
Network Configuration ..... 36
Adding a Route ..... 37

isarRoute ..... 37
VxWorks ..... 38
Introduction ..... 38
General Flow of Events ..... 38
Operations Using Application isarconfig ..... 39
isarconf.x ..... 39
ipconf.yy ..... 39
Initialization ..... 40
Loading END mux Entity ..... 40
Open a Connection ..... 40
Operations Using the API ..... 41
Initialization ..... 41
Loading END mux Entity ..... 41
Open a Connection ..... 43
Adding a Route ..... 43
apiRoute Application ..... 44
Example Diagram ..... 45

**CHAPTER 7 VBR-CBR Support Utility**

Overview ..... 47
CBR ..... 47
Interactive Mode (menu) ..... 47
Parameter Descriptions ..... 47
Program interface ..... 49
VBR ..... 50

**CHAPTER 8 Adapter Quick Test**

Overview ..... 53

**CHAPTER 9 API Library Functions**

Introduction ..... 57
Usage ..... 58
isar_msg ..... 59
isar_rcv_ind ..... 62
isar_oam_ind ..... 64
isar_alarm_ind ..... 66

**CHAPTER 10 API Guide**

Operations and Programming Overview ..... 67
Basic Sequence of Events ..... 67
API Architecture ..... 68
Common Structures ..... 69
Header ..... 69
Tune ..... 69

API Groups ..... 71
Alarm ..... 71
Structure ..... 71
Prototypes ..... 72
API Sub Commands ..... 72
Alarm Open ..... 72
Alarm Close ..... 73
Alarm Info ..... 73
Alarm Tune ..... 73
Alarm Enable ..... 73
Alarm Disable ..... 73
Alarm Clear ..... 73
Alarm Log ..... 73
Alarm Events ..... 73
Control ..... 74
Structure ..... 75
Prototype ..... 75
API Sub Commands ..... 75
Initialize ..... 76
Trace enable ..... 78
Trace disable ..... 78
Loopback enable ..... 78
Loopback disable ..... 79
RS tune ..... 79
Unload ..... 79
Close ..... 80
Query ..... 80
Structures ..... 80
Prototype ..... 82
API Sub Commands ..... 82
Query Functions ..... 82
PHY ..... 85
Structures ..... 85
Prototype ..... 87
API Sub Commands ..... 87
PHY Functions ..... 87
VCC ..... 89
Structures ..... 89
Prototypes ..... 91
API Sub Commands ..... 91
VCC Functions ..... 91
OAM ..... 99
Structures ..... 99
Prototypes ..... 99
API Sub Commands ..... 99
OAM Functions ..... 100
Support ..... 102
Support API ..... 102

Support Functions ..... 102
ERROR Codes ..... 103
API Error Codes ..... 104
Driver Error Codes ..... 105

## CHAPTER 11 Troubleshooting

Overview ..... 109
Interpreting LEDs ..... 109
Problems and Possible Solutions ..... 110
Link and Status Problems ..... 110
Boot Problems ..... 112
Network Problems ..... 114

## APPENDIX A ATM isartest Guide

Introduction ..... 115
Purpose ..... 115
Menu Tree ..... 115
Requirements ..... 117
Getting Started ..... 118
Main Menu ..... 119
Description of Basic Operation ..... 119
Menu Selections ..... 119
[a]larm ..... 119
[C]lear ..... 119
[c]lose ..... 120
[d]isable ..... 120
[e]nable ..... 120
[i]nfo ..... 120
[o]pen ..... 120
[t]une ..... 121
[x]it ..... 121
[c]ontrol ..... 121
[i]nit ..... 121
[l]oopback ..... 122
[r]sTune ..... 122
[t]race ..... 123
[u]nld ..... 123
[x]it ..... 123
[d]isplay ..... 123
[c]onn ..... 123
[p]aramters ..... 123
[r]xbuf ..... 124
[s]avebuf ..... 124
[S]eqList ..... 124
[t]xbuf ..... 124
[T]imers ..... 124

[v]clist ..... 124
[h]elp ..... 125
[o]am ..... 125
[c]lose ..... 125
[d]isVC ..... 125
[D]isRS ..... 125
[e]nVC ..... 125
[E]nRS ..... 125
[o]pen ..... 125
[t][T]x ..... 125
[x]it ..... 125
[p]hy ..... 126
[c]ell ..... 126
[e]nable ..... 126
[g]en ..... 126
[i]nfo ..... 126
[I]ntr ..... 126
[m]asks ..... 127
s[o]net ..... 127
[s]tats ..... 127
[S]tatus ..... 128
[t]oggle ..... 128
[x]it ..... 128
[P]arams ..... 128
[e]cho ..... 128
[f]lag ..... 128
[i]nfoTmo ..... 129
[l]en ..... 129
[L]ogfile ..... 129
[m]tu ..... 129
[s]eed ..... 130
[t]xrxflow ..... 130
[T]xQfull ..... 130
[q]uery ..... 130
[a]dap ..... 130
[d]river ..... 130
[h]wconf ..... 131
[l]ink ..... 131
[L]ib ..... 131
[p]ort ..... 131
link[s]tat ..... 132
[x]it ..... 132
[t]ests ..... 132
[t][T]x ..... 134
[v]c ..... 134
[a]ctvc ..... 134
[c]lose ..... 135
[d]isable ..... 135

[e]nable ..... 135
[i]nfo ..... 135
[o]pen ..... 135
[s]tats ..... 136
[t][T]x ..... 136
[x]it ..... 137
[x]it ..... 137

## APPENDIX B ATM isaralarm Guide

Introduction ..... 139
Purpose ..... 139
isaralarm Menu Tree ..... 139
Requirements ..... 140
Getting Started ..... 140
Main Menu ..... 141
Description of Basic Operation ..... 141
Menu Selections ..... 141
[a]larm ..... 141
[C]lear ..... 141
[c]lose ..... 141
[d]isable ..... 142
[e]nable ..... 142
[i]nfo ..... 142
[o]pen ..... 142
[t]une ..... 143
[x]it ..... 143
[d]isplay ..... 143
[a]larm ..... 143
[c]onn ..... 144
[e]rrbuf ..... 144
[f]lag ..... 145
[h]ost ..... 145
[p]hy ..... 145
[s]tat ..... 145
[t]xbuf ..... 146
[v]clist ..... 146
[h]elp ..... 146
[o]am ..... 146
[c]lose ..... 146
[d]isVC ..... 146
[D]isRS ..... 147
[e]nVC ..... 147
[E]nRS ..... 147
[o]pen ..... 147
[t][T]x ..... 147
[x]it ..... 147
[p]hy ..... 147

[c]ell ..... 147
[e]nable ..... 148
[g]en ..... 148
[i]nfo ..... 148
[I]ntr ..... 148
[m]asks ..... 148
s[o]net ..... 149
[s]tats ..... 149
[S]tatus ..... 149
[t]oggle ..... 149
[x]it ..... 150
[P]arams ..... 150
[f]lag ..... 150
[l]ogfile ..... 150
[v]cDebounce ..... 150
[q]uery ..... 150
[a]dap ..... 151
[d]river ..... 151
[h]wconf ..... 151
[l]ink ..... 151
[L]ib ..... 152
[p]ort ..... 152
link[s]tat ..... 152
[x]it ..... 152
[x]it ..... 152

**APPENDIX C Specifications**

Interface Specifications ..... 153
Power ..... 153
Connectors/Cables ..... 153
Operating Environment ..... 153
Storage Environment ..... 154

**APPENDIX D Regulatory Statements**

FCC ..... 155
Canadian ..... 155
European ..... 156
EN60950–IEC60950 Safety Standard ..... 156
Interphase Fiber Products Compliance, LASER ..... 157
Usage Restrictions ..... 157
Restrictions d'utilisation ..... 157
Interphase Fiber Products Compliance, LED ..... 157

# Using This Guide

## Purpose

This Users Guide provides information about the Interphase ***i***SPAN® 4576 PMC ATM Over OC-3/STM-1 Communications Interface card. It describes general features, hardware and software installation procedures (with safety precautions), and the software modules that comprise the product. It also provides detailed information about the external software interface and about how the software modules interact.

## Audience

This manual assumes that its audience has a general understanding of computing and networking terminology.

## 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.

1. Compare the revision level for the Users Guide with the one you have. If a later revision is available, click on the document name (such as 4576 Users Guide or 3000 BDK Users Guide).
2. A form will open, which you are required to fill out to request Interphase documentation. Press the `submit` button on the bottom of the page after filling it out, the document will be sent to you via e-mail.

# Documentation Updates

The latest documentation (in Adobe® Acrobat® pdf) for our current products are available on our Web site. Interphase recommends our customers visit the web site to verify that they have the latest version of the documentation.

1. Access the following web page:

   http://www.interphase.com
2. Move the mouse (or other pointer) and click on `Product Selector`.
3. On the Interphase Product Selector page, use the `Product Selector` to either select the type of product required, or if not sure, select `All Products`.
4. A new web page with a list of the currently offered products will appear. Choose your product by clicking on the product name.
5. A new web page appears with an overview of the product. Click on `Documentation`.

6. Compare the revision level for the Users Guide with the one you have. If a later revision is available, click on the document name (such as ***i***SPAN 4576 Users Guide or ***i***NAV 3000 BDK Users Guide).

7. A form will open, which you are required to fill out to request Interphase documentation. Press the `submit` button on the bottom of the page after filling it out, the document will be sent to you via e-mail.

## Driver Updates

Contact our Technical Support Department at swlib@iphase.com to determine if updated drivers are available for your product.

When contacting technical support, please be sure to provide your name, company name and address, phone number, product name, driver version (if applicable), OS and version (if applicable) and serial number. Providing this information will help speed up our response.

# Introduction 1

## Overview

The *i*SPAN® 4576 PMC ATM Over OC-3/STM-1 Communications Interface card supports a data rate of 155 Mbps over fiber. This card adheres to the PMC (PCI Mezzanine Card) mechanical layout for use in mezzanine systems, such as CompactPCI® systems. It is based on the Mindspeed™ RS8234 ATM integrated SAR, and Mindspeed CX28250 PHY. The RS8234 SAR has a 32-bit, 33 MHz PCI interface, with 32 K VCs, that supports two CX28250 PHYs on a UTOPIA 1 bus, along with CBR, VBR, and UBR traffic in hardware.

The 32-bit, 33 MHz PCI interface provides the bandwidth necessary to continuously send and receive ATM SONET frames at 155 Mbps rate. Onboard DMA engines on the PCI bus transfer ATM data to and from the board. Both 3.3 V and 5 V PCI signaling environments are supported.

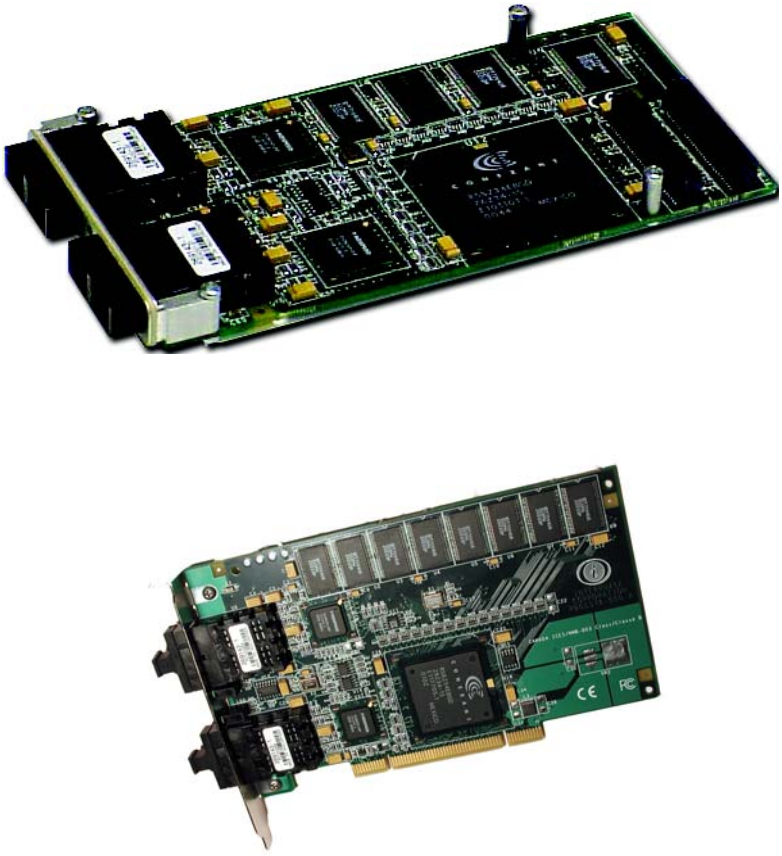


**Figure 1-1. 4576 Communications Interface Card**

# Product Features

## PCI

- 32-bit, 33 MHz PMC compliant adapter
- PCI Rev 2.1 with universal (3.3 V or 5 V signaling)
- PCI bus slave features:
  - 32-bit, address decoded with medium speed device
  - Memory and Configuration Read & Write cycles supported
  - Single access only, no bursting
- PCI bus master features:
  - 32-bit, zero wait transfers for up to 132 MBps burst DMA rate
  - 512x32-word Receive FIFO and 16x36-word Transmit FIFO buffering
  - Burst sizes up to thirteen 32-bit words for read, and fourteen 32-bit words for write

## ATM

- Single OC-3/STM-1 capable SAR, single or dual PHY
- Physical interface for 155 Mbps optical transceiver with SC connector
- 256-byte receive FIFO, variable-length transmit FIFO, one to nine cells
- 4 MB (optional 8 MB) of on-board memory for storage of ATM connection parameters
- 32 K VCs simultaneous full-duplex connections
- AAL0 and AAL5 adaptation layers:
  - AAL0: Single- or multi-cell frames
  - AAL5: Encapsulation provided in hardware
- OAM F4/F5 frame support
- Constant Bit Rate (CBR), Variable Bit Rate (VBR), and Unspecified Bit Rate (UBR) classes of service
- Automatic Protection Switching (APS) (device-level support for Class 1+1)

## Platforms

- VxWorks® (5.4) API for PowerPC™
- Solaris™ (7, 8, 9) API for SPARC® 32/64
- Linux® (2.4/2.6)
  - API for Intel® and PowerPC
  - ATM Miniport for Intel and PowerPC

### Optional Features

- Single ATM connection, or redundant two connections
- Optical transceivers with ST connectors
- Multimode/Single mode fiber

# System Requirements

## Hardware

The 4576 must be installed in a PMC slot with 3.3 V and 5 V available.

## Memory

### Hard Disk

The system must have 1 Meg of hard disk space available.

### Random Access Memory (RAM)

The system requires a minimum of 290 K and a maximum of 43.1 Meg of RAM. The amount of RAM you require is dependent on user configuration and can be approximated by the summation of the items in Table 1-1:

**Table 1-1. Memory Requirements**

| Item | Multiplier | Resource |
|---|---|---|
| SegVccs (number of segmentation VCs) | 128 | VC table |
| SegQSize (segmentation queue size) | (MTU (maximum frame size) + 128 + 8) | tbd + tsq |
| RemQSize (reassembly queue size) | (MTU + 128 + 16 + 12) | irbd +rsq + rdb |
| VC VP selection (256) | 16 | vcvps |
| EEPROM (1) | 256 | EEPROM |

Table 1-2 shows the user-configurable parameter values from the `isar` config file.

**Table 1-2. User-Configurable Parameter Values**

| Item | Allowed Values |
|---|---|
| MTU | 48 through 64 K |
| SegVccs | 1 through 64 K |
| SegQSize | 64, 256, 1 K, 4 K |

**Table 1-2. User-Configurable Parameter Values (cont)**

| Item | Allowed Values |
|---|---|
| RsmQSize | 64, 256, 1 K, 4 K |

A minimum configuration is shown in Table 1-3. The values were obtained by using the small user values (SegVccs=32, SegQSize=64, RsmQSize=64, MTU=2 K).

**Table 1-3. Minimum Configuration Memory Requirements**

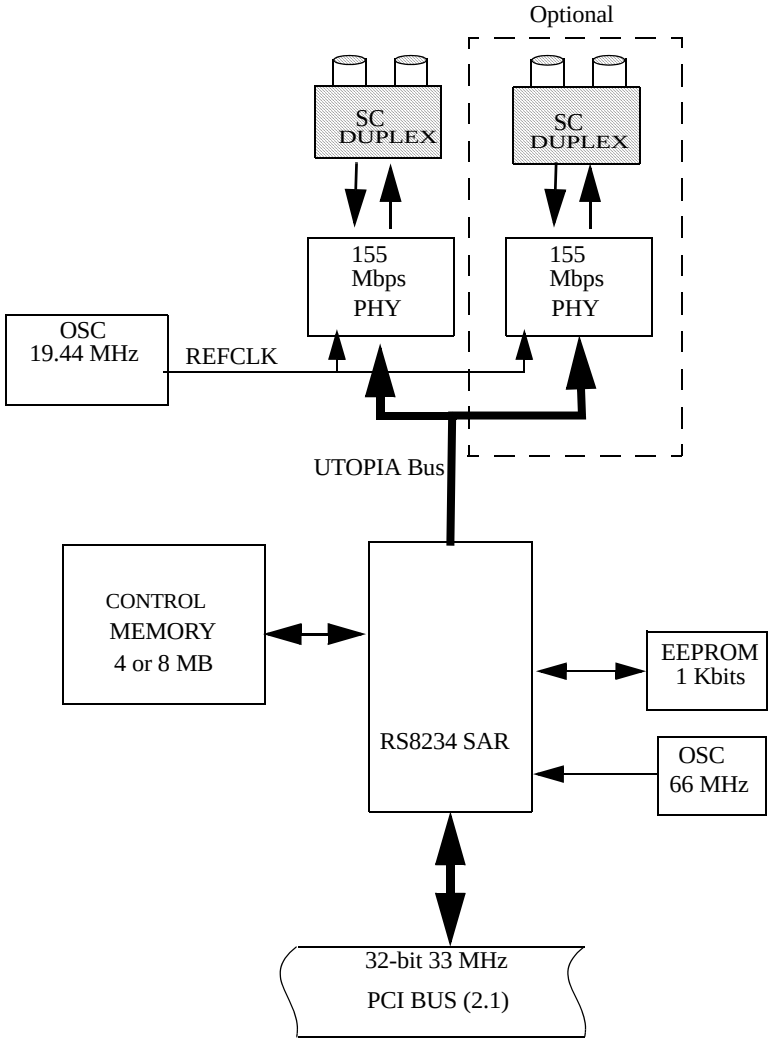
| Item | Multiplier | Value |
|---|---|---|
| SegVccs = 32 | 128 | 4 K |
| SegQSize = 64 | (2 K + 136) | 140 K |
| RsmQSize = 64 | (2 K + 156) | 141 K |
| vcvps = 256 | 16 | 4 K |
| eeprom = 256 | | 1 K |
| | **Total** | 290 K |

A maximum configuration is shown in Table 1-4. The values were obtained by using the large user values (SegVccs=64 K, SegQSize=4 K, RsmQSize=4 K, MTU=4 K).

**Table 1-4. Maximum Configuration Memory Requirements**

| Item | Multiplier | Value |
|---|---|---|
| SegVccs = 64 K | 128 | 8.4 M |
| SegQSize = 4 K | (4 K + 136) | 17.3 M |
| RsmQSize = 4 K | (4 K + 156) | 17.4 M |
| | **Total** | 43.1 M |

## Block Diagram




## Mindspeed RS8234 ATM SAR

- Conforms to the ATM Forum recommendations
- 32-bit 33 MHz PCI 2.1 compliant with universal signaling
- UTOPIA 1 interface, with two PHY chip selects
- 32-bit local bus interface
- 32 K of total Virtual Channels
- Hardware support for CBR,VBR, and UBR services
- OAM F4/F5 frame support

### Mindspeed CX28250 PHY

- Meets ATM Forum standards
- 155 Mbps full duplex operation
- OC-3/STM-1
- UTOPIA 1 bus interface

### Optical Transceiver

- 1X9 form factor
- SC connector (Contact Interphase for other connectors.)
- Multimode 1300 nm LED for short range (< 2 km)
- Single mode 1300 nm LASER for intermediate range (<15 km) or long range (40 km)

## Controller Functions

### PCI Bus

- Complies to PCI Revision 2.1 specification for 32-bit, 33 MHz environment.
- Universal signaling
- 3.3 V and 5 V

### UTOPIA Interface

- UTOPIA 1 bus
- UTOPIA signals are shared between two PHY devices

### Control Memory

The ***i***SPAN 4576 uses the control memory to store parameters for segmentation and reassembly. The maximum size of the control memory is 4 MBytes for up to 32 K VCs.

## *i*SAR Application Programing Interface (API)

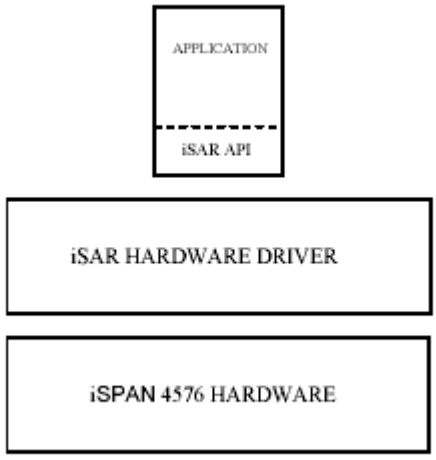
The ***i***SAR API provides a seamless, robust interface between an application and the underlying hardware, isolating the application from dealing directly with the complexity of the hardware, simplifying the application development process.

The API is implemented as a series of synchronous messages exchanged between the application and API. These messages are divided into six main groups:

- Query
  - Current state, configuration, and statistical information

- Control
  - Configuration, management
- VCC (connections)
  - Set-up/tear-down
  - Transmit/receive
  - Statistical information
  - Flow control
- Phy
  - Selection, toggle
  - Read registers
- OAM
  - Set-up/tear-down
  - Transmit/receive
  - Statistical information
  - Flow control
- Alarms
  - SAR
  - Phy
  - VCC
  - Line

The ***i***SAR API data path is through the API, not the native host interface (i.e., streams or sockets). Asychronous events such as receive data and alarm indications are executed via user-specified callback functions.



## Automatic Protection Switching

The RS8234 SAR has a UTOPIA 1 bus, with two PHY chip selects. In conjunction with the CX28250 PHYs, the second PHY chip select allows the second PHY to share the same UTOPIA bus for APS capability. Both PHYs transmit and receive all data. Both PHYs receive data from the redundant network, but only the primary PHY transfers data back to the ATM SAR. APS operating parameters and event notification are available through the API, including the selection of active and/or protective PHY.

The receive outputs of the protective PHY are in high impedance state.

Interphase provides the ***i***SPAN 4576 in two basic configurations to support APS operation:

- Single PHY for 1:n implementation
- Dual PHY for 1+1 implementation

## Software Package

Each of the supported platform software packages includes the following components:

- Low level driver (binary)
- Associated API library (binary)
- Configuration, diagnostic, and troubleshooting tools (binary)
- API header files: Required for application development
- Sample configuration file
- Functional sample source of diagnostic utilities
- Users documentation: Documentation includes details on installing the 4576 hardware module, an overview of the API, and detailed information regarding each API function call

# Hardware Installation 2

## Overview

The 4576 ATM interface card is designed to be installed in PMC slots. This chapter describes the procedures for physically installing the card, which include:

- Inspecting the card, described in the next topic.
- Installing the card in a host PMC slot, described in *Installing the Hardware* on page 10.
- Connecting the card to the ATM network, described in *Connecting to the Network* on page 11.

The only tools required are a grounding strap and a #1 Phillips screwdriver. For specification information, see *Specifications on page 153*.

## Inspecting the Card

Before installing the card in your computer, visually inspect it for damage that might have occurred during shipment from the factory.

### *CAUTION*

**The card is packed in an antistatic bag to protect it during shipment. Keep the card in its protective antistatic bag until you are ready to install it in the host computer. To prevent damage to the card due to electrostatic discharge, wear a grounding strap and handle the card only by its edges. Do not touch its components or any metal parts other than the faceplate.**

1. Open the shipping container and carefully remove its contents.
2. Inspect each item for damage. If you find any omissions or damage, contact your supplier and the carrier (for example, UPS or Federal Express) that delivered the package.

### *WARNING*

**Do not install, or apply power to, a damaged board. Failure to observe this warning could result in extensive damage to the board and/or the system.**

# Installing the Hardware

## WARNING

**Your computer operates at voltages that can be lethal. Before you remove the computer cover, carefully review the following procedures and observe all cautions and warnings to protect yourself and to prevent damage to the system. Use only insulated or nonconductive tools**

To install the card in a host machine, attach a grounding strap to your wrist or ankle and do the following:

1. With power disconnected, gain access to the mezzanine location on the motherboard.
2. If the motherboard is easily removed, as in most CompactPCI systems, remove it from the chassis.
3. Remove the blank cover from the aperture of the PMC slot.
4. Carefully remove the card from its antistatic bag.
5. Hold the card at an angle and insert the card through the rear of the faceplate of the motherboard while aligning the dual mating connectors on the motherboard to the connectors (P1 and P2) on the card.

   Figure 2-1 on page 11 illustrates the installation of a typical PMC card in a CompactPCI motherboard.
6. Align the standoff post on the motherboard with the matching hole in the card and carefully press the card into place.
7. Fasten the cards together with screws.
8. Reinstall all parts removed in earlier steps.

Continue the installation with the procedure in the next section.

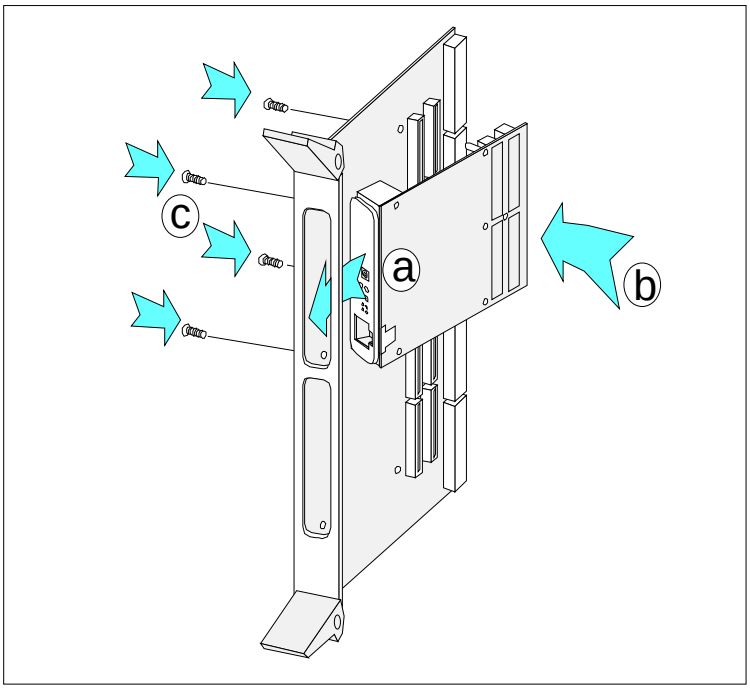



**Figure 2-1. Installing a PMC Card**

# Connecting to the Network

With the card installed in the chassis as discussed in the previous section, you are now ready to connect the card to the network. Table 2-1 lists the cables and connectors required for the card.

**Table 2-1. Cable/Connector Requirement**

| Connector | Medium | Configuration |
|---|---|---|
| SC Duplex | Multimode fiber | 62.5/125 micron |
| SC Duplex | Single Mode fiber | 10/125 micron |

To connect the card to the network:

1. Attach the appropriate network connector to the PMC ATM card.

   Figure 2-2 shows the SC Duplex connection for ATM over fiber.

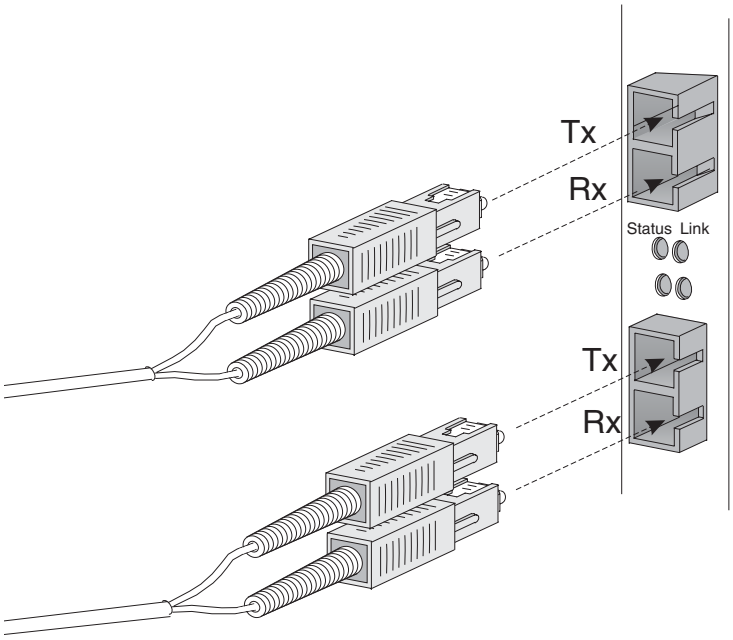



**Figure 2-2. SC Duplex Connection**

2. Turn on the power to the host computer.

Installation of the card is complete.

Go to the appropriate software installation chapter based on your system requirements:

- *Installing the Solaris Driver* on page 13
- *Installing the VxWorks Driver* on page 19
- *Installing the Linux Driver* on page 23

# Installing the Solaris Driver 3

## Overview

This chapter explains how to install the card's Solaris driver in an end-station running Solaris version 7, 8, or 9. It provides information about the following:

- Tasks that should be done before you start the installation, described in the next topic, Before You Start
- Driver installation, described in *Installing the Driver* on page 15
- Driver removal, described in *Removing the Driver* on page 17

## Before You Start

Before you install the driver, it is recommended that you:

- Read any *Release Notes* in your installation kit.
- Make sure the computer meets the minimum requirements listed in *System Requirements* on page 3.
- Install the card in the computer (see *Hardware Installation* on page 9).
- Read through the entire sequence of installation steps.
- Back up the system.

## Package Contents

The following files are installed at "`usr/isar/src`":

- `apiRecv.c` AAL5 receive only application one adapter, one VC
- `apiRecvM.c` AAL5 receive only application multi-adapter, multiple VCs
- `apiRecvRaw.c` AAL0 receive only application one adapter, one VC
- `apiSend.c` AAL5 transmit/receive application one adapter, multiple VCs
- `apiSendRaw.c` AAL0 transmit/receive application one adapter, one VC
- `apiRoute.c` IP route add/delete/show application
- `isaralarm.c` Test application for alarms
- `isartest.c` Test application for any API
- `isar_alarms.h` Alarm API header file
- `isar_api.h` Main API header file
- `isar_codes.c` API error support functions
- `isar_codes.h` API error codes
- `isar_phy.h` Phy API header file
- `isar_solaris.h` OS specific API header file

- `isar_types.h` API data types header file

The following file is installed at "`/etc/atm/`":

- `isarconf.0`

The following files are installed at "`/bin/`", for each set of [`filename`, `filename32`, `filename64`] `filename` is the parent executable script that calls either `filename32` or `filename64` based on the system configuration indicated by '`isainfo -b`'.

The `apiRecv*` and `apiSend*` functions are set up to work with each other. The send functions set a sequence number and stop flag in the packet payload. The receive functions validate sequence numbering and calculate overall time between first packet and last packet received.

- `apiRecv` AAL5 receive only application for one adapter, one VC
- `apiRecv32`
- `apiRecv64`
- `apiRecvM` AAL5 receive only application multi-adapter, multiple VCs
- `apiRecvM32`
- `apiRecvM64`
- `apiRecvRaw` AAL0 receive only application for one adapter, one VC
- `apiRecvRaw32`
- `apiRecvRaw64`
- `apiSend` AAL5 transmit/receive application one adapter, multiple VCs
- `apiSend32`
- `apiSend64`
- `apiSendRaw` AAL0 transmit application one adapter, one VC
- `apiSendRaw32`
- `apiSendRaw64`
- `apiRoute` IP route add, delete, show application
- `apiRoute32`
- `apiRoute64`

- `isaralarm` Test application for opening and handling alarms
- `isaralarm32`
- `isaralarm64`
- `isarconfig` Configuration application that parses `isarconf.x` files
- `isarconfig32`
- `isarconfig64`
- `isarRoute` Script to issue system route command and use `apiRoute` for API
- `isartest` Test application to exercise any API
- `isartest32`

- isartest64
- isartool — Diagnostic tool for low level debug only
- isartool32
- isartool64
- isarxbr — Application to determine CBR and VBR parameters
- isarxbr32
- isarxbr64

# Installing the Driver

To get started:

- Insert and mount the CD-ROM.

Then do the following:

1. Log in as **root**.
2. Check the available disk space by entering **df -k**
3. If needed, check for an existing installation of the driver by entering **pkginfo**

   If ISARatm is in the list of installed packages, remove it as explained in *Removing the Driver* on page 17.
4. Run the installation script by entering the **pkgadd** command.

   **pkgadd -d SX000861-Xxx**

   Where Xxx is the version number of your software.

   The following is displayed:

```
The following packages are available:
1    ISARatm   x576 iSAR API driver
               (sparc) SX00861-x00
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) (?, ??, q) : 1
```

   The installation might take several minutes to complete.

   The following is a sample of the screen output during driver installation. The prompts that appear on your screen during driver installation depend on your responses.

```
Processing package instance <ISARatm> from </path/SX00861-x00>

Interphase x576 iSAR API driver
(sparc) SX00861-x00
Interphase Corp

A directory must be provided to house all
Source and Library files needed to develop
software applications which access the iSAR API.

Directory Name: (default /usr/isar)
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
```

```
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <ISARatm> [y,n,?] y
Installing Interphase x576 iSAR API driver as <ISARatm>

## Executing preinstall script.
## Installing part 1 of 1.
/etc/atm/isarconf.0
/etc/rc2.d/S98isar
/kernel/drv/isar
/kernel/drv/sparcv9/isar
/usr/bin/isaralarm
/usr/bin/isaralarm32
/usr/bin/isaralarm64
/usr/bin/isartest
/usr/bin/isartest32
/usr/bin/isartest64
/usr/bin/isartool
/usr/bin/isartool32
/usr/bin/isartool64
/usr/bin/isarxbr
/usr/bin/isarxbr32
/usr/bin/isarxbr64
/usr/isar/src/isar_alarms.h
/usr/isar/src/isar_api.h
/usr/isar/src/isar_codes.c
/usr/isar/src/isar_codes.h
/usr/isar/src/isar_phy.h
/usr/isar/src/isar_types.h
/usr/isar/src/isaralarm.c
/usr/isar/src/isartest.c

/usr/isar/src/isarcbr.c
/usr/isar/src/isarcbr.h
/usr/lib/libisar.so
/usr/lib/sparcv9/libisar.so
[ verifying class <none> ]
Modifying /etc/devlink.tab
Modifying /etc/path_to_inst
[ verifying class <sed> ]
## Executing postinstall script.
iSAR path=/usr/isar
Dec 10 12:04:05 eng-lab-097 isar: Interphase x576 iSAR API driver ver 3.0 build sun.4

Installation of <ISARatm> was successful.

*** IMPORTANT NOTICE ***
        This machine must now be rebooted in order to ensure
        sane operation.  Execute
               shutdown -y -i6 -g0
        and wait for the "Console Login:" prompt.
```

5. During installation the system will prompt for a source directory path.

```
A directory must be provided to house all
Source and Library files needed to develop
software applications which access the iSAR API.
Directory Name: (default /usr/isar)
```

The Solaris API driver installs all files, source and binary, needed to develop user mode software to interface with the API. By default, these files are deposited in the "`/usr/isar`" directory. The user has the option to specify a different directory for these files to be installed.

To accept the default directory location press the **Enter** key.

To specify another location, the enter the full path of the directory to which the files should be deposited and press the **Enter** key.

6. At the final prompt, `Do you want to continue with the installation of this package`, respond with **`y`** if you want to continue.

   (Or respond **`n`** to cancel the installation and delete the temporary file holding the work done to this point.)

7. When installation of the driver is complete, remove the CD-ROM from the local drive and reboot the machine with the command:

   **`shutdown -y -g0 -i6`**

## Removing the Driver

To remove a driver from the end station, do the following:

1. Log in as root.
2. Enter the **`pkgrm`** command followed by the driver name. For example, to remove `ISARatm`, enter the command:

   **`pkgrm ISARatm`**

3. Answer the prompts or press **Enter** to accept each default, as shown in Figure 3-1 on page 18.
4. Reboot the machine by entering:

   **`shutdown -y -g0 -i6`**

### *CAUTION*

**When the driver package is removed from a system, the configuration file `/etc/atm/isarconf.0` can be saved as `/etc/atm/isarconf.save`. If the driver package is installed again, `/etc/atm/isarconf.save` can be moved to `/etc/atm/isarconf.0` to quickly restore the configuration of the previous installation.**

## NOTE

**You must reboot the system before you can reinstall the driver.**

```
prompt->pkgrm ISARatm

The following package is currently installed:
   ISARatm        Interphase x576 iSAR API driver
                   (sparc) SX00861-x00

Do you want to remove this package? {y,n,?,q] y

## Removing installed package instance <ISARatm>

This package contains scripts which will be executed with super-user
permission during the process of removing this package.

Do you want to continue with the removal of this package [y,n,?,q] y
## Verifying package dependencies.
## Processing package information.
## Executing preremove script.
## Removing pathnames in class <sed>
Modifying /etc/path_to_inst
Modifying /etc/devlink.tab
## Removing pathnames in class <none>
/usr/lib/libisar.so
/usr/lib/64/libisar.so
/usr/bin/isarxbr64
/usr/bin/isarxbr32
/usr/bin/isarxbr
/usr/bin/isartest64
/usr/bin/isartest32
/usr/bin/isartest
/usr/bin/isartool64
/usr/bin/isartool32
/usr/bin/isartool
/usr/bin/isaralarm64
/usr/bin/isaralarm32
/usr/bin/isaralarm
/kernel/drv/sparcv9/isar
/kernel/drv/isar
/etc/rc2.d/S98isar
/etc/atm/isarconf.0
```

**Figure 3-1. Driver Removal Sample**

# Testing the Card

Go to *Adapter Quick Test* on page 53 for procedures for performing a quick test on the card.

# VBR-CBR Support

Go to *VBR-CBR Support Utility on page 47* for VBR or CBR support functions.

# Installing the VxWorks Driver 4

## Overview

This chapter explains how to install the card's VxWorks driver in an end station running VxWorks version 5.4 or later. It provides information about the following:

- Tasks that should be done before you start the installation, described in the next topic, *Before You Start*
- Driver installation, described in *Install Driver* on page 21
- Driver removal, described in *Unload Driver* on page 21

### NOTE

**`isar_board.c` must either be included in all applications, or the application must provide the BSP structure defined in the file.**

## Before You Start

Before you install the driver, it is recommended that you:

- Read any documentation in your VxWorks and BSP system.
- Get familiar with the Tornado environment.
- Make sure the computer meets the minimum requirements listed in *System Requirements* on page 3.
- Install the card in the computer (see *Hardware Installation* on page 9).
- Read through the entire sequence of installation steps.
- Back up the system.
- Make sure you have the correct driver package.

## Package Contents

This release includes following object modules and source files:

- `objs/isarPpc.out` A pre-compiled driver module for the PowerPC 604/603
- `objs/isartest.o` A pre-compiled test module for the Motorola MCP750 BSP
- `objs/isarPkg.o` A pre-compiled test utility that integrates the driver and test modules
- `isar_api.h` The API header file needed to build application modules
- `isar_board.c` This file needs to be modified to specify which BSP the driver will work with (default is Motorola MPC750)

- `isar_board.h` The header file needed by `isar_board.c`
- `isar_codes.h` A header file needed to build application modules
- `isar_diag.h` A header file needed to build isartest
- `isar_types.h` A header file needed to build application modules
- `isar_vxworks.h` A header file needed to build application modules
- `isaralarm.c` Alarm/OAM utility source file
- `isarconfig.h` A header file needed to build application modules
- `isarconf.0` Configuration file read by the application when the driver is initialized
- `isartest.c` API test utility source file
- `isartool.h` The header file needed to build `isartest`
- `isarxbr.c` Support application source to calculate VBR parameters
- `isarxbr.h` The header file needed to build `isarxbr`
- `isarcbr.c` Support application source to calculate CBR parameters
- `isarcbr.h` The header file needed to build `isarcbr`
- `makefile` The make file for compiling the driver (if source package) and test modules.
- `makerule` PowerPC build rules to be customized for each target BSP
- `apiRecv.c` AAL5 receive only application for one adapter, one VC
- `apiRecvM.c` AAL5 receive only application for multi-adapter, multiple VCs
- `apiRecvRaw.c` AAL0 receive only application for one adapter, one VC
- `apiRoute.c` IP route add/delete/show application
- `apiSend.c` AAL5 transmit/receive application one adapter, multiple VCs
- `apiSendRaw.c` AAL0 transmit/receive application one adapter, one VC
- `config.c` Support function for `isarconfig` that does file parsing
- `configs.c` Support function that contains file parsing tables
- `isarconfig.c` Configuration application
- `pvcs.c` Support function to parse a file for VC parameters

- `config.h` Header file for config.c
- `isar_alarms.h` Alarm API header file
- `isar_phy.h` Phy API header file

- `apiRecv.o` AAL5 receive only application for one adapter, one VC
- `apiRecvM.o` AAL5 receive only application for multi-adapter, multiple VCs

| | |
|---|---|
| • `apiRecvRaw.o` | AAL0 receive only application for one adapter, one vc |
| • `apiRoute.o` | IP route add/delete/show application |
| • `apiSend.o` | AAL5 transmit/receive application one adapter, multiple VCs |
| • `apiSendRaw.o` | AAL0 transmit/receive application one adapter, one VC |
| • `isaralarm.o` | Test application for API alarms |
| • `isarconfig.o` | Configuration application |
| • `isartool.o` | Engineering tool for diagnostics and debug only |
| • `isarxbr.o` | CBR and VBR parameter calculation function |

## Install Driver

The driver module is independent of the Tornado environment and can be installed in whatever directory you specify. For this example, the directory is called `ISAR4576ATM`. The driver package is provided in zip format. Unzip the package in the `ISAR4576ATM` directory.

## Download Driver

There are two ways to download the driver (with PowerPC and MCP750 BSP as examples):

- Download `isarPpc.out` and `isartest.o` modules separately to a target system.
- Download the integrated module `isarPkg.o` to a target system.

## Unload Driver

There are two ways to unload the driver (with PowerPC and MCP750 BSP as examples):

- Enter **`unld inphAtmPpc.out`** from a WindSh.
- Enter **`reboot`** from a WindSh.

## Testing the Card

Go to *Adapter Quick Test* on page 53 for procedures for performing a quick test on the card.

## VBR-CBR Support

Go to *VBR-CBR Support Utility on page 47* for VBR or CBR support functions.

# Installing the Linux Driver 5

## Overview

This chapter explains how to install the card's Linux *i*SAR ATM/API driver in an end station (Intel or PowerPC platform) running the Linux 2.4.x or 2.6.x kernel. It provides information about the following:

- Tasks that should be done before you start the installation, described in the next topic, Before You Start
- PowerPC driver installation, described in *Installing the ATM/API Driver on a PowerPC Platform* on page 27
- Intel driver installation, described in *Installing the ATM/API Driver on Intel Equipment* on page 29
- Command-line options, described in *Command-Line Options* on page 32

## Before You Start

Before you install the driver, it is recommended that you:

- Read any documentation contained in your installation kit.
- Make sure the computer meets the minimum requirements listed in *System Requirements* on page 3.
- Install the card in the computer (see *Hardware Installation* on page 9).
- Read through the entire sequence of installation steps.
- Back up the system.

NOTE: All references to `${OS}` should be replaced with your host OS type,

such as Redhat, Montavista, Suse, etc.

All references to `${CPU}` should be replaced with your host CPU type,

such as i386, x86_pentium4, etc.

## Package Contents

The package installs into `/usr/src/<redhat,packages,montavista>/BUILD/isaratm/` then copies application files to `/usr/src/iphase/isaratm`. The following descriptions describe the relevant files for each location:

### /usr/src/<redhat,packages,montavista>/BUILD/isaratm/

- `AUTHORS` States Interphase as the author of this code
- `COPYING` States copy usage

- `COMPAT` States basic build and kernel compatibility
- `README` Build and usage information
- `files.txt` List of files copied to `/usr/src/iphase/isaratm` and built
- `Makefile` Master makefile for the driver and library
- `make_api` API library makefile
- `make_drvr.k24` Driver makefile for kernel 2.4.x
- `make_drvr.k26` Driver makefile for kernel 2.6.x
- `out` Binaries
- `src` Source files for driver, library and applications

## /usr/src/<redhat,packages,montavista>/BUILD/isaratm/src/

- `ia_dumps.c` Driver, app support function for dumping structures
- `ia_os_linux.c` Driver file
- `isar_api.c` Library file
- `isaratm.mod.c` Module symbol file created from build
- `isar_linux.c` Library file
- `linux_api.c` Driver API file
- `linux_atm.c` Driver STACK file, and ISR functions

### Header Files

- `cx28250.h`
- `eeprom.h`
- `ia_alarm.h`
- `ia_dumps.h`
- `ia.h`
- `ia_incs.h`
- `ia_linux.h`
- `ia_rs.h`
- `ia_types.h`
- `isar_alarms.h`
- `isar_api.h`
- `isar_build.h`
- `isar_codes.h`
- `isar_diag.h`
- `isar_linux.h`
- `isar_types.h`
- `rs8234.h`

- rs_dumps.h
- rs_log.h
- rs_oam.h
- rs_rsm.h
- rs_sch.h
- rs_seg.h
- rs_sep.h

## /usr/src/iphase/isaratm/src/

- apiRecv.c AAL5 sample receive only, one adapter, one VC
- apiRecvM.c AAL5 sample receive only, multi adapter, multiple VCs
- apiRecvRaw.c AAL0 sample receive only, one adapter, one VC
- apiSend.c AAL5 sample transmit/receive, one adapter, multiple VCs
- apiSendRaw.c AAL0 sample transmit/receive, one adapter, one VC
- config.c File parser support
- configs.c File parser support
- config_tool.c File parser
- ia_dumps.c Support function to dump driver structures
- isaralarm.c Sample test program for alarms
- isar_api.c Library file
- isarcbr.c CBR support file for isarxbr
- isar_close.c Support, sample VC close function, for isartool
- isar_codes.c Support functions for error codes, API names
- isar_conf.c Support, sample config function, for isartool
- isarconfig.c Configuration tool
- isar_diag.c Support, sample diagnostic functions, for isartool
- isar_dtin.c Support, sample user input function, for isartool
- isar_linux.c Library file
- isar_pav.c Support, sample parameter and value parser, for isartool
- isar_pckt.c Support, sample packet handler, for isartool
- isar_phy.c Support, sample phy API functions, for isartool
- isar_probe.c Support, sample probe/query function, for isartool
- isar_prom.c Support, sample serial eeprom function, for isartool
- isar_pvc.c Support, sample VC parameter parser, for isartool
- isartest.c Sample test program for all API functions
- isartool.c Sample engineering diagnostic tool
- isar_tune.c Support, sample RS tuning function, for isartool

- `isar_vc.c` Support, sample connection function, for `isartool`
- `isarxbr.c` Application to calculate VBR or CBR parameters
- `pvcs.c` Sample, support PVC parser and connection open function
- `rs_dumps.c` Support function to dump hardware structures

## Header Files

- `config.h`
- `cx28250.h`
- `eeprom.h`
- `ia_alarm.h`
- `ia_dumps.h`
- `ia.h`
- `ia_incs.h`
- `ia_linux.h`
- `ia_rs.h`
- `ia_types.h`
- `isar_alarms.h`
- `isar_api.h`
- `isar_build.h`
- `isarcbr.h`
- `isar_codes.h`
- `isar_conf.h`
- `isarconf.h`
- `isarconfig.h`
- `isar_diag.h`
- `isar_linux.h`
- `isar_pav.h`
- `isar_phy.h`
- `isartool.h`
- `isar_types.h`
- `isarxbr.h`
- `rs8234.h`
- `rs_dumps.h`
- `rs_log.h`
- `rs_oam.h`
- `rs_rsm.h`
- `rs_sch.h`

- rs_seg.h
- rs_sep.h

## Installing the ATM/API Driver on a PowerPC Platform

The Linux driver contained in this distribution is a partial open source release. The portion of the driver connecting the ***i***SAR ATM/API to the operating system is presented as open source. The portion of the driver implementing the ***i***SAR ATM/API is provided as a precompiled binary library.

To install the driver:

1. Insert and mount the distribution CD-ROM.
2. Change the directory to the Linux API driver directory:

   **cd /CDROM/drivers/api/linux/ppc**

   Where **CDROM** is the mount point for the CD.
3. Copy the driver package to a system directory and untar the distribution image:

   **tar -xvf SX00859-Xxx**

   Where **Xxx** is the revision number of the distribution.
4. An isar directory is created which contains both the open source and binary libraries. Change to this directory.

   **cd isaratm**
5. If you are using a cross compiler to build the driver for a target system whose platform differs from that which the compilation tools are installed, the following variables must either be edited in the supplied script 'doppc' or set and exported. The CROSS_COMPILE variable must be set to the path and prefix of the tools used to build the driver images. For example, if PowerPC cross compilation tools are being used on an Intel platform the CROSS_COMPILE variable would be set as follows:

   **CROSS_COMPILE=/usr/local/powerpc-linux/bin/powerpc-linux**

   **export CROSS_COMPILE**
6. The root of the Linux source tree must be exported to the operating system:

   **TOPDIR=/usr/src/linux**

   **export TOPDIR**
7. The architecture of the platform on which the module will be run must be exported to the operating system:

   **ARCH=ppc**

   **export ARCH**
8. The module can now be created by executing the following command:

   **make (if variable set and exported) or doppc (if modified)**

   You can also add the previous environment variables to the end of the make command:

```
make CROSS_COMPILE=/usr/local/powerpc-linux/bin/powerpc-linux-
TOPDIR=/usr/src/linux ARCH=ppc
```

9. The new driver module and following files are located in the `out` subdirectory. The following files should be copied to the target system, if necessary, and installed in the following directories:

   `isaratm.o - /lib/modules/<kernel version>/kernel/drivers/atm`

   `isarconf.0 - /etc/atm`

   `isartest - /usr/local/sbin`

   `isaralarm - /usr/local/sbin`

   `isarxbr - /usr/local/sbin`

   `isartool - /usr/local/sbin`

10. Update module dependencies once the module has been copied to `/lib/modules`:

    **`depmod -a`**

11. Load the module with the following command:

    For 2.4.x kernel versions:

    **`modprobe isaratm (load module and initialize with default parameters)`**

    **`or`**

    **`modprobe isaratm <isar_atm options> (reference Command-Line Options)`**

    **`or`**

    **`modprobe isaratm isar_atm_init=0 (load module only, no init)`**

    For 2.6.x kernel versions:

    Open the file `/etc/modprobe.conf` and find the line:

```
install isaratm /sbin/modprobe --ignore-install isaratm && /sbin/mknod-isaratm
```

    This entry loads the module, creates the node and initializes the adapter with default parameters. To load the module and initialize with user parameters reference *Command-Line Options* on page 32 and edit the line:

```
install isaratm /sbin/modprobe --ignore-install isaratm <options> && /sbin/mknod-isaratm
```

    To only load the module and defer the initialize process use the option `isar_atm_init=0`.

    Upon successful load of the driver, the following message will be displayed by the Linux operating system.

    `Warning: loading isaratm.o will taint the kernel: non-GPL license Proprietary`

## NOTE

**This warning is normal. The message indicates that this release is a partially open source distribution and contains a linkable binary library.**

12. For applications to communicate with the *i*SAR ATM/API driver, a device node must be created. A listing of registered devices can be seen by entering the following command:

    **cat /proc/devices**

13. There should be an "`isar`" entry for the 4576 adapter. The number preceding `isar` is the major number for this device. The following command can be used to create a device node:

    **mknod /dev/isar c <major number> 0**

## Installing the ATM/API Driver on Intel Equipment

The Linux driver contained in this distribution is a partially open source release. The portion of the driver connecting the *i*SAR ATM/API to the operating system is presented as open source. The portion of the driver implementing the *i*SAR ATM/API is presented as a precompiled binary library.

To start, you must prepare a Linux kernel source directory. The following instructions describe the method to prepare a RedHat Linux system:

1. If the following directories do not exist, create them:

   `/usr/src/${OS}`

   `/usr/src/${OS}/BUILD`

   `/usr/src/${OS}/RPMS`

   `/usr/src/${OS}/SPECS`

2. The system must have the kernel-source RPM installed. If this has been done, the system will have a `/usr/src/linux-X.X.X-X` directory. The build process uses `/usr/src/linux` as the kernel source root directory. Create the `/usr/src/linux` directory via a symbolic link:

   **ln -s /usr/src/linux-X.X.X-X /usr/src/linux**

3. Enter the following commands:

   **cd /usr/src/linux**

   **make mrproper**

4. If you have a single-processor Pentium II® system, enter the following:

   **cp configs/kernel-2.4.18-686.config .config**

   For a multi-processor Pentium II system, enter the following:

   **cp configs/kernel-2.4.18-686-smp.config .config**

5. Enter the following:

   **make oldconfig dep**

You are ready to build the binary RPM from the source RPM provided with your adapter.

To install the driver:

1. Insert and mount the distribution CD-ROM.

2. Change the directory to the Linux API driver directory:

   **cd /CDROM/drivers/api/linux/${CPU}**

   Where **CDROM** is the mount point for the CD.

3. Copy the driver package to system directory and untar the distribution image:

   **tar -xvf SX00yyy-xxx**

   Where **yyy** is the Interphase part number reference:

   850 = PPC

   860 = Redhat Intel i386

   960 = Montavista Intel x86_pentium4

   and where xxx is the revision number of the distribution.

   Two directories are created, 2.4 and 2.6. Change directories into the one for your system. Use the system command "**uname -r**" to get the machine kernel version.

4. The Intel version of the Linux *i*SAR ATM/API driver package uses the RedHat Package Manager (RPM) to build and install the driver. The source code package can be installed with the following command:

   **rpm -ivh isaratm-4.0-YY.XX.src.rpm**

   Where **YYY** is the host type and **XX** is the build number of the source RPM distribution.

   lppc = PPC

   lx86 = Redhat Intel i386

   mvista = Montavista Intel x86_pentium4

5. The installation of the source RPM copies the module source and a specification file to the /usr/src/${OS} directory. Change to this directory:

   **cd /usr/src/${OS}**

6. Use the RPM utility to build a binary package from the source with the following command:

   **rpmbuild -bb SPECS/isaratm.spec**

## *NOTE*

**The RPM utility will build the driver and create a binary package for this system as specified in the RPMS/i386 subdirectory. You will not be able to install the binary package created on this system on another system running a different version of the Linux kernel.**

7. Install the binary package with the following command:

   **rpm -ivh RPMS/${CPU}/isaratm-4.0-YY.XX.${CPU}.rpm**

The installation process copies the binary file to the `/lib/modules` directory, and installs the "`isarxbr`" utility required for the adapter's CBR and VBR configuration information

8. Update the modules dependences. This command should not produce any output. If instead it prints a list of unresolved references, you may have copied the wrong configuration file into `/usr/src/linux/.config` directory.

   **`depmod -a`**

9. Load the module with the following command:

   For kernel version 2.4.x

   **`modprobe isaratm`** (load the module with default parameters, enable IP)

   or

   **`modprobe isaratm <isar_atm options>`** (see *Command-Line Options* on page 32)

   or

   **`modprobe isaratm isar_atm_init=0`** (load module only, no IP)

   For kernel version 2.6.x

   Open the file `/etc/modprobe.conf` and find the line:

   ```
   install isaratm /sbin/modprobe --ignore-install isaratm && /sbin/mknod-isaratm
   ```

   This entry loads the module with default parameter.

   or add options (see *Command-Line Options* on page 32) for an entry that will load the module with the optional init parameters and enable IP.

   install `isaratm /sbin/modprobe–ignore -install isaratm OPTIONS && /sbin/mknod-isaratm`

   or add the `isar_atm_init=0` option for an entry the only loads the module. No IP operations are enabled and the user must initialize the adapter at a later time.

   install `isaratm /sbin/modprobe --ignore-install isaratm isar_atm_init=0 && /sbin/mknod-isaratm`

   Upon a successful load of the driver, the following message will be displayed by the Linux operating system:

   ```
   Warning: loading isar.o will taint the kernel: non-GPL license Proprietary
   ```

## *NOTE*

**This warning is normal. The message indicates that this release is a partially open source distribution and contains a linkable binary library.**

10. The installation of the binary package creates a directory tree, `/usr/src/iphase/isaratm`, which contains sample code for the creation of applications that use the ***i***SAR ATM/API. Change to this directory:

```
cd /usr/src/iphase/isaratm
```

The provided example utilities `apiRecv`, `apiRecvM`, `apiRecvRaw`, `apiSend`, `apiSendRaw`, `isarconfig`, `isartool`, `isartest`, and `isaralarm` can be built using the **`make`** command.

# Command-Line Options

The Interphase 4576 Linux *i*SAR ATM/API driver module supports several command-line options as listed below:

- **isar_seq_qsize**

  The `isar_seq_qsize` parameter allows you to set the transmit queue size. This parameter is the number of entries in the queue and must be set to 64, 256, 1024, or 4096. The default is 256.

- **isar_rsm_qsize**

  The `isar_rsm_qsize` parameter allows you to set the receive queue size. This parameter is the number of entries in the queue and must be set to 64, 256, 1024, or 4096. The default is 256.

- **isar_mtu**

  The `isar_mtu` parameter allows you to set the maximum CPCS-PDU size in bytes.

- **isar_sch_pcr**

  The `isar_sch_pcr` parameter allows you to set the default peak cell rate for a VC in cells per second. The default is 354,838 cells per second.

- **isar_sch_mcr**

  The `isar_sch_mcr` parameter allows you to set the default minimum cell rate for a VC in cells per second. This value must be a factor of the peak cell rate, `isar_sch_pcr`. The default is 354 cells per second.

- **isar_vci_bits**

  The `isar_vci_bits` parameter is the number of VCI bits the driver needs to use. This is a binary exponent value with a default of 12, 4096 VCIs. For example, to allow for the use of 1024 different VCIs, this value should be set to 10 (2^10 = 1204).

- **isar_vpi_bits**

  The `isar_vpi_bits` parameter is the number of VPI bits the driver needs to use. This is a binary exponent value with a default of 0, 1 VPI. For example, to allow for the use of eight different VPIs, this value should be set to 3 (2^3 = 8).

- **isar_trace**

  The `isar_trace` parameter enables verbose debugging.

- **isar_intr_tmr**

  The `isar_intr_tmr` and `isar_intr_cnt` parameters allow the user to throttle receive interrupts. The default value for the `isar_intr_tmr` parameter is 1024, which causes the adapter to interrupt if, after 1024 microseconds, less than `isar_intr_cnt` receives have occurred.

- **isar_intr_cnt**

  The `isar_intr_tmr` and `isar_intr_cnt` parameters allow the user to throttle receive interrupts. The default value for the `isar_intr_cnt` parameter is 8, which causes the adapter to interrupt when 8 receives have occurred.

- **isar_log_mask**

  The **`isar_alarm_mask`** command allows the user to enable notification of certain alarm events. Set the following bits to enable the indicated alarm:

```
ISAR_LOCD     BIT(18)    /* E   -Loss of Cell Delineation          */
ISAR_LOS      BIT(19)    /* E   -Loss of Sync                      */
ISAR_OOF      BIT(21)    /* E   -Out of Frame                      */
ISAR_LOF      BIT(22)    /* E   -Loss of Frame                     */
ISAR_LOP      BIT(23)    /* E   -Loss of Pointer                   */
ISAR_AIS_L    BIT(24)    /* E   -Alarm Indication Status - Line    */
ISAR_RDI_L    BIT(25)    /* E   -Remote Defect Indication - Line   */
ISAR_AIS_P    BIT(26)    /* E   -Alarm Indication Status - Path    */
ISAR_RDI_P    BIT(27)    /* E   -Remote Defect Indication - Path   */
```

  By default, `ISAR_LOS` and `ISAR_LOF` are enabled.

For example, to load the driver with larger transmit and receive queues, load the driver with the following command:

```
insmod isaratm isar_seg_qsize=1024 isar_rsm_qsize=1024
```

# VBR-CBR Support

Go to *VBR-CBR Support Utility on page 47* for VBR or CBR support functions.

# IPOA Operations 6

## Overview

This chapter explains how to enable and use IPOA functions for Solaris and VxWorks.

## Linux

The Linux version of the driver utilizes the system ATM and network stacks for operation (miniport), for information refer to http://linux-atm.sourceforge.net or http://linux.org or Interphase customer service.

When opening a connection for QOS modes of CBR, VBR (single leaky bucket) or VBR23 (dual leaky bucket) the following details which socket structure entries are converted to an API connection open structure element.

If `vcc->qos.txtp.traffic_class` equals `ATM_CBR`, then

```
isar_vcc_t.uv.open.mode = ISAR_CBR;
```

and set cells per second with:

```
isar_vcc_t.uv.open.xbr1 = vcc.qos.txtp.pcr;
```

If `vcc->qos.txtp.traffic_class` equals `ATM_VBR`, then

if `vcc->qos.txtp.min_pcr` equals `0`, then

```
isar_vcc_t.uv.open.mode = ISAR_VBR;
```

else

```
isar_vcc_t.uv.open.mode = ISAR_VBR23;
```

and set the exponent/mantissa formatted, from the application `isarxbr`, peak and sustained cell rates with:

```
isar_vcc_t.uv.open.xbr1 = vcc.qos.txtp.pcr;
isar_vcc_t.uv.open.xbr2 = vcc.qos.txtp.min_pcr;
```

## Solaris

### Introduction

The Interphase driver, `SX00861-xxx`, supports an API interface and an IPOA interface. There are three basic steps to set up and configure for IPOA operation; initialization of the driver, opening the desired connections, and configuring the network layer.

## Adapter Initialization

When initializing the adapter, with the API **ISAR_INIT**, there are two fields to set up; isar_ctrl_t.uc.init.flags and isar_ctrl_t.uc.init.addr[8].

### Flags

Set the CLIPOA enable bit, BIT(3), into this field.

### Addr[8]

This field holds two values, IP address and netmask, at offset 0 and 4 respectively. These values are used by the API **ISAR_VC_OPEN** in cases where the SPA (Source Protocol Address) is empty in the isar_vcc_t.uo.open structure. Thus, these values represent the default or primary IP address for the adapter.

## Connection Open

When opening a VPI/VCI connection for IPOA operation there are two fields to set; isar_vcc_t.uo.open.enable and isar_vcc_t.uo.open.addr[8].

### Enable

Set the ISAR_VCC_IPOA enable bit, BIT(2), into this field. This informs the driver where to route the traffic data.

### Addr[8]

This field holds two values, SPA (source address) and TPA (target address), at offset 0 and 4 respectively. If the SPA is NULL then it is filled in by the driver with the SPA from the initialization structure. The TPA is the IP address for the other end of the connection.

## Network Configuration

There are two basic steps to setting up the network interface. In the following steps #bd represents the adapter number.

1. **ifconfig isar#bd plumb**
2. **ifconfig isar#bd SPA netmask 0xffffff00 up**

The interface is now ready to transmit and receive data via the IP stack. There are two sample programs available for reference that use the socket interface for UDP send and receive functions.

## Adding a Route

When adding a route to a PVC the user can either use the supplied script, `isarRoute`, or issue the API and configure the upper layer separately. The basic premise is that `apiRoute` will add an IP and mask to a PVC connection list. When a transmit request is handled by the driver, it attempts to find the destination IP in its hash table. This lookup is based on the following algorithm:

```
for(x = 0; x < connection_count; x++)
  if(dest_ip & pvc[x].hash.mask) == (pvc.hash.ip))
     vcid = pvc.vcid;  /* hash entry found */
```

### isarRoute

This script will issue the API to the driver and issue the upper layer command. There are four functions that the script will handle: add , delete, show, and help. The argument list is identical to the system function, route. It simply takes the standard route arguments, issues them to the host, and then converts them for the API function and issues the command to the driver.

**add** - usage: `isarRoute add <net | host>  addr gateway`

The netmask for to be used by the driver for IP to VPI/VCI hash lookup is derived from the route address, addr.

`addr = x.x.0.0` will produce a netmask of 255.255.0. 0

`addr = x.x.x.0` will produce a netmask of 255.255.255.0

`addr = x.x.x.x` will produce a netmask of 255.255.255.255

The gateway address is the TPA (target IP address) of the PVC the route is to be added to.

**delete** - usage: `isarRoute delete <net | host> addr gateway`

This has the same parameter usage as the add function.

**show** - usage: `isarRoute show gateway`

This function only issues an API command to the driver, no upper layer function, and will display all IP routes in the driver for a PVC (gateway). Entry 0 is always the assigned TPA from the API connection open function.

Entry `Address::NetMask`

`0` 1.1.1.1 255.255.255.255

`1` 2.2.2.0 255.255.255.0

**help** - usage: `isarRoute` help

usage:"

```
echo "     : isarRoute add <net | host> addr(1.1.1.1) gateway(2.2.2.2)"
```

```
echo "     : isarRoute delete <net | host> addr(1.1.1.1) gateway(2.2.2.2)"
echo "     : isarRoute show gateway(2.2.2.2)"
echo "     : net = 255.255.0.0 mask, host = 255.255.255.0 mask"
echo "     : addr = destination IP , gateway = VCC's target IP"
```

# VxWorks

## Introduction

This document describes the operations necessary to enable IPOA functionality for the ISAR 4576 product on VxWorks version 5.4 or greater. The first section covers the operation using the application `isarconfig` that is supplied in the driver package. The second section covers the operation for a user-written application.

### *NOTE*

**All applications must include the package file `isar_board.c`. This file contains a structure defining BAP parameters and a function: `_vxipld` that can be called to configure the IP interface.**

The nomenclature used to associate adapter numbers to interface names is alpha in nature. Thus, interfaces for adapter 0 is "`isarA`", adapter 1 is "`isarB`", etc., followed by a numeric reference to the unit number for that interface. Thus, the first interface on adapter 0 would be "`isarA0`".

## General Flow of Events

The following is an outline of the basic set of events, and the order in which they occur, to bring up IPOA functionality.

1. Initialize the adapter with the CLIPOA enabled, an IP address, an IP net mask, and set the number of IP buffers to pre-allocate (see *Adapter Initialization* on page 36).
2. Load the driver into the END mux with the system call **`muxDevLoad`**. Example for adapter 0, net unit 0:

   ```
   END_OBJ *pEnd = muxDevLoad ( 0, isar_EndLoad, "10.1.1.2", 1, NULL);
   ```

3. Start the END mux operation with the system call **`muxDevStart.`**

   ```
   muxDevStart(pEnd);
   ```

4. Attach a protocol to the END mux entity with the system call **`ipAttach`**.

   ```
   ipAttach (0, "isarA");
   ```

5. Set the net mask for the protocol interface with the system call **`ifMaskSet`**.

   ```
   ifMaskSet ("isarA0", 0xffffff00);
   ```

6. Set the address for the protocol interface with the system call **ifAddrSet**.

   **ifAddrSet ("isarA0", "10.1.1.2");**

7. Open VPI/VCI connections in the driver with the ISAR_VCC_IPOA and ISAR_ARP_OFF bits set in the enable field and the spa/tpa fields filled in the source and target IP addresses for the connection.

## Operations Using Application isarconfig

This section assumes that the user will be using the supplied application, **isarconfig.o**, to initialize the adapters and load the IPOA driver entries into the END mux of VxWorks.

Two file types will be used to hold the initialization and configuration parameters for **isarconfig.o** to parse; isarconf.x, ipconf.yy. (Where x is the adapter number, and yy is the adapter number and net unit number.) The files are to be located in /etc/atm. (Refer to the isarconf.0 file in the package for more details).

***NOTE***

**Be sure to set IP_MAX_UNITS to number of interfaces to be used, remember that the on board Ethernet counts for one, in the BSP.**

### isarconf.x

When initializing the adapter for IPOA functions there are four entries to be referenced in the file for parsing into an isar_ctrl_t structure. These entries are:

- LINK 0 # adapter number to initialize
- CLIPOA enable # 0/off/disable, 1/on/enable
- IPaddress 10.1.1.2 # IP (source protocol) address
- IPnetmask 255.255.255.0 # IP netmask

The IPaddress and IPnetmask will be used for a VC opened that does not specify an SPA (Source IP Address).

### ipconf.yy

When loading an IPOA entity into the END mux the **isarconfig.o** application needs three parameters; adapter/netunit number, IP address, and IP net mask. These entries are:

- LINK 0x00000000 upper 16-bit = adapter number, lower 16-bit = net unit
- IPaddress 10.1.1.2 # IP (source protocol) address
- IPnetmask 255.255.255.0 # IP netmask

## Initialization

To initialize an adapter with **isarconfig.o** use the following syntax:

Adapter 0

```
Prompt> isarconfig "-l0 -f/etc/atm/isarconf.0 init"
```

Adapter 1

```
Prompt> isarconfig "-l1 -f/etc/atm/isarconf.1 init"
```

## Loading END mux Entity

To load an instance into the END mux with **isarconfig.o** use the following syntax:

Adapter 0, unit 0....ipconf.00 - LINK = 0x00000000, IPaddress = 10.1.1.2, IPnetmask = 255.255.255.0

```
Prompt> isarconfig "-l0 -f/etc/atm/ipconf.00 ipld"
```

Adapter 0, unit1....ipconf.01 - LINK = 0x00000001 IPaddress = 10.1.2.2, IPnetmask = 255.255.255.0

```
Prompt> isarconfig "-l0 -f/etc/atm/ipconf.01 ipld"
```

Adapter 1, unit 0....ipconf.10 - LINK = 0x00010000 IPaddress = 10.1.3.2, IPnetmask = 255.255.255.0

```
Prompt> isarconfig "-l1 -f/etc/atm/ipconf.10 ipld"
```

Adapter 1, unit 1....ipconf.11 - LINK = 0x00010001 IPaddress = 10.1.4.2, IPnetmask = 255.255.255.0

```
Prompt> isarconfig "-l1 -f/etc/atm/ipconf.11 ipld"
```

At the successful completion of each load the following will be displayed:

```
Adapter 0, unit 0


_vxipld complete...
muxName          : isarA
netName          : isarA0
ipAddr           : 10.1.1.2
ipMask           : 255.255.255.0
```

Alpha characters represent the adapter number where `A` = adapter #0, `B` = adapter #1, etc.

## Open a Connection

When opening connections with **isarconfig.o** there are a number of parameters to define for parsing in order to fill in the `isar_vcc_t` structure for the API. The following is an example for opening connections on adapter #0.

```
Prompt> isarconfig "-l0 -f/etc/atm/isarconf.0 pvcs"
```

The following is a simple example entry in `isarconf.x` for UBR connection on adapter#0 using the SPA from the initialization.

```
# IPoA target=10.100.1.4 no-ARPs, AAL5, UBR

<PVC>

vpi=1 vci=110 type=AAL5 mode=UBR tqt=0 xbr1=0 xbr2=0 ipoa=on tpa=10.1.1.4
arpnot=On enable=On

</PVC>
```

This example is also for adapter#0 but uses a different SPA, still encompassed by the netmask for that interface.

```
# IPoA target=10.100.1.4 no-ARPs, AAL5, UBR

<PVC>

vpi=1 vci=110 type=AAL5 mode=UBR tqt=0 xbr1=0 xbr2=0 ipoa=on spa=10.1.1.3
tpa=10.1.1.5 arpnot=On enable=On

</PVC>
```

## Operations Using the API

This section assumes that the user will be using their own application to initialize and load the END mux, with `isar_board.c` included in the application. There is a function, **`_vxipld()`**, in `isar_board.c` for reference or usage if an application chooses to use `_ux.ipldl` it simply needs to declare a character array of name `fname`, define `IPOA_KERNEL`, and follow the syntax rules outlined in the previous section for using `isarconfig` and file `ipconf.yy` otherwise, reference the pseudo code below. (Note that **`_vxipld`** assumes the controller is initialized and obtains the parameters from a file.)

### Initialization

To initialize the adapter for IPOA operations, there are three structure elements to modify along with all the normal structure modifications. These are as follows:

- `isar_ctrl_t->uc.flags |= CLIPOA;  /* or in the clipoa bit to the flags field */`
- `*(ip_addr_t *)&(isar_ctrl_t->uc.addr[0]) = 0x0a010102; /* set global IP address in hex */`
- `*(ip_addr_t *)&(isar_ctrl_t->uc.addr[4]) - 0xffffff00; /* set the global net mask`, currently not used */

Now issue the **`API`**, **`ISAR_INIT`**, with **`isar_msg(&isar_ctrl_t)`**.

### Loading END mux Entity

To load an entity into the END mux for VxWorks there are five basic steps, two for the mux and three for the IP interface. To support this operation there are nine entities involved as follows:

- `baseName` – "isar"
- `baseId` – 'A'
- `ifName` – Name to attach to the mux with

- `netName` – Name to use for the interface
- `isarMuxDevName` – External character array used by the driver when first called by the mux
- `hwUnit` – Adapter number
- `netUnit` – Unit number relative to the adapter/mux interface
- `ipAddr` – IP address
- `ipMask` – Net mask

The following shows a typical sequence in pseudo code:

```
extern char isarMuxDevName[];
extern END_OBJ * isar_EndLoad (char *initString, void *dummy);
END_OBJ *pEnd;
struct in_addr iaddr;

        /* format the unit value for isar_EndLoad
         * upper 16bits = adapter#, lower 16bits = net number
         */
        unit = hwUnit;
        unit <<= 16;
        unit |= netUnit;

        sprintf (ifName, "%s%c", baseName, (hwUnit + baseId));
        sprintf (isarMuxDevName , "%s%c", baseName, (hwUnit + baseId));
        sprintf (netName, "%s%c%d", baseName, (hwUnit + baseId), netUnit);

        iaddr.s_addr = ipAddr;
        inet_ntoa_b (iaddr, ipstring);

        /* add entity to the END mux
         * unit = hwunit-netunit
         * ipstring = ip address
         */
        pEnd = muxDevLoad(unit, isar_EndLoad, ipstring, 1, NULL);

        if (pEnd == NULL) {
                printf(" muxDevLoad failed for device entry!\n");
                return;
        } else {
```

```
        if (muxDevStart(pEnd) == ERROR) {
        printf(" muxDevStart failed for device entry!\n");
        return;
        }
}

if (ipAttach(netUnit, ifName) != OK) {
       printf(" ipAttach failed for %s-%d\n", ifName, netUnit);
} else {

       /* Set Ip addr and mask */

       rtn = ifMaskSet(netName, htonl(ipmask));

       if (rtn) {
       printf("_vxipld: ifMaskSet for %s:%x failed\n", ifName, ipmask);
       }

       rtn = ifAddrSet(netName, ipstring);

       if (rtn != 0) {
       printf("ifAddrSet for %s:%s failed\n", ifName, ipstring);
       return;
       }
}
```

## Open a Connection

In addition to the normal parameters for opening a VPI/VCI connection the following are needed for IPOA connections:

- `MEM_COPY((char *)&isar_vcc_t->uv.open.addr[0], (char *)spa, 4);`
- `MEM_COPY((char *)&isar_vcc_t->uv.open.addr[4], (char *)tpa, 4);`
- `isar_vcc_t.uv.open.enable |= (ISAR_VCC_IPOA | ISAR_ARP_OFF);`

Now issue the API, **`ISAR_VCC_OPEN`**.

## Adding a Route

There are three API functions for IP routing: add, delete, and info. To add a target route to an existing PVC there are two mechanisms to use. Either issue the API within an application or use the **`apiRoute`** application. The two primary fields are the address for the

route and the mask to use for the IP to VPI/VCI hash table lookup. If the mask is 255.255.0.0 and the addr is 1.1.x.x then all frames for IP address 1.1.x.x will go out over this PVC. (Refer to *API Guide* on page 67 for full details.)

## apiRoute Application

The `apiRoute` application takes three arguments: `command`, `route_addr`, `gateway`.

`command` — `0` = add, `1` = delete, `2` = show

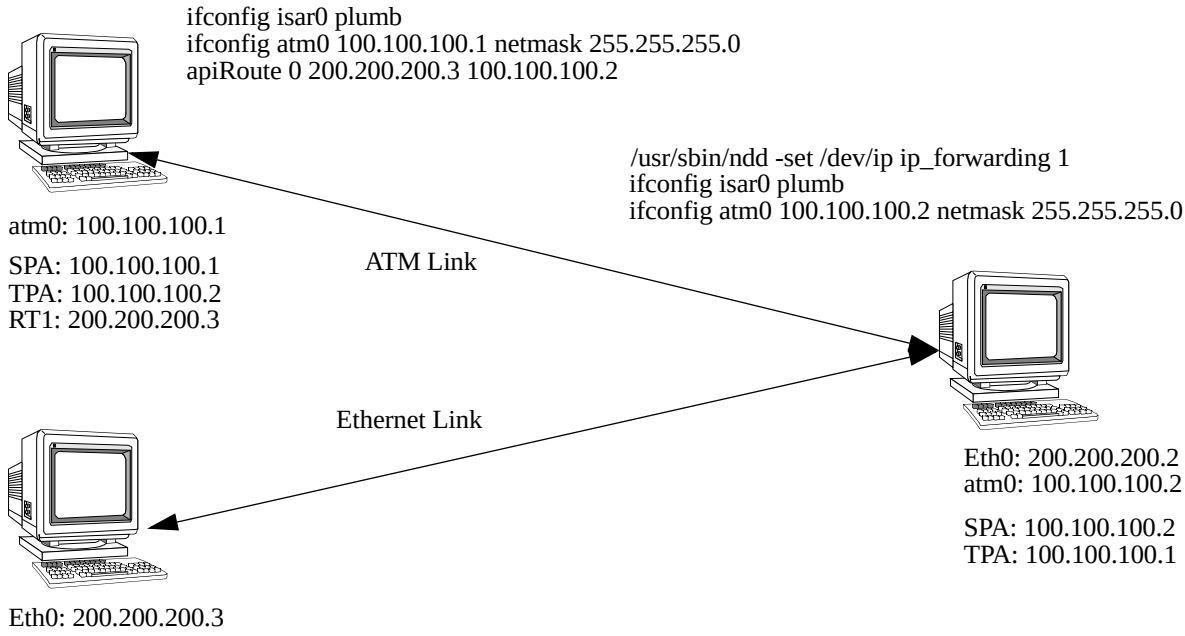
`route_addr` — IP address to add or delete

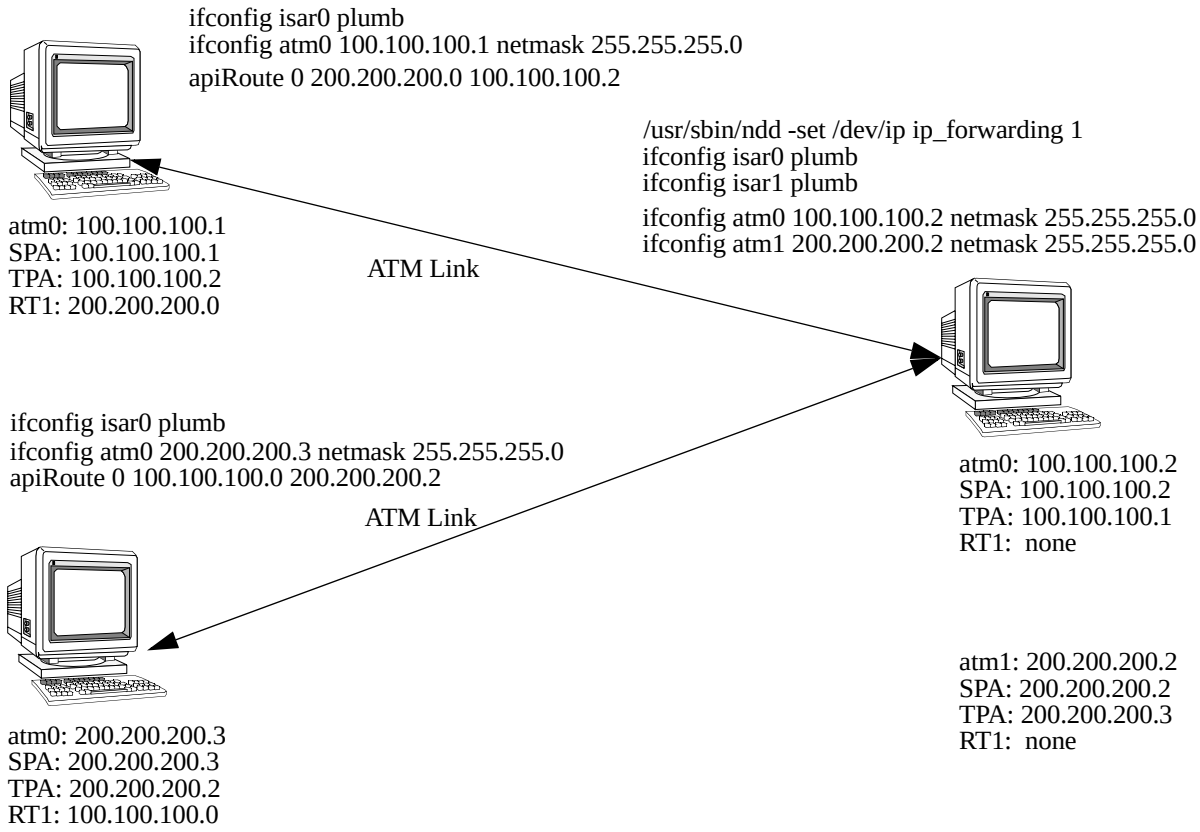
`gateway` — Assigned target IP address of the connection to add or delete from.

The mask field is implied from the `route_addr`, if `route_addr` is `1.1.0.0` the mask is 255.255.0.0.

The application will search through all available adapters to find the `vcid` associated with the requested gateway and then issue the API.

# Example Diagram



# VBR-CBR Support Utility 7

## Overview

The `isarxbr` utility is used as a set up tool or guideline for some driver initialization parameters concerning CBR operations and for facilitating floating point operations for VBR concerned with deriving values needed for opening a connection.

### CBR

With the flexibility of the RS8234 SAR interface, there are many possible configurations associated with determining cell line rate, rate resolution, and fitting various rates in the hardware scheduling table. The **`isarxbr`** utility can be used prior to driver initialization to determine the proper values for the `sch_pcr` and `sch_mcr` values in the control structure and to verify that the desired rates will actually be accepted by the driver (that they will fit in the hardware table).

#### Interactive Mode (menu)

Once the `isarxbr` utility is executed, the application provides a main menu as follows:

```
Main menu
1 - VBR Functions
2 - CBR Functions
x - exit menu
```

Select **`2`** for CBR:

```
CBR Functions Menu...
1 - Create CBR Model
2 - Compute CBR Conf params
x - exit menu
```

Select `1` to set up the basic requirements for the computation. The current parameters are first displayed followed by the request to edit the parameters. A return through a field will maintain the current value and a "." will exit the editing function.

The basic intent of the application is to find appropriate values that will allow the selected rates to fit in the hardware scheduling table and use the following algorithm, for all PCRs, for all MCRs up to the lowest rate, for all allowable RDVs on the rates, to find the best fit. The hierarchy of best fit is the number of exact rate matches and the number of exact fits.

#### Parameter Descriptions

##### PCR

This is the Peak Cell Rate. If the UBR line cell rate is not critical then a value of 1 may be entered, allowing the application to look through all three possible maximum line rates.

### MCR

This is the minimum cell rate or rate resolution for all rates. The application will start with this value and limit it to the lowest rate selected.

### RDV

This is the rated deviation factor. If it is critical to achieve an exact rate, then set this value to 0. Otherwise the application will use this value as a rate modification limit, on the low side, while trying to find a fit (limit = rate - (rate/rdv);). Thus, a request for a rate of 14000 cells per second may return a value of 13850 cells per second to be used in the VCC open.

### Verbose

This enables verbose printing of the calculation process if set to 1.

### Rate 0-8

These are the user desired rates. Rate[0] must be the lowest rate.

In the following display example, the PCR has been limited to 354838 and the cell rate resolution has been set to 50 cells per second with a rate deviation factor of 10%.

Current Parameters:

```
PCR (351063, 354838, 358695, or 1 (try all of them))
         PCR = 0              cells/sec
         MCR = 40             cells/sec
         RDV = 10             % of c/s
     Verbose = 0              printf
     Rate[0] = 0              cells/sec
     Rate[1] = 0              cells/sec
     Rate[2] = 0              cells/sec
     Rate[3] = 0              cells/sec
     Rate[4] = 0              cells/sec
     Rate[5] = 0              cells/sec
     Rate[6] = 0              cells/sec
     Rate[7] = 0              cells/sec
```

Edit CBR Parameters:

Rate[0] must be the lowest rate

```
         PCR:  354838
         MCR:  50
         RDV:  10
     Verbose:  1
     Rate[0]:  300
     Rate[1]:  1200
     Rate[2]:  6000
     Rate[3]:  .
```

CBR Functions Menu:

```
1 - Create CBR Model
2 - Compute CBR Conf params
x - exit menu
```

Select 2 to begin the calculation process.

```
Calculating CBR values...

  pcr    mcr clk table #fit #match #rate
351360    72  93  4880   3     0    18 ***
352800    50  93  7056   3     2    43 ***
352800    60  93  5880   3     2    46 ***
352800   100  93  3528   3     2    34
353400    50  93  7068   3     1    22
353400    60  93  5890   3     1    14
353400    75  93  4712   3     1    14
353400   100  93  3534   3     1    14
353400   150  93  2356   3     1    10
354000    50  93  7080   3     3    30 ***
354000    60  93  5900   3     3    16
354000    75  93  4720   3     3    18
354000   100  93  3540   3     3    22
354000   150  93  2360   3     3    14
-
SchPCR = 354000
SchMCR =     50
Rate#    User Rate  Calculated  TableCheck
 0             300         300        Pass
 1            1200        1200        Pass
 2            6000        6000        Pass

All Available Rates
   100,    150,    200,    250,    300,    400,    500,    600,
   750,   1000,   1200,   1500,   2000,   2950,   3000,   5900,
  6000,   8850,  11800,  14750,  17700,  23600,  29500,  35400,
 44250,  59000,  70800,  88500, 118000, 177000,
```

The first table that gets displayed shows any computations that at least found a fit for all, possibly adjusted, rates. The `***` represent the last best fit. The values printed for `SchPCR` and `SchMCR` are to be used when initializing the driver. The values in the second table for `Calculated` are to be used when opening a connection. The list at the end is a summary of all available rates using this configuration that will not produce any CDV (Cell Deviation Variance), due to not having an exact fit in the hardware scheduling table.

## Program interface

The **`isarxbr`** CBR function can also be included in an application, source and header are provided in the package (`isarcbr.c` and `isarcbr.h`). The same structure is used to pass receive parameters and return computed values.

```
#define MAX_RATES   8

typedef struct cbr_entry_s {
          U32    uRate;       /* input - user desired rate           */
          U32     cRate;      /* output - application calculated rate */
          U32     tblchk;     /* cbr table check status, 0 = pass or fit */
} cbr_entry_t;

typedef struct calc_cbr_s {
          U32    sch_pcr;     /* input  = line cellrate, if null then
354838 */
                              /* output = isarconf value                */
          U32     sch_mcr;    /* input  = min cellrate , if null, then 40
*/
                              /* output = isarconf value                */
          U32     rdv;         /* input  = rate deviation factor
*/
          U32     numUser;    /* input  = number of user rates in table
*/
          U32    flag;         /* input  = non-zero turns on verbose mode
*/
          cbr_entry_t cbre[MAX_RATES]; /* input  = user requested rates
*/
                              /* output  = holder for calculated rates*/
} calc_cbr_t;
```

Fill in the structure and call `isarcbr_calc` with prototype:

```
U32 isarcbr_calc(calc_cbr_t *uct)
```

On return, verify that all supplied rates have a 0 in the `cbre[x]->tblchk` fields. If so, then use the values in the `cbre[x]->cRate` fields when opening connections.

## VBR

Due to floating point calculations, the driver requires setting the `SCR`, `PCR`, and `MBS` for a virtual circuit, this release of the driver requires the use of an application to obtain these values. The **`isarxbr`** utility can be utilized to configure the driver.

Once the **`isarxbr`** utility is executed, the application provides a main menu as follows:

```
Main menu
1 - VBR Functions
2 - CBR Functions
x - exit menu
```

Select **`1`** for VBR:

```
VBR Functions menu...
1 - Create Hardware Model
2 - Compute VBR I/L params
x - exit menu
```

Select **`1`** to create a Hardware Model. The Create Hardware Menu is displayed follows:

```
Create Hardware Menu...
q - query driver
d - default hw configuration
e - edit hw configuration
s - show hw configuration
x - exit menu
```

Select **d** to create a default hardware configuration. The utility displays the following. If the system has the API driver, select the **q** option to retrieve the configuration from the driver (which must already be initialized). Any change to the hardware configuration parameters will affect the xbr calculations.

```
Default hw configuration...

systemClock:      33000000       hz
slotPeriod:       94             clks/slot
tableSize:        2112           slots
txFIFOLen:        9              cells
CDVT:             0.000035       secs
totalFIFOLen:     11             cells
lineCellRate:     351063.843750  cells/sec
```

Select **x** to return to the Main Menu. Select **2** to compute VBR I/L parameters. The following menu is displayed:

```
Compute VBR IL Values...
1 - VBR1 single leaky bucket
2 - VBR2 dual leaky bucket
x - exit menu
```

A VBR1 example follows. Select **1** to compute the xbr1 value for the driver. When prompted, enter the PCR value to set up for a virtual circuit. The utility will calculate the xbr1 value needed to pass to the API in the isar_vc_open() message. Note that xbr2 is not required for this VBR mode.

```
Current VBR1 params...
          PCR:  0               cells/sec

Edit VBR1 params...
          PCR:  60000
          I(PCR) = 5.851064
          L(PCR) = 1.287235
          IExp =  12 (0xc)
          IMan = 237 (0xed)
          LExp =  10 (0xa)
          LMan = 148 (0x94)
          xbr1 = 0x852518ed
```

A VBR2 example follows. Select **2** to compute xbr1 and xbr2 values for the driver. When prompted, enter the required PCR, SCR. The utility calculates the xbr1 and xbr2 values to send in the isar_vc_open() message.

```
Current VBR2 params...
          PCR:  0               cells/sec
          SCR:  0               cells/sec
          Burst:  0               cells
```

```
Edit VBR2 params...
          PCR:  60000
          SCR:  14000
          Burst:  255

I(PCR) = 5.851064
L(PCR) = 1.287235
IExp =  12 (0xc)
IMan = 237 (0xed)
LExp =  10 (0xa)
LMan = 148 (0x94)
xbr1 = 0x852518ed

I(SCR) = 25.075989
L(SCR) = 4884.417969
IExp =  14 (0xe)
IMan = 291 (0x123)
LExp =  22 (0x16)
lMan = 99 (0x63)
xbr2 = 0x8b18dd23
```

# Adapter Quick Test 8

## Overview

Included in this distribution is an open source, user mode application named `isartest` along with a binary version. The `isartest` executable can be built when the user compiles the API driver.

The application allows the user to test all functionality contained in the driver feature set. `isartest` is presented as open source to provide the user with a working example of code that interfaces with the API driver.

Although the application's usage is documented in *ATM isartest Guide on page 115*, the following example allows quick testing of the adapter's transmit and receive functionality without requiring an in-depth knowledge of the application usage.

### *NOTE*

**`isartest` is designed for testing against another station or adapter running `isartest` or loopback. If this is not the case, then refer to setting the raw receive flag in the parameter section in *ATM isartest Guide on page 115*.**

1. Connect a loopback cable to the 4576 adapter.
2. Execute the application by entering the following:

   **`isartest`**

   The following is displayed on the console, where for Version, `x` is the major release level, y is the minor release level, `z` is the platform and `xx` is the build number:

   ```
   Enter board number: 0
   Set up Log File (y/n)? y
   filename: log
   isartest for 4576 Version x.y.z.xx
   SunOS moon 5.8 Generic_108528-02 sun4u sparc SUNW,Ultra-30 4576
   isartest for 4576 Version x.y.z.xx
   [a]larm [c]ontrol [d]isplay [h]elp [o]am [p]hy [P]arams [q]uery
   [t]ests [v]c [x]it
   ```

3. Prior to testing the capabilities of the adapter, you must initialize the adapter with default configuration values. Select **`c`** for the Control option and press enter.

   The following is displayed on the console:

   ```
   Control API
   [i]nit [l]oopback [r]sTune [t]race [u]nld:
   ```

4. Select the **`i`** option to initialize the adapter and press enter.

   The following is displayed on the console:

```
[d]efault [e]tc/atm/isarconf.0 [f]ile : e
```

The following information is displayed on the console:

```
Display parameters (y/n)? : n
Adapter not yet initialized...setting up info
isar_control(513) initialize link  passed
isartest for 4576 Version x.y.z.xx
[a]larm [c]ontrol [d]isplay [h]elp [o]am [p]hy [P]arams [q]uery [t]ests [v]c
[x]it)?
```

5. Select the **v** option to configure a VC and press enter.

   The following is displayed on the console:

```
[a]ctvc [c]lose [d]isable [e]nable [i]nfo [o]pen [s]tats [t][T]x [x]it:
```

6. Select the o option to open a VC and press return.

   The following information is displayed:

```
[e]tc/atm/isarconf.0 [f]ile [m]anual :
```

7. Enter **m**.

   The following information is displayed:

```
[e]tc/atm/isarconf.0 [f]ile [m]anual : m
Current parameters:
QOS: UBR
TYPE: AAL5
PCR: 351063
MCR: 0
TQD: 0
ENABLE NOW - NO OAM LOCK....change(y/n)?
Base VPI: 0
Count   : 1
Base VCI: 32
Count   : 1
VCOPEN
vcopen: vpi 0 vci 32 vcid=0x300123fa000 status=0x0
   : mode 1 type 5 xbr1 0 xbr2 0
-------------------------------------------------
VCC Menu
[a]ctvc [c]lose [d]isable [e]nable [i]nfo [o]pen [s]tats [t][T]x
[x]it:
```

8. Enter **x**

9. Select **t** to test the adapter's transmit and receive capabilities.

   The following information is displayed:

```
TEST SELECTIONS
Use the upper case selection for Verbose
[b][B] tx   xMB on all vc's of frame 1024
```

```
[c][C] echo xMB on all vc's of frame 1024
[d][D] tx   xMB on random vc's of frame 1024
[e][E] echo xMB on random vc's of frame 1024
[o][O] open-send-close
[v][V] vc disable/enable
```

10. Select **c** and answer the subsequent questions as the following displays:

```
Set Transfer size
i.e 16KB=16384, 32KB=32768, 64KB=65536
i.e 128KB=131072, 256KB=262144, 512KB=524288
i.e 1MB=1048576, 10MB=16777216, 100MB=268435456
1048576
do data (y/n)?y
frame size [f]ixed, [i]incr [r]andom : f
------------------Test Parameters-----------------------------
Host:  Bd#0 SunOS lab-dhcp-104 5.9 Generic sun4u sparc SUNW,Ultra-60 4576
Target:Bd#0 SunOS lab-dhcp-104 5.9 Generic sun4u sparc SUNW,Ultra-60 4576

Base Frame Size: 1024 - Fixed
Data Compare   : Enabled
Test Data Size : 1048576...0x100000
Duplex Mode    : Round Trip - Echo
VC Count       : 1
VC Selection   : Sequential

------------------Test Results---------------------------------
Time: 0.164
        TxB         RxB       Total     MBsec   Mbsec   IOsec
     1048576     1048576     2097152  12.787      96   12080

        TxFrm   TxOam   TxRty   Flow    RxFrm   RxOam   RxDrp   BadData
Target  1024    0       0       0       1024    0       0       0
Host    1024    0       0       0       1024    0       0       0

Display the individual connection results (y/n)?n
```

The above results should not show anything except `TxFrm` and `RxFrm` values and these should all be identical at 1024.

# API Library Functions 9

## Introduction

This chapter contains the library function calls (entry points) in the ISAR API. These entry points provide configuration management and the ability to transmit and receive data, packets, and OAM cells.

For all but the support functions, there is a generic structure type containing a header followed by a set of function-specific structures enclosed in a union. With this structure type there is only one entry point needed for the API to deliver messages and return information and status. The return status from the entry point is simply for the transport function. The status for the specific function is returned in the supplied structure header.

### *NOTE*

**For VxWorks, `isar_board.c` must either be included in all applications, or the application must provide the BSP structure defined in the file.**

The following library function is provided:

- int isar_msg (void *msg)

The following API groups for `isar_msg` are provided:

- `ISAR_QUERY`
- `ISAR_CTRL`
- `ISAR_VCC`
- `ISAR_OAM`
- `ISAR_ALARM`
- `ISAR_PHY`

The following user callback functions are provided:

- isar_rcv_ind (void *ind)
- isar_oam_ind (void *ind)
- isar_alarm_ind (void *ind)

The following library support functions are provided.

- char *err2name (int code);
- char *err2desc (int code);
- char *err2str (int code);
- char *ctrl2name (int code);
- char *ctrl2desc (int code);
- char *ctrl2str (int code);

- char *vcc2name (int code);
- char *vcc2desc (int code);
- char *vcc2str (int code);
- char *oam2name (int code);
- char *oam2desc (int code);
- char *oam2str (int code);
- char *qury2name (int code);
- char *qury2desc (int code);
- char *qury2str (int code);
- char *alrm2name (int code);
- char *alrm2desc(int code);
- char *alrm2str (int code);
- char *phy2name (int code);
- char *phy2desc (int code);
- char *phy2str (int code);

# Usage

The recommended sequence for management tasks using the library functions is as follows:

1. Device discovery and initialization.
2. Establishing connections.
3. Moving data.
4. Query statistics and status.
5. Shutting down.

## isar_msg

NAME

`isar_msg` – send a message to the driver and return status.

**SYNOPSIS**

#include <`isar_api.h`>

#include <isar_alarms.h>

#include <isar_phy.h>

```
int isar_msg(void *msg);
```

**DESCRIPTION**

The `isar_msg()` function delivers a message to the driver and if appropriate, returns information from the adapter, driver, library, or connection such as configuration, state, and statistics.

**INPUTS**

The input data structure is defined by the API group, all of which contain a generic header to specify the API command, link number, byte count and status. The following groups and associated types are available:

| GROUP | TYPES | DESCRIPTION |
|---|---|---|
| **QUERY** | isar_query_t | |
| | ISAR_LIB_INFO | Library info |
| | ISAR_DRVR_INFO | Driver info |
| | ISAR_ADAP_INFO | Adapter info |
| | ISAR_PORT_INFO | Adapter/port info |
| | ISAR_LINK_INFO | Link info |
| | ISAR_LINK_STATS | Link statistics |
| | ISAR_RS_CONF | RS8234 configuration |
| **CTRL** | isar_ctrl_t | |
| | ISAR_INIT | Initialize, configure driver |
| | ISAR_TRACE_ENABLE | Enable verbose logging |
| | ISAR_TRACE_DISABLE | Disable verbose logging |
| | ISAR_LPBK_ENABLE | Enable loopback |
| | ISAR_LPBK_DISABLE | Disable loopback |
| | ISAR_RS_TUNE | Modify interrupt interface |

| | | |
|---|---|---|
| | ISAR_UNLOAD | Release all resources, reset |
| | ISAR_CLOSE | Close this application's receive resources and library thread |
| **VCC** | isar_vcc_t | |
| | ISAR_VCC_OPEN | Open a connection |
| | ISAR_VCC_CLOSE | Close a connection |
| | ISAR_VCC_ACTVC | Query for active connections |
| | ISAR_VCC_INFO | Query connection configuration |
| | ISAR_VCC_STATS | Query connection statistics |
| | ISAR_VCC_ENABLE | Wake up a connection |
| | ISAR_VCC_DISABLE | Suspend a connection |
| | ISAR_VCC_SEND | Transmit a frame |
| | ISAR_VCC_ROUTE_ADD | Add an IP route to a connection |
| | ISAR_VCC_ROUTE_DEL | Remove an IP route |
| | ISAR_VCC_ROUTE_INFO | Query a connections route table |
| **OAM** | isar_oam_t | |
| | ISAR_OAM_OPEN | Open OAM on a connection |
| | ISAR_OAM_CLOSE | Close OAM on a connection |
| | ISAR_OAM_ENABLE | Wake up OAM, chip or connection |
| | ISAR_OAM_DISABLE | Suspend OAM, chip or connection |
| | ISAR_OAM_SEND | Transmit an OAM cell |
| **ALARM** | isar_alarm_t | |
| | ISAR_ALARM_OPEN | Open, configure alarm interface |
| | ISAR_ALARM_CLOSE | Close alarm interface |
| | ISAR_ALARM_INFO | Query alarm configuration |
| | ISAR_ALARM_TUNE | Modify configured parameters |
| | ISAR_ALARM_ENABLE | Wake up the alarm interface |
| | ISAR_ALARM_DISABLE | Suspend the alarm interface |
| | ISAR_ALARM_CLEAR | Clear (ack) an alarm event |
| | ISAR_ALARM_LOG | Configure system log events |

| | | |
|---|---|---|
| **PHY** | isar_phy_t | |
| | ISAR_PHY_INFO | Query phy count and revision information |
| | ISAR_PHY_SELECT | Select working phy |
| | ISAR_PHY_SWITCH | Switch working phy |
| | ISAR_PHY_STATUS | Query CX28250 status |
| | ISAR_PHY_STATS | Query CX28250 statistics |
| | ISAR_PHY_GEN | Query CX28250 gen-use registers |
| | ISAR_PHY_CELL | Query CX28250 cell registers |
| | ISAR_PHY_SONET | Query CX28250 SONET registers |
| | ISAR_PHY_MASKS | Query CX28250 intr masks |
| | ISAR_PHY_INTRS | Query CX28250 intr indications |

## OUTPUTS

| | |
|---|---|
| Return STATUS | Message transport result. |
| struct->hdr.status | API function status code. |

## RETURN VALUES

If the handshake completed successfully, `0` is returned and the OUTPUT completion status in the control structure indicates the result of the control request. Otherwise, `-1` is returned, the state of the OUTPUT status field in the control structure is unreliable.

## SEE ALSO

*API Guide* on page 67

## FILES

`isar_types.h, isar_api.h, isar_alarms.h, isar_phy.h, isar_codes.h`

## isar_rcv_ind

### NAME

`isar_rcv_ind` – User supplied receive indication callback

### SYNOPSIS

#include <`isar_api.h`>

#include <isar_xxxx.h> /* xxxx = os type: linux, vxworks or solaris */

```
void isar_rcv_ind(void *isar_rcv_ind_s);
```

### DESCRIPTION

This user function will be called for all received packets in the open, enabled connection, as identified by the VCID.

This callback is supplied in the `isar_vc_open()` function, along with the optional argument.

The driver will perform any packet reassembly and/or decapsulation required by the packet AAL type (such as AAL5 trailers).

It is the responsibility of the user function to free the data buffer back to the system using the supplied memory function macro MEM_FREE in the isar_ostype.h header file.

### INPUTS

The input data structure `isar_`rcv_ind_s' specifies all the operating parameters for this receive:

```
typedef struct isar_rcv_ind_s {
        ISAR_VCID   vcid;       /* isar VCC handle              */
        VOIDP       arg;        /* indication argument          */
        U32         length;     /* buffer length in bytes       */
        VOIDP       datap;      /* data buffer pointer          */
        VOIDP       lbolt;      /* tick count from system boot time, Solaris only */
        U32         seq_no;     /* driver rx frame sequence number, Solaris only */
        U32         pdu_uu_cpi; /* the uu and cpi cs5 bytes and the pdu checks field
} iasr_rcv_ind_t;
```

`pdu_uu_cpi` field bit format:

| 31-19 | 18 | 17 | 16 | 15-08 | 07-00 |
|---|---|---|---|---|---|
| Reserved | CLP | CI_LAST | CI | UU | CPI |

`CLP` = Value of the CLP header bit ORed across the CPCS-PDU
`CI_LAST` = Value of the PTI[1] header bit in the last cell of the CPCS-PDU
`CI` = Value of the PTI[1] header bit ORed across the CPCS-PDU
`UU` = AAL5 CPCS UU field
`CPI` = AAL5 CPCS CPI field

## RETURN VALUES

None.

## SEE ALSO

*isar_msg* on page 59

## FILES

`isar_api.h` – Contains all related data structures and definitions.

## isar_oam_ind

### NAME

`isar_oam_ind` – User-supplied OAM indication callback

### SYNOPSIS

#include <`isar.h`>

```
void isar_oam_ind(void *isar_rcv_ind_s);
```

### DESCRIPTION

This user function will be called for all received OAM cells, as identified by the `vcid`.

This callback is supplied in the `isar_oam_open()` function, along with the optional argument.

All cells are transparent to the driver, and the entire 48-byte cell is put into the buffer. The driver does no processing of these cells. The specific OAM type must be parsed from the cell data

It is the responsibility of the user function to free the data buffer back to the system.

### INPUTS

The input data structure ‘isar_`rcv_ind_s`' specifies all the operating parameters for this receive, and includes the following:

typedef struct isar_rcv_ind_s {

```
ISAR_VCID   vcid;       /* isar VCC handle          */
VOIDP       arg;        /* indication argument      */
U32         length;     /* buffer length in bytes   */
VOIDP       datap;      /* data buffer pointer      */
VOIDP       lbolt;      /* tick count from system boot time, Solaris only */
U32         seq_no;     /* driver rx frame sequence number, Solaris only */
U32         pdu_uu_cpi;/* the uu and cpi cs5 bytes and the pdu checks field
```

} iasr_rcv_ind_t;

`pdu_uu_cpi` field bit format:

| 31-19 | 18 | 17 | 16 | 15-08 | 07-00 |
|---|---|---|---|---|---|
| Reserved | CLP | CI_LAST | CI | UU | CPI |

`CLP` = Value of the CLP header bit ORed across the CPCS-PDU
`CI_LAST` = Value of the PTI[1] header bit in the last cell of the CPCS-PDU

`CI` = Value of the PTI[1] header bit ORed across the CPCS-PDU
`UU` = AAL5 CPCS UU field, not applicable to OAM frames
`CPI` = AAL5 CPCS CPI field, not applicable to OAM frames

**RETURN VALUES**

None

**SEE ALSO**

*isar_msg* on page 59

**FILES**

`isar.h` – Contains all related data structures and definitions.

## isar_alarm_ind

### NAME

`isar_alarm_ind` – User supplied alarm indication callback function

### SYNOPSIS

#include <`isar_api.h`>

#include `<isar_alarms.h>`

```
void isar_alarm_ind(void *alarm_ind_s);
```

### DESCRIPTION

This function is called whenever any enabled alarm condition has occurred.

This callback is supplied in the `isar_alarms()` function, along with the enabling of any of the supported conditions.

Alarm conditions are “edge triggered”, meaning that their corresponding bit is set at the origin of the alarm. Once read, the bit is cleared although the condition may still exist.

Some conditions, such as vc open or close, will require application debounce. See *Chapter 10 API Guide on page 67* for further explanation.

If the auto clear function was not enabled in the **`isar_alarm`** open function, the application must do an **`ISAR_ALARM_CLEAR`** on the appropriate state bits before another of those event types can be reported again.

### INPUTS

See *Chapter 10 API Guide on page 67* for details on the supported alarm conditions and bit assignments.

### RETURN VALUES

None.

### SEE ALSO

*isar_msg* on page 59

### FILES

`isar_api.h, isar_alarms.h` – Contains all related data structures and definitions.

# Operations and Programming Overview

Figure 10-1 is a overview of API functions.

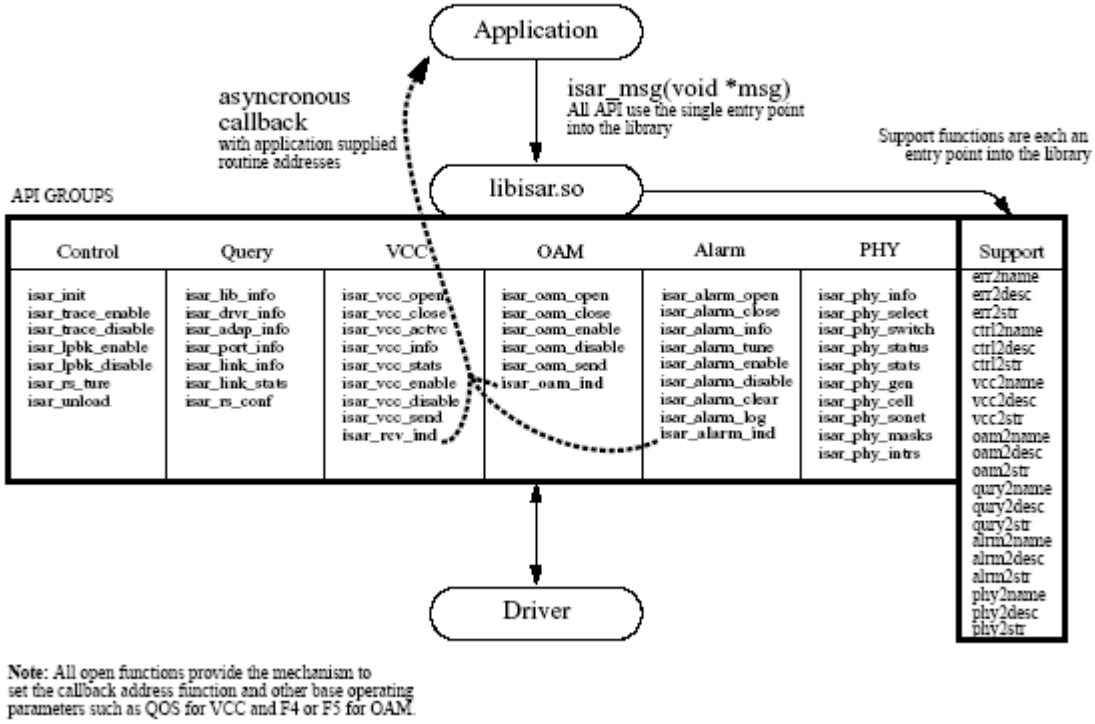



**Figure 10-1. API Functions Overview**

## Basic Sequence of Events

1. Query (probe) the driver for configuration status, driver, adapter, link (mtu), and connection information.
2. If the driver has not already been configured: parse the configuration file.
3. Initialize the interface (Solaris and Vxworks, set IPOA if desired) and set the interrupt tuning parameters.
4. Enable OAM chip level operations, if desired.
5. Open connections with desired traffic shaping parameters.
6. Open OAM on connections, if desired.

7. Set up alarms (if desired).
8. Open and close or enable and disable AAL or OAM connections.
9. Transmit and receive data.
10. Query statistics.
11. Close connections.
12. Close the application.
13. Exit

# API Architecture

The API architecture is comprised of seven logical groups; alarm, control, query, phy, oam, vcc (connection), and support. All but the support group share a common header type followed by a union with elements for each sub-command type and use the common entry point `isar_msg`( void *msg). Where message is a pointer to the appropriate structure type for the particular API group the message is for.

# Common Structures

## Header

```
typedef struct isar_hdr_s {
       U32    status;       /* completion status                */
       U32    type;         /* API type code                    */
       U32    link;         /* link number, where applicable    */
       U32    bytesOut;     /* api->drv buffer length           */
       U32    bytesIn;      /* drv->api buffer length           */
       ISAR_USR_TAGtag;     /* opaque user tag/id               */
       U32    api;          /* reserved, filled in by the library*/
} isar_hdr_t;
```

### status

This field indicates completion status of the message. It is recommended that before issuing a message, the field be filled in with a defined pattern to insure that the returned value has actually been filled in (see *Support* on page 102 for more details) since a successful status indication is 0.

### type

This field identifies the message.

### link

This field identifies the specific link (adapter) the message is intended for.

### bytesOut

This field specifies the size of the message, in bytes, that the user is sending. The driver uses this count to copy the message from user to kernel space.

### bytesIn

This field specifies the size of the completed message, in bytes, that the user is expecting to be returned from the driver. The driver uses this count to copy the message from kernel to user space.

### tag

This field is not used by the API or the driver. It is intended to be used as an identifier by the application.

### api

This field is reserved, anything placed in this field will be overwritten by the library.

## Tune

Another common type shared by functions is `tune_t` used to control tunable parameters. It has two components; one for control, one for the actual value. The control element has four action modes; ignore, off, set, and factory.

```
typedef struct tune_s {
       U32    action;       /* control function   */
       U32    value;        /* parameter value    */
} tune_t;
```

### action

Some API types contain multiple fields for tuning functions. This field is used to control the actions performed on individual fields.
`ISAR_TUNE_IGN` – Ignore, do not alter
`ISAR_TUNE_SET` – Set parameter to value
`ISAR_TUNE_OFF` – Turn parameter off, disable
`ISAR_TUNE_FAC` – Set parameter to factory default, typically this is off

### value

The actual value to use if action is `ISAR_TUNE_SET`.

# API Groups

## Alarm

This API group is used to configure, control, and query the alarm interface. There are three types of alarms available; SAR, PHY, and VC. The SAR alarms deal with packet error thresholds, the PHY alarms deal with line state changes, and the VC alarms indicate when connections are opened or closed. The same structure type, `isar_alarm_t`, is used in both directions; API to driver, driver to API.

### Structure

```
typedef struct isar_alarm_s {
       isar_hdr_t  hdr;    /* api common header           */
/*
* info parameters
* api->drv: ISAR_ALARM_CLEAR(state)
* drv->api: ISAR_ALARM_INFO
*/
       U32    state;       /* current link state          */
       U32    rxcrcs;      /* link rsm crc count          */
       U32    rxlens;      /* link rsm len count          */
       U32    rxtmos;      /* link rsm tmo count          */
/*
* tune parameters
* api->drv: ISAR_ALARM_OPEN, ISAR_ALARM_TUNE
* drv->api: ISAR_ALARM_INFO
*/
       tune_t mask;        /* alarm masks                 */
       tune_t rxcrc;       /* rsm crc threshold           */
       tune_t rxlen;       /* rsm len threshold           */
       tune_t rxtmo        ;/* rsm tmo threshold          */
       tune_t sfdt;        /* Sig Fail/Degrade            */
       tune_t k1k2;        /* K1K2 bytes                  */
/*
* open parameters
* api->drv: ISAR_ALARM_OPEN
* drv->api: ISAR_ALARM_INFO
*/
       U32    oflag;       /* open flags                  */
       ISAR_IND_ARG        arg;   /*alarm indication arg   */
       ISAR_USR_TAG        tag;   /* opaque user tag       */
       ISAR_ALARM_IND      *ind;  /* alarm indication func */
/*
* additional info
* drv->api: ISAR_ALARM_INFO
*/
       U32    phystate;    /* PHYs state words            */
       U32    rxk1k2;      /* Rx K1/K2 bytes              */
       U32    vpivci;      /* channel's VPI/VCI           */
       ISAR_VCID    vcid;  /* VCC handle                  */
/*
* system logging mask
* api->drv: ISAR_ALARM_LOG
* drv->api: ISAR_ALARM_INFO
```

```
*/
       U32    log;   /* which events to send to system log file*/
} isar_alarm_t;
```

## Prototypes

```
U32 isar_msg (void *alarm_s);

typedef void (ISAR_ALARM_IND)(void *); /* callback */
```

## API Sub Commands

- ISAR_ALARM_OPEN – Activate alarms
- ISAR_ALARM_CLOSE – Terminate alarms
- ISAR_ALARM_INFO – Retrieve current alarm configuration information
- ISAR_ALARM_TUNE – Modify existing thresholds
- ISAR_ALARM_ENABLE – Wake up alarms
- ISAR_ALARM_DISABLE – Suspend alarms
- ISAR_ALARM_CLEAR – Clear an alarm condition
- ISAR_ALARM_LOG – Set system logging mask

## Alarm Open

When opening the alarm interface there are three major fields to control the how and which events can occur; oflag, mask, and ind.

### oflag

The oflag, open flag, currently has three bit level functions available:

ISAR_ALARM_GLOBAL BIT(0)

ISAR_ALARM_EN_NOW BIT(1)

ISAR_ALARM_IND_CLR BIT(2)

The global bit registers alarms for all adapter to the application. The enable now bit indicates that alarms are to begin operations at the completion of the open. The indication clear bit instructs the driver to clear the event bits when the alarm is posted, otherwise the application must clear them with the ISAR_ALARM_CLEAR API before any new alarms for those bits can be posted.

### mask

The mask field determines which event bits to enable for event posting.

### ind

This is the function for the API library to call for alarm processing.

### Alarm Close

The alarm close API terminates all event processing. If logging functions have been enabled they will continue to function.

### Alarm Info

This API allows the user to retrieve the current alarm structure in the driver. This function not only returns current configuration parameters but also supplies real-time phy state information, useful in debouncing events.

### Alarm Tune

This API allows the user to modify current configuration and threshold parameters for general alarms, pdu errors, and APS parameters. These entities are identified in the alarm structure by the parameter type of `tune_t`.

### Alarm Enable

This API allows the user to wake up the alarm event process. The event process could have been suspended from the open function or a previous alarm disable API.

### Alarm Disable

This API allows the user to suspend current alarm events. To allow alarm event processing to continue the alarm enable API must be issued.

### Alarm Clear

This API is used to acknowledge an alarm event to the driver. The driver will not report another specific event until the current one has been cleared. This is only necessary if the user did not specify auto clear in the oflag field for the alarm open operation.

### Alarm Log

This API is used to identify which phy events should be sent to the system log file. The available bits are referenced below in the Alarm Event description for event bits, bits 31-17. To disable the logging function issue this API with the log field set to 0.

### Alarm Events

When an alarm event is received, the handling function should walk all bits in the structure element `state` to determine which events are being reported. The alarm structure is real-time logical or'd in nature.

If the auto clear bit was not set for the `ISAR_ALARM_OPEN` function, the user application servicing the indiction must do an `ISAR_ALARM_CLEAR` on the specific state bits before any of those events can be reported again.

### Event Bits (mask and state)

| APS | | | | PHY/Framer | | | | | | | | | | | | SAR/PDU | | | | | | | | PCI | | | | Control | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| k1k2 | degr | fail | psbf | rdip | aisp | rdil | aisl | lop | lof | oof | lol | los | locd | sigd | phy | rtmo | rlen | rcrc | runf | rovf | sunf | qvld | – | vdel | vadd | – | pci | – | – | en | conf |

#### SAR Events

These events are packet-related such as under- and over-flow, CRC, and length and time-outs. A connection ID may be supplied with some SAR events identifying the `vpi`, `vci`, and the `vcid` structure element. The tunable parameters set thresholds for the event reporting (i.e. if the user is only interested in CRC errors after '`n`' have occurred).

#### PHY Events

These events are based on the physical and SONET layer states. The physical layer events are generally associated with loss of signal, i.e. cable pull or degradation of line quality. The SONET layer events are intended for APS device level support such as the reporting of a K1K2 byte change in the received SONET frames. When servicing a physical layer state change, it may be necessary to debounce the events, since PHY events are real time and some states tend to change back and forth during a cable pull. The debounce can be accomplished by either buffering the alarm events with a timer or by polling the phy state using the `ISAR_ALARM_INFO` API.

#### VC Events

These events indicate connections in the driver have been opened or closed. It is the responsibility of the application to determine which connections are involved. This is accomplished in two steps; debounce and query. The debounce function queries the driver using **`ISAR_LINK_INFO`** to wait for the active vc count field to settle out. Then query the driver with **`ISAR_ACTV_VC`** for the settled-out vc count to determine which connections are now in the driver. This new list can now be compared with the current list in the application. In this way an application handling OAM functions can maintain a list of connections to monitor.

## Control

This API is a generic control call. All control API structures are comprised of a common header with a union holding each of the individual API sub command structure types. There is also a support structure for the initialize API to pass in VPI and VCI counts to be configured if the user has chosen list mode connection management in the flags field.

## Structure

```
typedef struct vcisvpi_s {
      U16 vpi;                          /* VPI value             */
      U16 vci;                          /* number of VCI's modulo 64*/
} vcisvpi_t;

typedef struct isar_ctrl_s {
      isar_hdr_t   hdr;                                          /*
request header*/
      union {
      struct ctrl_init_s {
            U32    mtu;              /* in - MTU                    */
            U32    seg_vccs;         /* in - seg VCC table entries */
            U32    seg_qsize;        /* in - seg queue(s) size     */
            U32    rsm_qsize;        /* in - rsm queue(s) size     */
            U32    sch_pcr;          /* in - line cell rate        */
            U32    sch_mcr;          /* in - qos granularity       */
            U32    flags;            /* in - sonet-sdh, clp, txclk */
            U8     addr[ISAR_ADDR_LEN]; /* in - default SPA/Mask      */
            U32    snoop;            /* in - VxWorks snoop enable  */
            void   *bsp;             /* in - VxWorks BSP           */
            U32    entries;          /* in - VCI/VPI entries       */
            vcisvpi_tentry[1];       /* in - entry 0 - n           */
      } init;

      struct ctrl_tune_s {
            tune_t intr_tmr;     /* timer-based interrupt holdoff   */
            tune_t intr_cnt;     /* status queue interrupt holdoff  */
            tune_t tx_fifo;      /* tx fifo depth (1-9)             */
      }
      } uc;
      struct ctrl_lbpk_s {
            U32    phy;          /* which phy to place in/out of loopback */
            U32    flag;         /* loopback mode                   */
} isar_ctrl_t;
```

## Prototype

```
u32 isar_msg(void *ctrl_s);
```

## API Sub Commands

- `ISAR_INIT` – Set operating parameters for the driver
- `ISAR_TRACE_ENABLE` – Enable driver and API tracing
- `ISAR_TRACE_DISABLE` – Disable driver and API tracing
- `ISAR_LPBK_ENABLE` – Enable on-board loopback, currently not implemented
- `ISAR_LPBK_DISABLE` – Disable on-board loopback, currently not implemented
- `ISAR_RS_TUNE` – Configure the number of status queue entries per interrupt
- `ISAR_UNLOAD` – Unload the current configuration and resources for the adapter.
- `ISAR_CLOSE` – Close out the receive interface thread, for this application, pid.

## Initialize

The application should query the link with `ISAR_LINK_INFO` to determine if the controller has been initialized. If the card is not initialized, the error code `IA_ERR_STATE` will be returned and the adapter can be initialized with any valid MTU size, otherwise use the returned MTU size for the current application.

### MTU

The MTU field sets the limits for packet sizes on receive and transmit operations. The minimum value is 48 bytes (hardware cell pdu size) and the maximum is 65536 bytes (including any overhead). The value used for the hardware will be rounded up to the next 1K boundary. Thus for user mtu of 48 bytes, hardware allocates a 1k buffer and a user mtu of 65520 would use system buffer of 65536 (64k). If the MTU size is max (64K) the user must take into account that the transmit function needs to include the CS5 Trailer in the last cell and that the receiver deals in multiples of 48 bytes and a 16-bit frame buffer size. Thus, the max TX PDU is 65512 which is 1365 cells for an RX hardware PDU of 65520 (including the CS5 trailer) and user PDU of 65512.

### flags

The **flags** entry allows the user to set a number of operation parameters.

| **31-16** | **15-09** | **08** | **07** | **06** | **05** | **04** | **03** | **02** | **01** | **00** |
|---|---|---|---|---|---|---|---|---|---|---|
| rxbt | rsvd | scrm | conn | mem | vpi | sque | clip | tclk | clp | frmr |

`frmr` – This bit controls the framer mode of operation, STM or SDH. The bit is cleared for STM and set for SDH.

`clp` – This bit controls the cell loss priority bit in the atm header for idle cells. If set, all idle cells have the CLP bit in the `atm` header set.

`tclk` – This bit determines the source of clock for the transmit interface on the framer. If this bit is cleared, the clock source is the on-board oscillator. If the bit is set, the transmit clock source is the recovered receive clock from the wire.

`clip` – This bit tells the driver to set up for classical IP operations over ATM. This bit is not used for the Linux version of the driver.

`sque` – This bit determines the location of the status queues for transmit and receive operations. If cleared, the queues reside in host memory, if set, the queues reside in adapter memory. This bit is mainly useful for systems that have cache coherency issues for shared data structures with hardware. Normal operation is to have the queues in host memory for performance reasons.

`vpi` – This bit controls the range of VPI values. If the bit is cleared, UNI operations are selected with 8 bits of VPI. If the bit is set, NNI operations are selected with 12 bits of VPI.

`mem` – This bit determines if the driver is to perform a memory test on the adapter prior to initialization. Keep in mind that if this bit is set it will extend the initialize operation completion time.

conn – This bit determines the connection mode management. If this bit is cleared then connection management is controlled by the entries and vcisvpi_s fields of the init structure, list mode. If this bit is set, connection management is open to any range valid value for VPI and any VCI 1-32767, dynamic mode. In dynamic mode the full 32 K connection range may not be attained if all connections are disparate. This is due to the fact that connection blocks for the chip set are done in modulo 64. So, a single connection of VPI/VCI 0/32 would consume 64 connection entries, 0-63 for that VPI. If VPI/VCI 0/33, the driver would simply enable it inside of the existing block. However if VPI/VCI 1/33 were opened, then another block of 64 entries would be consumed.

scrm – This bit controls cell scrambling mode. If cleared, normal cell scrambling is enabled. If this bit is set, cell scrambling is disabled.

rsvd – These bits are currently reserved and should be cleared.

rxbt –

- Solaris/Linux

  These bits define the receive backlog threshold. This value, multiplied by 4096, determines the number of bytes the driver is allowed to backlog in the user receive list. The backlog byte count increments when a packet arrives from the chip and is decremented when a packet is retrieved by the library, user application. When a packet is received, if packet size plus current backlog size is greater than the threshold, the packet will be dropped by the driver. This parameter is necessary to insure that system memory is not completely consumed by the driver in situations where the user level application's bandwidth is not sufficient to keep up with incoming traffic.
- VxWorks

  These bits define the number of cluster blocks, cBlk, to pre-allocate for CLIPOA connections if CLIPOA is enabled. The driver will then allocate two memory control blocks, mBlk, for each cBlk of MTU size.

Once the driver has been initialized, the parameters cannot be changed. If the parameters need to be modified then the user must issue the **ISAR_UNLOAD** API. This will cause the driver to clear out all internal structures, return all system resources and reset the hardware. Now the user can initialize the adapter with the new parameters.

### addr

This 8-byte field is used to set the default SPA (Source Protocol Address) and mask for IPOA connections. During the **ISAR_VCC_OPEN** API, if the SPA field in the open structure is empty the driver will fill it in with this value.

### seg_vccs

This field is available to limit the number of transmit connections possible at one time and is only valid if the conn bit in the flags field is cleared. The entry determines the size of the segmentation table for the chip set, typically it is set to the value of the sum of the entries. Valid range is 64 to 32768 (or 65536 if using the 8-MB memory configured interface).

#### seg_qsize - rsm_qsize

These entries determine the size of the status and command queue for the chip set and accordingly, how many buffers of size MTU to allocate for the hardware from system DMA memory. Valid entries are 64, 256, 1024, or 4096.

#### sch_pcr - sch_mcr

These entries determine the line cell rate for the card and the shaping and size of the QoS scheduling table for the hardware. The values supplied in the default `isarconf.0` file are set up for a line cell rate of 354838, even though the value is 354000, and a granularity of 50 cells per second. This means that rates used to open connections with must be a multiple of `sch_mcr` in order to not incur cell delay variation slower than the requested cell rate.

***NOTE***

**Contact Interphase support if this configuration does not fit your requirements or run the supplied application `isarxbr` to determine the best values to use.**

#### entries - vcisvpi_s

The entries and entry array, `vcisvpi_s`, inform the driver as to which VPIs and associated VCIs are to be configured in the hardware for possible operation, if the `conn` bit in the flags field is cleared. The entry's element states how deep the entry array is. The entry array describes the VPI and how many VCIs to enable for that VCI. The VCI value must be modulo 64.

### Trace enable

This API will enable tracing in the library and the driver. For the library this amounts to `printf's` to `stdout`. For the driver, the messages go the console or system log.

### Trace disable

This API terminates tracing operations.

### Loopback enable

This API sets one of four loopback modes for the adapter.

#### phy

This field determines which phy to set this loopback mode for.

### flag

| 31-04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|
| rsvd | pci2sar | pci2phy | line2line | line2phy |

`line2phy` – This mode of loopback takes data from the cable, line, and routes the data through the framer and back out the line.

`line2line` – This mode of loopback takes data from the cable, line, and routes the data directly back out the line.

`pci2phy` – This mode of loopback is the more traditional mode where data from the host is routed through the SAR, then to the framer, back to the SAR, and then back to the host.

`pci2sar` – This mode of loopback is similar to `pci2phy` except that data does not leave the SAR and does not involve the framer.

## Loopback disable

This API will terminate any previous mode of loopback and place the adapter in normal operation.

## RS tune

The default configuration of the card is to enable the hardware such that it interrupts the host for each status queue entry. This would normally equate to one interrupt per frame received. However, the driver disables interrupts on the adapter while servicing an existing interrupt.

This is an acceptable configuration for most applications. However, data intensive applications may need to use the **`ISAR_RS_TUNE`** API to modify the interrupting characteristics of the hardware in order to optimize system performance. This API sets the number of status queue entries per interrupt for the hardware and a parallel timer. Once a status queue entry is written by the hardware, the timer starts, so either the status queue reaches its interrupt count or the timer expires to deliver an interrupt to the host.

`intr_tmr` – This entry is in microseconds, thus a 1 ms timer would be a value of 1000.

`intr_cnt` – This entry should not exceed the queue size set in the initialization structure.

`tx_fifo` – This entry sets the depth of the transmit fifo. The size of the TX FIFO is a parameter used in QOS VBR parameter calculations. Valid range for this value is 1 to 9. It is recommended that this value is left at the default (9).

## Unload

This API is used to put the driver back to an un-initialized state. All memory resources are returned to the host, all timers and semaphores are released and the adapter is reset.

### Close

This API is used to indicate to the library and driver that the application, pid (process ID) is closing. The driver will insure that all connections are closed, the receive list is drained, and the read semaphore released. The library will close out the background block on read thread. For Linux this API must be issued if a connection was ever opened by the application. For Solaris or VxWorks it is simply a way of doing clean-up for any receives that may be left in the driver backlog queue.

## Query

This API is a generic query call. All query API structures are comprised of a common header with a union holding each of the individual API sub-command structure types.

### Structures

typedef struct isar_query_s {

```
isar_hdr_thdr;/* request header                                */
union {

struct query_lib_s {
       char   vendor[ISAR_ILEN];  /* out - vendor              */
       char   desc[ISAR_ILEN];    /* out - description         */
       char   environ[ISAR_ILEN]; /* out - environ OS-PROC_BIT */
       char   version[ISAR_ILEN]; /* out - release/build number */
}lib;

struct query_drvr_s {
       char   vendor[ISAR_ILEN];  /* out - vendor              */
       char   desc[ISAR_ILEN];    /* out - description         */
       char   environ[ISAR_ILEN]; /* out - environ OS-PROC_BIT */
       char   version[ISAR_ILEN]; /* out - release/build number */
       U32    adapters;           /* out - adapter count       */
       U32    uptime;             /* out - adapter uptime      */
}driver;

struct query_adap_s {
       U32    adapter;            /* in  - adapter index       */
       char   vendor[ISAR_ILEN];  /* out - manufactor          */
       char   model[ISAR_ILEN];   /* out - model string        */
       char   bus[ISAR_ILEN];     /* out - bustype string      */
       char   name[ISAR_ILEN];    /* out - name string         */
       U32    devid;              /* out - device id           */
       U32    subid;              /* out - subsystem id        */
       U32    clsrev;             /* out - class/rev           */
       U32    hwrev;              /* out - class/rev           */
       U32    serial;             /* out - serial number       */
       U32    ports;              /* out - on board memory size */
       U32    sram;               /* out - on board memory size */
       U32    sar;                /* out - SAR type/rev        */
       U32    phy;                /* out - SAR type/rev        */
       U8     umac[ISAR_MAC_LEN];/* out - user MAC address*/
       U8     fmac[ISAR_MAC_LEN];/* out - factory MAC address*/
} adapter;
```

```
        struct query_port_s {
               U32   adapter;             /* in  - adapter index        */
               U32   port;                /* in  - port index           */
               U32   type;                /* out - phy type/rev         */
               U32   speed;               /* out - rate (Mbits/sec)     */
               U32   connector;           /* out - rate (Mbits/sec)     */
               U32   state;               /* out - phy state word       */
        } port;

        struct query_link_s {
               U32   adapter;             /* in - adapter number        */
               U32   port;                /* in - port number           */
               U32   state;               /* out - link up (true,false) */
               U32   mtu;                 /* out - max packet size      */
               U32   schPcr;              /* out - line cell rate       */
               U32   schMcr;              /* out - minimum cell rate    */
               U32   schAllot;            /* out - bandwidth allocated  */
               U32   segVccs;             /* out - maximum total VCs    */
               U32   rsmVccs;             /* out - maximum total VCs    */
               U32   activeVcs;           /* out - active VCs           */
               U32   segQsize;            /* out - maximum total VCs    */
               U32   rsmQsize;            /* out - maximum total VCs    */
               U32    uptime;             /* out - uptime in seconds    */
               U8     addr[ISAR_ADDR_LEN];/* out - SPA and netmask*/
        } link;

        struct query_lstat_s {
               U32    aal0In;             /* out - counter              */
               U32    aal0Out;            /* out - counter              */
               U32    pcktsIn;            /* out - counter              */
               U32    pcktsOut;           /* out - counter              */
               U32    octetsIn;           /* out - counter              */
               U32    octetsOut;          /* out - counter              */
               U32    oamIn;              /* out - counter              */
               U32    oamOut;             /* out - counter              */
               U32    discards;           /* out - counter              */
               U32    losCount;           /* out - counter              */
               U32    uptime;             /* out - uptime in seconds    */
        } lstats;

        struct rs_conf_s {
               U32    sysclk;             /* out - clock in megahertz   */
               U32    slotper;            /* out - clocks per slot      */
               U32    tblsize;            /* out - scheduling table size*/
               U32    txfifo;             /* out - tx fifo size         */
               U32    intr_tmr;           /* out - timer intr holdoff   */
               U32    intr_cnt;           /* out - statQ intr holdoff   */
        }rsconf;
} uq;
} isar_query_t;
```

## Prototype

```
u32 isar_msg (void *query_s)
```

## API Sub Commands

- **ISAR_LIB_INFO** – Get the library revision information
- **ISAR_DRVR_INFO** – Get driver information (revision , release date, etc.)
- **ISAR_ADAP_INFO** – Get adapter information (PCI info, number of ports, size of memory, etc.)
- **ISAR_PORT_INFO** – Get port information (type, speed, etc.)
- **ISAR_LINK_INFO** – Get link information (queue sizes, number of available connections, etc.)
- **ISAR_LINK_STATS** – Get link statistics (packets in/out, bytes in/out, etc.)
- **ISAR_RS_CONF** – Get hardware information used by "`isarxbr`" to calculate VBR connection parameters

## Query Functions

The Library and Driver revision fields are built at compile time, thus they can be used to verify that the two entities are in sync. The following descriptions for the sub-commands simply show a typical print output for the return data described in the above structure unions.

### Library information

```
vendor : Interphase
desc   : x576 iSAR API driver
env    : SunOS 5.8 sparc 32-bit
version: 3.0 build sun.4
```

### Adapter information

```
bd#    : 0
vendor : Interphase
model  : 4576
bus    : PMC
name   : SX00653-A01       (Interphase part number for the serial eeprom)
devid  : 0x823414f1        (PCI vendor-device ID for the Conexant chipset)
subid  : 0x10107e          (PCI subsystem and subvendor ID, Interphase)
class  : 0x2030005
hwrev  : 0x10
serial : 0x0
ports  : 0x1
sram   : 0x401800          (memory space occupied by card)
sar    : 0x5               (SAR revision from the chip)
phy    : 0x77              (PHY revision from the chip)
umac   : 000077123456      (User MAC)
fmac   : 000077123456      (Factory MAC)
```

### Driver information

```
vendor :   Interphase
desc   :   x576 iSAR API driver
env    :   SunOS 5.8 sparc 32-bit
version:   3.0 build sun.4
bd cnt :   1
```
uptime : 172964 seconds

### Port information

bd# : 0
port : 0
type : 119 (phy revision)
speed : 155 Mbits/sec (line rate supported)
conn : SC-MM (SC connector multi-mode)
state : 0x3 (enabled, active...reference isar_phy.h)

### Link information

bd# : 0
port : 0
state : 0x1fff (driver state bits)
mtu : 4096
SchPcr : 351063 (configured line rate, QoS parameter)
SchMcr: 166 ( QoS cell rate granularity)
SchBand: 0 (QoS bandwidth allocated)
SegVcc : 4096
RsmVcc: 1024
ActVcc : 2
SegQsz : 64
RsmQsz: 64
uptime : 321184 seconds

### Link Statistics

aal0In : 0x0
aal0Out: 0x0
pcktsIn : 0x0
pcktsOut: 0x15
octetsIn: 0x0
octetsOut: 0x0
oamIn : 0x0
oamOut: 0x0
discards: 0x0
losCount: 0x0
uptime : 321249 seconds

### Hardware information (RS_CONF)

Sysclk : 33000000 mhz
Clk/Slt : 94
TblSize : 2114
Txfifo : 8
IntQtmr: 0 (user configured interrupt timer, tuning parameter)
IntQcnt : 0 (user configured interrupt status queue count, tuning parameter)

# PHY

This API group allows the user to query phy, Conexant CX28250, registers, and control phy selection. For specific register bit information you must obtain a CX28250 data sheet, available from MindSpeed Technologies™.

## Structures

```
typedef struct isar_phy_s {
       isar_hdr_t   hdr;
       union  {
       struct phy_info_s {
              U32    count;              /* number of phys             */
              U32    active;             /* active phy number          */
              U32    version;            /* phy's versions 16 bits each*/
              U32    state;              /* phy's state 16 bits each   */
       } info;

        /*
          * The following structures return their respective register
          * contents. Please refer to the Conexant CX28250 data
          * sheet for register definitions & bit assignments.
          */

       struct phy_status_s {
              U32   rxaps;                /*   8 - RXAPS           */
              U32   rxcell;               /*   8 - RXCELL          */
              U32   rxline;               /*   8 - RXLIN           */
              U32   rxpath;               /*   8 - RXPT            */
              U32   rxsect;               /*   8 - RXSEC           */
              U32   txcell;               /*   8 - TXCELL          */
       } status;

       struct phy_stats_s {            /* bits - register*/
              U32   locd;                 /*   8 - LOCDCNT         */
              U32   heccor;               /*   8 - CORRCNT         */
              U32   hecunc;               /*   8 - UNCCNT          */
              U32   oof;                  /*   8 - OOFCNT          */
              U32   secbip;               /* 16 - B1CNTx           */
              U32   linbip;               /* 18 - B2CNTx           */
              U32   pthbip;               /* 16 - B3CNTx           */
              U32   linrei;               /* 18 - LFCNTx           */
              U32   pthrei;               /* 16 - PFCNTx           */
              U32   nomtch;               /* 16 - NONCNTx          */
              U32   txcells;              /* 19 - TXCNTx           */
              U32   rxcells;              /* 19 - RXCNTx           */
        } stats;

        struct phy_gen_s {              /* bits - register       */
            U32   gen;                  /*   8 - GEN             */
            U32   clkrec;               /*   8 - CLKREC          */
            U32   outstat;              /*   8 - OUTSTAT         */
            U32   version;              /*   8 - VERSION         */
            U32   utop1;                /*   8 - UTOP1           */
            U32   utop2;                /*   8 - UTOP2           */
            U32   udf2;                 /*   8 - UDF2            */
        } gen;
```

```
        struct phy_cell_s {             /* bits - register         */
            U32   cgen;                 /*   8 - CGEN              */
            U32   cval;                 /*   8 - CVAL              */
            U32   idlpay;               /*   8 - IDLPAY            */
            U32   idlmsk;               /* 32 - IDLMSKx            */
            U32   rxhdr;                /* 32 - RXHDRx             */
            U32   rxidl;                /* 32 - RXIDLx             */
            U32   rxmsk;                /* 32 - RXMSKx             */
            U32   txhdr;                /* 32 - TXHDRx             */
            U32   txidl;                /* 32 - TXIDLx             */
        } cell;


struct phy_sonet_s {            /* bits - register                 */
            U32   txsec;                /*   8 - TXSEC             */
            U32   txlin;                /*   8 - TXLIN             */
            U32   txpth;                /*   8 - TXPTH             */
            U32   txk1;                 /*   8 - TXK1              */
            U32   txk2;                 /*   8 - TXK2              */
            U32   txs1;                 /*   8 - TXS1              */
            U32   txc2;                 /*   8 - TXC2              */
            U32   txz01;                /*   8 - TXZ01             */
            U32   txz02;                /*   8 - TXZ02             */
            U32   apsthresh;            /*   8 - APSTHRESH         */
            U32   rxk1;                 /*   8 - RXK1              */
            U32   rxk2;                 /*   8 - RXK2              */
            U32   rxs1;                 /*   8 - RXS1              */
            U32   rxc2;                 /*   8 - RXC2              */
            U32   rxg1;                 /*   8 - RXG1              */
            U32   rxz01;                /*   8 - RXZ01             */
            U32   rxz02;                /*   8 - RXZ02             */
        } sonet;

        struct phy_masks_s {            /* bits - register         */
            U32   ensumint;             /*   8 - ENSUMIT           */
            U32   ensec;                /*   8 - ENSEC             */
            U32   enlin;                /*   8 - ENLIN             */
            U32   enpth;                /*   8 - ENPTH             */
            U32   encellt;              /*   8 - ENCELLT           */
            U32   encellr;              /*   8 - ENCELLR           */
            U32   enaps;                /*   8 - ENAPS             */
        } masks;

        /*
         * IMPORTANT - these clear when read & are needed by the driver!!
         */
        struct phy_intrs_s {            /* bits - register         */
            U32   sumint;               /*   8 - SUMINT            */
            U32   secint;               /*   8 - SECINT            */
            U32   linint;               /*   8 - LININT            */
            U32   pthint;               /*   8 - PTHINT            */
            U32   txcellint;            /*   8 - TXCELLINT         */
            U32   rxcellint;            /*   8 - RXCELLINT         */
            U32   apsint;               /*   8 - APSINT            */
        } intrs;
} up;
```

```
} isar_phy_t;
```

## Prototype

```
U32 isar_msg (void *phy_s);
```

## API Sub Commands

- **ISAR_PHY_INFO** – Get CX28250 revision information
- **ISAR_PHY_SELECT** – Select the working phy
- **ISAR_PHY_SWITCH** – Toggle working to the other phy
- **ISAR_PHY_STATUS** – Get CX28250 status
- **ISAR_PHY_STATS** – Get CX28250 statistics registers
- **ISAR_PHY_STATS** – Get CX28250 statistics
- **ISAR_PHY_GEN** –Get CX28250 general/user registers
- **ISAR_PHY_CELL** – Get CX28250 cell registers
- **ISAR_PHY_SONET** – Get CX28250 SONET layer registers
- **ISAR_PHY_MASKS** – Get CX28250 interrupt mask registers
- **ISAR_PHY_INTR** – Get CX28250 interrupt registers

## PHY Functions

The phy functions that are simply register reads:

### phy information

| | | |
|---|---|---|
| count: | 1 | |
| active: | 0 | |
| version: | 0x0000 0x0077 | (phy version from the CX28250 register) |
| state(1-0): | 0x0000 0x0003 | (reference `isar_phy.h`) |

### phy select

This API is used to directly enable a specific phy for primary operation. The phy not selected is put into standby, tx only, mode.

### phy toggle

This API is used to toggle the phy port operation modes. If phy 0 was in primary and phy 1 in standby, then after this API, they would be phy 0 standby and phy 1 primary.

### phy status, stats, gen, cell, sonet, masks

These APIs return the current contents of a specific group of phy registers.

### phy intr

This API also returns current register contents. However, since it is the interrupt register of the phy, using this API may cause the driver to lose alarm event status or logging information as a read from this register clears the interrupt indication.

## VCC

The connection-oriented VCC API group is used to open, close, query statistics, send, and receive data on specific VPI-VCI circuits. The type **ISAR_VCID**, which is a return value for the open function, is a handle to be used for identifying the connection of interest for any data or connection operations.

### NOTE

**For Linux, any application that opens a connection must either issue the ISAR_CLOSE API when exiting the application, or issue the API when entering the application as a normal clean-up function for any previous application. This is necessary to terminate the library's background thread block-on-read function.**

### Structures

typedef struct isar_vcc_s {

```
isar_hdr_t hdr;          /* request header              */
  ISAR_VCID vcid;        /* VCC handle                  */

  union {
      struct vcc_open_s {
          U32vci;        /* virtual curcuit identifier  */
          U32vpi;        /* virtual path identifier     */
          U32type;       /* AAL0,AAL5                   */
          U32mode;       /* UBR,VBR,ABR,CBR             */
          U32xbr1;       /* QoS PCR/I1L1 parameters     */
          U32xbr2;       /* QoS MCR/I2L2 parameters     */
          VOIDPrcv_ind;  /* receive indication callback */
          VOIDPrcv_arg;  /* receive argument            */
          VOIDPtag;      /* opaque user tag/id          */
          U32txq_depth;  /* max txs on TxQ at one time  */
          charaddr[VCC_ADR_LEN]; /* address             */
          U32enable;     /* enable vc/oam flags         */
      } open;

      struct vcc_info_s {
          U32actvc;      /* active vcc number           */
          U32slot;       /* VCC table slot number       */
          U32state;      /* vc state                    */
          U32vci;        /* table index's VCI           */
          U32vpi;        /* table index's VPI           */
          U32type;       /* AAL type (0 or 5) */
          U32mode;       /* mode (UBR, VBR, CBR) */
          U32tqdepth;    /* TxQ threshold               */
          U32tqinuse;    /* TxQ enqueued                */
          U32uptime;     /* uptime in secs              */
          VOIDPrcv_ind;  /* rx callback                 */
          VOIDPrcv_arg;  /* rx callback argument        */
          VOIDPoam_ind;  /* rx oam callback             */
          VOIDPoam_arg;  /* rx oam callback arg         */
```

```
        } info;

        struct vcc_stats_s {
            U32aal0In;     /* counter                      */
            U32aal0Out;    /* counter                      */
            U32pcktsIn;    /* counter                      */
            U32pcktsOut;   /* counter                      */
            U32octetsIn;   /* counter                      */
            U32octetsOut;  /* counter                      */
            U32oamIn;      /* counter                      */
            U32oamOut;     /* counter                      */
            U32discards;   /* counter                      */
            U32rxCrc;      /* counter                      */
            U32rxLen;      /* counter                      */
            U32rxTmo;      /* counter                      */
            U32uptime;     /* uptime                       */
        } stats;

        struct vcc_send_s {
            U32ctrl;       /* control parameters           */
            U32length;     /* buffer length in bytes       */
            VOIDPdatap;    /* data buffer pointer          */
            VOIDPtag;      /* opaque user tag/id           */
    } send;

struct vcc_route_s {
       char   addr[ISAR_IP_LEN]; /* NETWORK ORDER IP address */
       char   mask[ISAR_IP_LEN]; /* netmask for route */
       } route;

struct vcc_route_info_s {
       U32    count; /* in = max entries to return, out = number of entries
returned */
       isar_raddr_t entry[1]; /* netmask for route */
       } route;

    } uv;

} isar_vcc_t;

typedef struct isar_rcv_ind_s {
       ISAR_VCID    vcid;         /* isar VCC handle                */
       ISAR_IND_ARG arg;          /* indication argument            */
       U32          length;       /* buffer length in bytes         */
       VOIDP        datap;        /* data buffer pointer            */
       VOIDP        lbolt;        /* SOLARIS ONLY, system tick count */
       U32          seq_no;       /* SOLARIS ONLY, rx sequence number*/
       U32          pdu_uu_cpi;   /* CI, CLP bits, CS5 UU and CPI values */
} isar_rcv_ind_t;

typedef struct isar_raddr_s {
        ip_addr_t addr;
        ip_addr_t mask;
} isar_raddr_t;
```

### Prototypes

```
U32   isar_msg (void *vcc_s);

typedef void (ISAR_RCV_IND)(void *);
```

### API Sub Commands

- **ISAR_VCC_OPEN** – Open a VPI/VCI connection.
- **ISAR_VCC_CLOSE** – Close a VPI/VCI connection.
- **ISAR_VCC_ACTVC** – Query connection information based on an index value.
- **ISAR_VCC_INFO** – Query connection information based on an **ISAR_VCID**.
- **ISAR_VCC_STATS** – Query connection statistics.
- **ISAR_VCC_ENABLE** – Wake up a connection for receive and transmit functions.
- **ISAR_VCC_DISABLE** – Put a connection to sleep, disables receive and transmit.
- **ISAR_VCC_SEND** – Transmit a frame on a connection.
- **ISAR_VCC_ROUTE_ADD** – Add an IP route to an existing IP connection.
- **ISAR_VCC_ROUTE_DEL** – Delete an IP route from an existing IP connection.
- **ISAR_VCC_ROUTE_INFO** -– Query IP routes for an existing IP connection.

### VCC Functions

#### Open VCC Connection

When opening a connection the user defines the VPI, VCI, AAL type, QoS mode, and transmit queue depth.

***NOTE***

**For Linux, any application that opens a connection must either issue the ISAR_CLOSE API when exiting the application, or issue the API when entering the application as a normal clean-up function for any previous application. This is necessary to terminate the library's background thread block-on-read function.**

If the QoS is VBR the user must run the application "**isarxbr**" to calculate the parameters needed for the `xbr1` and `xbr2` values for the open structure. If the QoS is CBR the "**isarxbr**" application can be used to calculate the appropriate values to use for the adapter initialization parameters `sch_pcr` and `sch_mcr` and verify that the desired rates will fit in the adapters scheduling table. If the QoS is CBR or VBR it is recommended that the `txq_depth` value be set to some appropriate percentage of the `seg_qsize` parameter used when the adapter was configured. This is because slow rate connections have the tendency to consume the driver and adapter resources due to queuing and `txq_depth` allows the driver to error off a transmit request for that connection if the current queue count is still on the card being segmented.

The rcv_ind entry is filled in with the address of the desired receive handler for that connection. The ISAR API does not use native system interfaces to pass data to the user, such as sockets or streams, it is an asynchronous callback from the API to the supplied rcv_ind. The rcv_arg and tag fields are not used by the driver and can be used to save application pointers, they are returned to the user for each receive callback.

vcid – Upon successful completion of the open function, this field is filled in by the driver with a connection handle to be used in all subsequent operations to identify this connection.

vci – This field identifies the virtual circuit indicator value for the connection.

vpi – This field identifies the virtual path indicator value for the connection.

type – This field determines the AAL operation for the connection, AAL0 or AAL5 only.

mode – This field determines the QoS for the connection, UBR, VBR1 (single leaky bucket), VBR2 (dual leaky bucket), or CBR.

xbr1 – This field is a QoS parameter for VBRx and CBR connections. For CBR, this value represents cells/second. If the rate does not fit into the scheduling table evenly, an error warning of IA_ERR_CBR_WARN will be returned. For VBRx, this value is a pre-formatted exponent-mantissa value derived from the isarxbr application and represents the PCR portion of the rate control.

xbr2 – This field is a QoS parameter for VBRx connections. This value is a pre-formatted exponent-mantissa value derived from the isarxbr application and represents the SCR and burst portions of the rate control.

rcv_ind – This field identifies the address of the application receive indication callback routine.

rcv_arg –This field is returned in the isar_rcv_ind_t for all receive callback operations for this connection.

tag – This field is not used by the driver and is returned as is on the open.

txq_depth - This field sets the limit for the number of pending transmit requests for this connection. This field is especially useful for rate controlled connections in order to insure that they do not consume all of the driver transmit resources. If a transmit request is made for a connection and that connection has txq_depth number of packets currently in the hardware's transmit queue, an error of IA_ERR_TXQ will be returned.

addr – This field is only for IP connections and identifies the SPA (Source Protocol Address) and TPA (Target Protocol Address) for the connection. If the SPA is NULL the SPA will be assigned with the value that was supplied in the ISAR_INIT function. The first four bytes represent the SPA and the last four bytes represent the TPA.

enable – Field is bit oriented and has various meanings based on the type of connection.

| 31-28 | 27-06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|
| ccnt | rsvd | term | rawc | arpd | ipoa | oaml | enow |

enow – Setting this bit will enable the connection for transmit and receive operations upon completion of the open. If this bit is cleared, the API **ISAR_VCC_ENABLE** must be issued before transmit and receive operations are allowed.

oaml – Setting this bit will lock out all OAM operations for this connection. If an application attempts to do an **ISAR_OAM_OPEN** API call, it will return an error.

ipoa – Setting this bit identifies the connection for IPOA operations (Solaris and VxWorks).

arpd – Setting this bit disables the IPOA ARP functions. It is recommended that this bit is set.

rawc –This bit applies to AAL0 connections. If this bit is set, the user will receive all 52 bytes of the AAL0 cell (header and payload), otherwise, if cleared, then only the 48-byte payload is delivered to the user application.

term – This bit applies to AAL0 connections and determines receive termination mode. If this bit is set, reception is terminated based on the cell count field in bits 31-28. If this bit is cleared, reception is terminated based on the PTI bit in the cell header being set.

rsvd – These bits are reserved and should be cleared to 0.

ccnt – These bits apply to AAL0 connections. If bit 05 (term) is set, these bits define the number of cells to receive before reception is terminated and the frame sent to the user application.

---

retmtu – This field returns the current MTU the driver has been initialized with for transmit and receive operations.

### Close VCC Connection

To close a connection only requires the vcid field. Note that the function will return immediately. However, the connection itself will remain open at the hardware level until all queued frames have been transmitted. Any subsequent open requests for this connection will fail with ISAR_ERR_CLOSE if the connection is still in the process of closing due to frames still in the queue. This condition is especially applicable to slow-rate CBR or VBR connections.

vcid – Connection handle identifying which circuit to close.

### Active VCC information

This API allows the user level to query connection information based on an index rather than a connection handle. It is useful for probing the driver to find out what kind of connections are opened at any time. This API in combination with the API **ISAR_LINK_INFO** can be used to probe the driver to find out how many and what kinds of connections are currently open. **ISAR_LINK_INFO** returns the number of active connections, then **ISAR_VCC_ACTVC** can be issued for each of the connections.

vcid – Connection handle returned for the requested index.

`actvc` – This field identifies the internal index of the connection to return information for.

`slot` – This field identifies the internal slot for this connection.

`state` – This bit-oriented field indicates the current state of the connection.

| 31-12 | 11 | 10 | 09 | 08 | 07-05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|
| rsvd1 | arpd | iperr | ipoa | ipen | rsvd | cls | oaml | oam | en | bsy |

`bsy` – This state bit is always set for any open connection.

`en` – If this state bit is set, the connection is enabled for transmit and receive.

`oam` – If this state bit is set, the connection is enabled for OAM traffic.

`oaml` – If this state bit is set, all OAM operations have been locked out.

`cls` – If this state bit is set, this connection is still in the process of closing from a previous **`ISAR_VCC_CLOSE`** API, which means that there are still transmit frames in the hardware queue.

`rsvd` – These state bits are currently reserved.

`ipen` -– This state bit is set for IP connections.

`ipoa` – This state bit indicates that IP operations are enabled. This bit will always be set if the connection was opened with ARP disabled.

`iperr` – This state bit indicates an error has occurred and typically is due to an ARP failure.

`arpd` – This state bit indicates that the ARP function is disabled for this connection.

`rsvd1` – These state bits are reserved.

---

`vci` – Virtual circuit indicator for this connection.

`vpi` – Virtual path indicator for this connection.

`type` – AAL type for this connection.

`mode` – QoS mode for this connection.

`tqdepth` – Transmit queue threshold for this connection.

`tqinuse` – Number of transmit packets currently in the hardware queue.

`uptime` – Number of seconds this connection has been opened.

`rcv_ind` – Address of the application receive callback function.

`rcv_arg` – Application receive argument for callback functions.

`oam_ind` – Address of the application OAM receive callback function.

`oam_arg` – Application OAM receive argument for callback functions

`addr` – For IPOA connections, this field contains the SPA and TPA, in that order, for this connection.

`lasterr` – For IPOA connections only, this field shows the last error.

### VCC information

This API allows the user to query connection information based on a connection handle and returns the same data as the **`ISAR_VCC_ACTVC`** API.

`vcid` – Connection handle to return information for.

### VCC Statistics

This API allows the user to query connection statistics based on a connection handle.

`vcid` – Connection handle to return statistics for.

`aal0In` – Number of received AAL0 frames, currently not used.

`aal0Out` – Number of transmitted AAL0 frames, currently not used.

`pcktsIn` – Number of received frames for this connection type.

`pcktsOut` – Number of transmitted frames for this connection type.

`octetsIn` – Number of bytes received, currently not used.

`octetsOut` – Number of bytes transmitted, currently not used

`oamIn` – Number of OAM frames received.

`oamOut` – Number of OAM frames transmitted.

`discards` – Number of dropped frames, currently not used.

`rxCrc` – Number of receive frame CRC errors.

`rxLen` – Number of receive frame length errors.

`rxTmo` – Number of receive frame time outs.

`uptime` – Number of seconds this connection has been opened.

### VCC enable

This API will wake up (enable) this connection for transmit and receive operations.

### VCC disable

This API will put to sleep (disable) this connection for transmit and receive operations.

### Transmit (Send) VCC Data

When transmitting data, the user supplies the `vcid` to identify the circuit to send on along with a buffer pointer (`datap`), `length`, control field (`ctrl`), and transparent `tag`.

The ctrl field allows the user to make the request a high priority packet which will cause it to go to the head of the current queue list. Normal transmits are always put at the tail of the current queue list. This field also allows the user to manipulate the ATM header field for this packet for the CLP (Cell Loss Priority) bit, CI (Congestion Indication), and UU (User-to-User) fields.

There are two main errors returned for transmit operations; ISAR_ERR_LIST, and ISAR_ERR_TXQ. The first indicates that all transmit queues are full on the hardware, the second indicates that the transmit queue for this connection is full, meaning that the number of transmits queued to the hardware for this connection has reached the txq_depth value configured during the open connection process. Both of these errors indicate that the application should go to a back-off algorithm or simply retry the operation. This is generally only an issue when using slow- rate CBR or VBR connections.

vcid – Connection handle for this transmit operation.

ctrl – This bit-oriented field allows the user to control various parameters for this transmit.

| 31-12 | 11 | 10 | 09 | 08 | 07-06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| uud | rsvd2 | uu | ci | clp | rsvd1 | f5e | f5s/raw | f4e | f4s | rsvd | pri |

pri – If this bit is set, the driver will indicate to the hardware that this is a high priority transmit and place it at the top of the transmit queue. Otherwise, it is placed on the end of the queue in normal first in first out fashion.

rsvd – This bit is reserved and should be cleared to 0.

f4s – If this bit is set, the frame will be sent as an F4 segment-to-segment OAM frame, if this connection has been opened and enabled for OAM operations.

f4e – If this bit is set, the frame will be sent as an F4 end-to-end OAM frame, if this connection has been opened and enabled for OAM operations.

f5s/raw – If this connection is AAL5, setting this bit will cause the frame to be sent as an F5 segment-to-segment OAM frame, if this connection has been opened and enabled for OAM operations. If the connection is AAL0, setting this bit indicates that the user data will include the ATM header and payload (52 bytes), otherwise, with the bit cleared, the SAR appends the ATM header to the frame (48 bytes).

f5e – If this bit is set, the frame will be sent as an F5 end-to-end OAM frame, if this connection has been opened and enabled for OAM operations.

rsvd1 – These bits are reserved and should be cleared to 0.

clp – Setting this bit will cause all cells for this frame to have the CLP (Cell Loss Priority) bit set in the ATM header. If the connection mode is VBR2, this bit enables VBR CLP Dual Leaky Bucket mode.

ci – Setting this bit will cause all cells for this frame to have the CI (Congestion Indication) bit in the ATM header, PTI[1].

uu – Setting this bit will indicates that the user is supplying the data byte for the AAL5 CS5 trailer UU field.

rsvd2 – These bits are reserved and should be cleared to 0.

uud – If the UU bit is set, this value will be placed in the AAL5 CS5 Trailer UU field of this frame.

---

length – Number of bytes to transmit.

datap – Application data buffer pointer for this transmit frame.

tag – Opaque user tag, the driver does not use this field. It is returned unchanged.

### Receive AAL Data

The applications supplied receive handler will be called by the API when incoming packets are received. The structure supplied to the handler contains an ISAR_VCID identifying the connection the data came in on, the transparent arg supplied in the open process, the length of the frame, and a pointer to the data buffer. It is the responsibility of the application to free the data buffer using the macro in the isar_ostype.h file, where ostype is solaris, linux or vxworks, MEM_FREE.

vcid – Connection handle identifying the source of this receive frame.

arg – This is the user supplied rcv_arg from the **ISAR_VCC_OPEN** function.

length – Number of bytes for this frame.

datap – Pointer to the frame buffer. This buffer must be freed by the application using the MEM_FREE macro in the isar_ostype.h header file (where ostype is solaris, linux or vxworks).

lbolt – Solaris only, this is the system tick counter at the time the driver received the frame.

seq_no – Solaris only, this is the sequence number of this frame relative to the driver.

pdu_uu_cpi – This field supplies CI, CLP, UU, and CPI information for this frame.

| 31-19 | 18 | 17 | 16 | 15-08 | 07-00 |
|---|---|---|---|---|---|
| rsvd | clp | ci_last | ci | uu | cpi |

cpi – This is the value from the CS5 trailer CPI (Common Part Indicator) field.

uu – This is the value from the CS5 trailer UU (User to User) field.

ci – This bit represents the OR'd value of all cell's PTI[1] bits, congestion indiction.

ci_last – This bit is the PTI[1] bit, congestion bit, from the last cell of this frame.

`clp` – This bit represents the OR'd value of all cell's ATM header CLP bits, cell loss priority.

`rsvd` – These bits are reserved.

### VCC route add

This API is for IPOA connections and adds an IP route to an existing connection, essentially adding a hash entry for the IP to VPI/VCI lookup for transmits. When an IP frame is transmitted the driver does an IP to VPI/VCI lookup based on the mask field. Thus, if addr 1.1.1.1 and mask of 255.255.0.0 was added as a route, any 1.1.x.x packet would go out over this connection.

`vcid` – Connection handle to add route to.

`addr` – Network order IP address to add to this connection's lookup table.

`mask` – Netmask to use for IP to VPI/VCI hash table lookup.

### VCC route delete

This API will remove an entry from a connections lookup table, usage is the same as the add function.

### VCC route information

This API will return '`n`' entries from a connections lookup table.

**vcid** – Connection handle to query for IP routes.

**count** – When issuing the API, this value indicates the maximum number of entries to return. On completion of the API, this value represents the actual number of entries returned.

**entry[n]** – This field is of type `isar_raddr_t` that contains and IP address and mask.

# OAM

This API group allows the user to open, close, enable, disable, send, and receive for OAM operations. The user is responsible for formatting the cell data, the driver is simply the transport function and does not contain any OAM logic.

The transmit and receive functions use the VCC API structures.

## Structures

```
typedef struct isar_oam_s {

       isar_hdr_t    hdr;          /* request header                    */
       ISAR_VCID     vcid;         /* VCC handle                        */

       union {
       struct oam_open_s {
              VOIDP         rcv_ind;      /* receive indication callback*/
              VOIDP         rcv_arg;      /* receive argument           */
              VOIDP         tag;          /* opaque user tag/id         */
              U32           enable;       /* enable after open          */
       } open;

       struct oam_close_s {
              U32           enable;       /* lock vc to block oam open's*/
       } close;

       struct oam_enable_s {
              U32           flag;         /* lock vc to block oam open's*/
       } enable;
       } uo;
} isar_oam_t;
```

## Prototypes

```
U32 isar_msg (void *oam_s);

void isar_oam_ind (void *rcv_ind);
```

## API Sub Commands

- **ISAR_OAM_OPEN**
- **ISAR_OAM_CLOSE**
- **ISAR_OAM_ENABLE**
- **ISAR_OAM_DISABLE**
- **ISAR_OAM_SEND**

## OAM Functions

### OAM open

Before OAM can be opened for a connection, the connection itself must have already been opened at the VCC level without the `ISAR_OAM_LOCK` option. In order to transmit or receive OAM data, the OAM interface must be enabled at the hardware level. The user provides a `vcid` to identify the connection, the address of the receive handler for the callback function, `rcv_ind`, along with a transparent argument, `rcv_arg`, that are returned for the callback.

### OAM close

The only parameter needed to close out OAM for a connection is the associated `vcid`.

### OAM enable

The enable API has two functions, hardware (chip) level and connection level, that are identified in the `flag` field. For the hardware level function, the RS8234 is programmed to receive OAM cells on all open connections. Note that this has an effect on VCs 3 and 4 for all VPIs, as the chip recognizes traffic on these channels as OAM F4. The connection-oriented enable function provides a filter mechanism in the driver.

### OAM disable

The disable API has two functions, hardware (chip) level and connection level, that are identified in the **flag** field. For the hardware level function, the RS8234 is programmed to ignore (discard) all OAM cells. The connection-oriented disable function provides a filter mechanism in the driver.

### OAM (send) transmit

The transmit function uses the same structure type as the AAL transmit function, `vcc_send_s`. The `ctrl` field is used to indicate the OAM type.

- `ISAR_VCS_F4SEG` – OAM F4 segment
- `ISAR_VCS_F4E2E` – OAM F4 end to end
- `ISAR_VCS_F5SEG` – OAM F5 segment
- `ISAR_VCS_F5E2E` – OAM F5 end to end

***NOTE***

**If either of the F4 bits are used, the hardware will transmit them on either VC 3 or 4 of the given VPI, no matter what `vcid` was indicated. It is the user's responsibility to format the cell data, the driver is only the transport function.**

### OAM receive callback

The user supplied callback function will be supplied with an `isar_rcv_ind_t` structure identifying the `vcid`, data pointer (`datap`) and `length` of the frame (cell) along with the `rcv_arg` and `tag` supplied in the open function. The user must parse the data buffer to determine the OAM type and function, the driver is only the transport function. The user is responsible for freeing the data buffer.

## Support

This set of APIs support analyzing and printing command and status codes. The support header file delivered with the driver is `isar_codes.h`. The groups of API, errors, control, query, alarms, and phy, each have a function to extract the name, description, or identifier string.

### Support API

```
char *err2name(int code);
char *err2desc(int code);
char *err2str (int code);

char *ctrl2name(int code);
char *ctrl2desc(int code);
char *ctrl2str (int code);

char *vcc2name(int code);
char *vcc2desc(int code);
char *vcc2str (int code);

char *oam2name(int code);
char *oam2desc(int code);
char *oam2str (int code);

char *qury2name(int code);
char *qury2desc(int code);
char *qury2str (int code);

char *alrm2name(int code);
char *alrm2desc(int code);
char *alrm2str (int code);

char *phy2name(int code);
char *phy2desc(int code);
char *phy2str (int code);
```

### Support Functions

An example of printing a control API and an error code is:

```
sprintf(stdout, "Function %s,%s failed with %d:%s\n", ctrl2name(ictrl.hdr.type,
ctrl2desc(ictrl.hdr.type), ictrl.hdr.status, err2str(ictrl.hdr.status);
```

## ERROR Codes

Error codes returned come in two groups: ISAR and IA. The ISAR error codes are typically high-level user interface type failures. The IA error codes are typically driver function type failures. Successful completions return 0 in the status field.

```
ISAR_ERR_NONE = 0, IA_ERR_NONE = 0
```

## API Error Codes

**Table 10-1. API Error Codes**

| Numeric Value | Display | Description |
|---|---|---|
| 1024 | ISAR_NOT_SUPP | API type is not supported |
| 1025 | ISAR_ERR_PTR | Error accessing a supplied pointer |
| 1026 | ISAR_ERR_TYPE | Invalid API type |
| 1027 | ISAR_ERR_GRP | Invalid API group |
| 1028 | ISAR_ERR_SUB | Invalid API sub group |
| 1029 | ISAR_ERR_LEN | Length is 0 or too small |
| 1030 | ISAR_ERR_BUFF | Null buffer pointer |
| 1031 | ISAR_ERR_LINK | Link number out of range |
| 1032 | ISAR_ERR_VCI | VCI out of range |
| 1033 | ISAR_ERR_VPI | VPI out of range |
| 1034 | ISAR_ERR_VCBLK | VC block modulo error when initializing in LIST mode |
| 1035 | ISAR_ERR_VCCS | Max VCCs exceeded when initializing in LIST mode |
| 1036 | ISAR_ERR_VCID | Invalid vcid, connection handle |
| 1037 | ISAR_ERR_AAL | Invalid AAL type |
| 1038 | ISAR_ERR_MODE | Invalid QoS mode |
| 1039 | ISAR_ERR_PCR | Invalid PCR value for initialization |
| 1040 | ISAR_ERR_IND | OAM open has null callback |
| 1041 | ISAR_ERR_INUSE | Connection is already open |
| 1042 | ISAR_ERR_CLOSE | Connection is still in the process of closing from a previous open-close. |
| 1043 | ISAR_ERR_IDX | Illegal index value |
| 1044 | ISAR_ERR_CNT | Illegal count value |
| 1045 | ISAR_ERR_MAGIC | Bad magic cookie |
| 1046 | ISAR_ERR_VDIS | Connection is disabled |
| 1047 | ISAR_ERR_MEMA | Memory allocation failure |
| 1048 | ISAR_ERR_INITED | Link already initialized |
| 1049 | ISAR_ERR_NOBUF | No user buffer available, not used |
| 1050 | ISAR_ERR_PTI | Invalid F5 PTI value, not used |
| 1051 | ISAR_ERR_ADAP | Invalid adapter number |
| 1052 | ISAR_ERR_PORT | Invalid port number |
| 1053 | ISAR_ERR_PROM | Eeprom data not available |
| 1054 | ISAR_ERR_SYNTAX | Parsing format syntax error |

## Driver Error Codes

**Table 10-2. Driver Error Codes**

| Numeric Value | Display | Description |
|---|---|---|
| 2048 | IA_ERR_IOC | Unknown **ioctl** command |
| 2049 | IA_ERR_SOFTC | Null or invalid softc, adapter control structure |
| 1050 | IA_ERR_SUBSYS | Invalid PCI subsystem ID |
| 2051 | IA_ERR_REQTBL | Request table type mismatch |
| 2052 | IA_ERR_STATE | Invalid state for this request |
| 2053 | IA_ERR_DRVRNOT | Driver not initialized |
| 2054 | IA_ERR_LINKNOT | Link not initialized |
| 2055 | IA_ERR_INITED | Link already initialized |
| 2056 | IA_ERR_MTU | MTU out of range |
| 2057 | IA_ERR_SEG_VCCS | SegVccs out of range for initializing in LIST mode |
| 2058 | IA_ERR_SEG_QSIZE | SegQSize out of range |
| 2059 | IA_ERR_SEG_MCR | Not used |
| 2060 | IA_ERR_RSM_VCCS | Not used |
| 2061 | IA_ERR_RSM_QSIZE | RsmQSize out of range |
| 2062 | IA_ERR_SCH_PCR | SchPCR out of range |
| 2063 | IA_ERR_SCH_MCR | SchMCR out of range |
| 2064 | IA_ERR_SCH_XBR1 | QoS parameter xbr1 out of range |
| 2065 | IA_ERR_SCH_XBR2 | QoS parameter xbr2 out of range |
| 2066 | IA_ERR_SCH_BCKT | All scheduler buckets in use |
| 2067 | IA_ERR_VCISVPI | Init value for vcisvpi_s out of range |
| 2068 | IA_ERR_SRAM | SAR RAM allocation failure |
| 2069 | IA_ERR_DMAA | DMA memory allocation failure |
| 2070 | IA_ERR_MEMA | Memory allocation failure |
| 2071 | IA_ERR_KMEM | Kernel memory allocation failure |
| 2072 | IA_ERR_BSML | Buffer too small for data |
| 2073 | IA_ERR_LIST | Null or empty list |
| 2074 | IA_ERR_COPY | Copy in or copy out failure |
| 2075 | IA_ERR_SYNC | DMA sync failed |
| 2076 | IA_ERR_MAGIC | Bad magic cookie |
| 2077 | IA_ERR_BUSY | Resource busy |
| 2078 | IA_ERR_NOACK | EEPROM failure |
| 2079 | IA_ERR_EMPTY | Tx resource queue is empty |

**Table 10-2. Driver Error Codes (cont)**

| Numeric Value | Display | Description |
|---|---|---|
| 2080 | IA_ERR_TXQ | `txq` threshold exceeded |
| 2081 | IA_ERR_BANDW | Bandwidth allocation failure |
| 2082 | IA_ERR_CBR_IDX | Invalid VCC index for scheduler |
| 2083 | IA_ERR_CBR_TBL | Rate will not fit in schedule table |
| 2084 | IA_ERR_TYPE | Bad internal receive buffer descriptor type |
| 2085 | IA_ERR_LOCK | OAM is locked for this connection |
| 2086 | IA_ERR_TUNE_ACT | Invalid tuning action |
| 2087 | IA_ERR_TUNE_VAL | Tuning value out of range |
| 2088 | IA_ERR_IEXP_MIN | Not used |
| 2089 | IA_ERR_IEXP_MAX | Not used |
| 2090 | IA_ERR_LEXP | Not used |
| 2091 | IA_ERR_MAN_MAX | Not used |
| 2092 | IA_ERR_TX_FIFO | Tx FIFO `len` out of range |
| 2093 | IA_ERR_ALM_NOT | Alarms not open on link |
| 2094 | IA_ERR_ALM_SOFTC | Invalid `softc` for this alarm structure |
| 2095 | IA_ERR_ALM_MAGIC | Bad magic cookie |
| 2096 | IA_ERR_ALM_RDY | Link not ready for alarms |
| 2097 | IA_ERR_ALM_BUSY | Alarms already in process for this link |
| 2098 | IA_ERR_ALM_USR | Alarm process ID mismatch |
| 2099 | IA_ERR_ALM_TBL | Alarm table code mismatch |
| 2100 | IA_ERR_IOC_CMD | Unknown **`ioctl`** command |
| 2101 | IA_ERR_IOC_TBL | Not used |
| 2102 | IA_ERR_IOC_FUNC | Linux: Unable to allocate semaphore func |
| 2103 | IA_ERR_IOC_LEN | Bad **`ioctl`** length |
| 2104 | IA_ERR_IOC_CI | Failed to copy in user data to kernel |
| 2105 | IA_ERR_IOC_CO | Failed to copy out user data from kernel |
| 2106 | IA_ERR_RCV_REG | Not used |
| 2107 | IA_ERR_ALM_CRC | Invalid CRC threshold value |
| 2108 | IA_ERR_ALM_LEN | Invalid `len` error threshold value |
| 2109 | IA_ERR_ALM_TMO | Invalid `rx tmo` threshold value |
| 2110 | IA_ERR_ALM_SFDT | Invalid `sfdt` threshold value |
| 2111 | IA_ERR_ALM_K1K2 | Invalid `k1k2` threshold value |

**Table 10-2. Driver Error Codes (cont)**

| Numeric Value | Display | Description |
|---|---|---|
| 2112 | IA_ERR_CBR_WARN | Rate did not fit in table evenly, transmit operation will produce CDV at table wrap |
| 2113 | IA_ERR_NULL_PTR | IPOA null ARP pointer |
| 2114 | IA_ERR_HASH_NULL | Link has no hash table |
| 2115 | IA_ERR_CLIP_NOT | Connection not enabled for IPOA |
| 2116 | IA_ERR_CLIP_ACT | IPOA state not active |
| 2117 | IA_ERR_CLIP_ERR | IPOA error detected |
| 2118 | IA_ERR_VCID_HASH | Unable to find hash entry |
| 2119 | IA_ERR_VCID_SPA | Null SPA, source IP address |
| 2120 | IA_ERR_VCID_TPA | Null TPA, target IP address |
| 2121 | IA_ERR_LLC_HDR | Invalid LLC SNAP header |
| 2122 | IA_ERR_LLC_TYPE | Invalid LLC SNAP type |
| 2123 | IA_ERR_ARP_OPER | Invalid ARP code |
| 2124 | IA_ERR_ARP_SHTL | Invalid ARP source address |
| 2125 | IA_ERR_ARP_SSTL | Invalid ARP source sub address |
| 2126 | IA_ERR_ARP_SPLN | Invalid ARP source protocol address |
| 2127 | IA_ERR_ARP_THTL | Invalid ARP target address |
| 2128 | IA_ERR_ARP_TSTL | Invalid ARP target sub address |
| 2129 | IA_ERR_ARP_TPLN | Invalid ARP target protocol address |
| 2130 | IA_ERR_ARP_SPA | Null ARP source address |
| 2131 | IA_ERR_ARP_TPA | Null ARP target address |
| 2132 | IA_ERR_ARP_AGE | ARP age time-out |
| 2133 | IA_ERR_ARP_COMP | Failed to compress ARP address |
| 2134 | IA_ERR_NET_DOWN | Solaris: IP stream interface is gone |
| 2135 | IA_ERR_HASH_DUP | Hash entry already exists |

# Troubleshooting 11

## Overview

This chapter provides possible solutions for common problems you might encounter while installing and operating the ATM card. Before proceeding with the troubleshooting, make sure you have carefully followed the steps for installing the hardware and the software, and have rebooted the system. Also, check the systems log for your operating system to see what kind of errors, if any, are being recorded. This may provide some insight into the problem.

If the information in this chapter does not resolve the problem, contact Interphase Customer Support at one of the locations listed in the front of this users guide.

## Interpreting LEDs

The card has two LEDs:

- Green Link LED
- Yellow/Green Status LED

Table 11-1 provides information about LED states.

**Table 11-1. LED States**

| LED | State | Definition |
|---|---|---|
| **Link** | Off | The card is not receiving a signal from a switch or remote system. |
| **Link** | Solid Green | The card is receiving a signal from a switch or remote system. |
| **Status** | Off | The card does not have power. |
| **Status** | Solid Yellow | The card has power. |
| **Status** | Flashing Yellow/Green | Flashing 1 sec. on yellow, then 1 sec. on green indicates the driver and board are ready.<br>Flashing ½ sec. on yellow, then ½ sec. on green indicates the driver is loaded and the interface has been configured. |

The green Link LED indicates the condition of the physical link between the card and the ATM switch. When the Link LED is lit, light or electrical current is flowing between the two sets of hardware; however, data may or may not be flowing through the cable.

The yellow Status LED serves two functions. When the machine is turned on, the LED glows a solid yellow to indicate that the card has power. Then it begins flashing to indicate that the software driver is loaded and an card interface has been configured.

# Problems and Possible Solutions

This section presents possible solutions for the following types of card problems:

- Link and status problems (*Link and Status Problems* on page 110)
- Boot problems (*Boot Problems* on page 112)
- Network problems (*Network Problems* on page 114)

## Link and Status Problems

Problems indicated by the Link and Status LEDs might be related to hardware failure, inadequate power access, or improper driver installation are shown in Table 11-2.

**Table 11-2. link and Status Problems**

| Problem | Possible Solution |
|---|---|
| **The Link LED is not lit.** | A failure in the hardware is preventing the card from completing a physical link with the switch.<br>**1.** Check the switch. It must be up and running for the card to make a link.<br>**2.** Unplug and reinstall the connectors.<br>**3.** Try a known good cable.<br>**4.** Check the transmit and receive wiring to see if they are reversed. See *Connecting to the Network* on page 11.<br>**5.** If a known good cable is installed, try another card in the PMC expansion slot.<br>If the green LED does not illuminate, the problem is probably at the switch. Confirm this by configuring the cable to another port on the switch. |
| **The Status LED is not lit.** | Power is not getting to the card.<br>**1.** Do a power reset of the computer by turning the power off and then on again.<br>**2.** Make sure the card is seated correctly in the PMC expansion slot. Check for a bent pin or crack in the connector.<br>**3.** Try another PMC expansion slot.<br>**4.** Try another card known to operate correctly. |

**Table 11-2. link and Status Problems (cont)**

| Problem | Possible Solution |
|---|---|
| **The Status LED is lit, but is not flashing.** | The driver is not installed correctly, or the network interface is not configured.<br>**1.** If the driver is not installed, install it.<br>**2.** If the driver is installed, check the system's error message log to see if any errors pertaining to the ATM driver were recorded during the boot process. This might provide some insight into the problem.<br>**3.** Reinstall the driver if it is not installed correctly.<br>**4.** If the driver is installed correctly, you might need to configure the network protocols, as described in the driver installation instructions for your system. |

## Boot Problems

Boot problems are usually hardware related. For example, the card is located in the wrong slot, or the bus address is incorrect or in conflict with another device. If the machine worked properly before the card was installed, then the problem is probably with the card or with hardware that was removed and replaced during card installation (Table 11-3).

**Table 11-3. Boot Problems**

| Problem | Possible Solution |
|---|---|
| **The computer does not boot up.** | **1.** Check to see if the card is properly seated in the expansion slot. Reseat the card and restart the machine.<br>**2.** Try a different expansion slot.<br>**3.** Remove the card and reboot the system.<br>**4.** If the system boots up and returns to a normal state, the card is probably defective. Confirm this by installing an card known to work correctly.<br>**5.** If a defective card is not the problem, try reinstalling the configuration utility (if applicable).<br>**6.** Try changing the interrupt setting or removing other hardware on the bus until the conflict is located.<br>Change one setting at a time and record the settings to keep track of your changes. |
| **The host card is not found.** | The system cannot find or does not recognize the card. The PCI system in your computer is supposed to automatically configure the bus address locations; therefore, an address conflict is probably not the problem. If the driver is installed correctly, a driver message should appear in the message log when the computer boots up.<br>**1.** If this is your first boot process since installing the card, the card hardware interrupt or configuration registers might not match what the system expects. Instead of rebooting the system, turn the power off and then on again to reset all components to an initial state.<br>**2.** The card might be installed in the wrong slot. Make sure the card is in the slot for which the driver is configured. See your system manual(s) for the location of the proper expansion slots.<br>**3.** Make sure the card is seated correctly in the slot.<br>**4.** Try another expansion slot, if available.<br>**5.** Try another card known to operate correctly.<br>**6.** Check connection of the network cable to the card card. Verify the cable is properly connected at both ends. Ping the failed system from another host on the network.<br>**7.** Check the error message log on your system to see if any errors pertaining to the ATM card or driver were recorded while booting. This might provide some insight into the problem. |

## Network Problems

The procedures in this section assume the following have occurred:

- The host workstation is up and running.
- The driver is loaded. (The yellow Status LED is flashing.)
- The card has formed a physical link with the switch. (The green Link LED is lit.)

If this is not the case, see the procedures in *Link and Status Problems* on page 110 and *Boot Problems* on page 112.

# ATM isartest Guide

# A

## Introduction

### Purpose

The purpose of this appendix is to provide information on the usage and functions of the ATM API application `isartest`. This application allows the user to exercise the card interface and hardware. It is intended to be run with another station running `isartest` or with a loopback cable, but can be configured to receive traffic from a non-`isartest` sender.

This tool is considered an engineering-level diagnostic and is subject to change without notice.

## Menu Tree

The following is a representation of the `isartest` menu tree:

```
[a]larm - Alarm API Functions
     [C]lear  - clear an alarm state
     [c]lose  - close the alarm interface
     [d]isable- suspend the alarm interface
     [e]nable - wake up the alarm interface
     [i]nfo   - query the current alarm structure
     [o]pen   - open and configure the alarm interface
     [t]une   - modify the parameters for the alarm interface
     [x]it    - leave this menu section
[c]ontrol - Control API functions
     [i]init  - configure the adapter
          [d]efault - use isartest default parameters
          [e]tc/atm/isarconf.x - use the installed configuration file
          [f]ile - use a selected file
          Display parameters (y/n)?
          Change(y/n)?
              segm[c]r - minimum cell rate, cell rate resolution
              [e]ntries - vpi-vci list
                  [c]reate new list
                  [m]odify entry
              [f]lag - sonet/sdh, clp, txclk source
              [m]tu - max packet size
              sch[p]cr - peak cell rate, line cell rate
              rsm[q]size - receive queue size
              [s]egqsize - transmit queue size
              seg[v]ccs - number of transmit connections
              [x]it - leave this menu section
     [l]oopback - enable/disable loopback mode
     [r]stune - modify the interrupt interface parameters
     [t]race  - enable/disable api/driver tracing
     [u]nld   - unload the current configuration and resources
[d]isplay - parameters or rxbuf or txbuf
```

```
     [c]onn   - current open connections in driver
     [p]armeters - isartest variables
     [r]xbuf  - last rx buffer, (not applicable to xx76)
     [s]avebuf- last rx buffer with error
     [S]eqlist- list of missed frame sequences from last test
     [t]xbuf  - last tx buffer
     [T]imers - last starting and ending time structures
     [v]clist - isartest connection structures and statistics
[h]elp    - print this list
[o]am
     [c]lose  - close oam for a connection
     [d]isVC  - suspend OAM on a connection
     [D]isRS  - suspend OAM at the hardware
     [e]nVC   - wake up OAM for a connection
     [E]nRS   - wake up OAM at the hardware
     [o]pen   - open OAM for a connection
     [t][T]   - transmit an OAM frame on a connection
     [x]it    - leave this menu section
[p]hy - Phy API functions
     [c]ell   - query cell registers
     [e]nable - enable a specific phy
     [g]en    - query general registers
     [i]nfo   - query information registers
     [I]ntr   - query interrupt registers
     [m]ask   - query interrupt mask registers
     s[o]net  - query sonet registers
     [s]tats  - query statistic registers
     [S]tatus - query status registers
     [t]oggle - switch the phy's
     [x]it    - leave this menu section
[P]arams - modify operation parameters
     [e]cho    - toggle the echo flag for the [v]->[t]x function
     [f]lag   - set operations flag
          bit 0  (1)  =verbose print
          bit 1  (2)  =verbose interface, trace enable
          bit 2  (4)  =increment frame size
          bit 3  (8)  =random frame size
          bit 4 (16)  =generate/compare data
          bit 8 (256) = receive raw, used if target is not running isartest
     [i]nfoTmo- base tmo for [t]est when requesting information from target
     [l]en    - set the base frame length
     [L]ogfile- change current logfile name
     [m]tu    - set the max mtu size for the test functions
     [s]eed   - set seed value for data pattern
     [t]xrxflow- set base limit for flow control for auto echo tests
     [T]xQfull- set base limit for number of queue full conditions
[q]uery - Query API functions
     [a]dap   - get adapter information (serial #, mac, pci, etc.)
     [d]river - get driver revision information
     [h]wconf - get hardware information needed by isarxbr for vbr
calculations
     [l]ink   - get driver configuration for a link
     [L]ib    - get library revision information
     [p]ort   - get port physical information
     link[s]tat- get statistics for a link
     [x]it    - leave this menu section
[t]ests - select/run automatic tests, Upper case selections enable Verbose
mode
     [b][B] tx   xMB on all vc's frame 1024
```

```
     [c][C] echo xMB on all vc's frame 1024
     [d][D] tx   xMB on random vc's frame 1024
     [e][E] echo xMB on random vc's frame 1024
     [o][O] open-send-close
     [v][V] vc enable/disable or close/open
[v]c - open or close connections
     [a]ctvc  - query for active connections
     [c]lose  - close connections
          [a]ll open vc's - close isartest connections and any in the driver
          [s]elect vc's - select specific connections to close
     [d]isable- suspend rx/tx for a connection
     [e]nable - wake up rx/tx for a connection
     [i]nfo   - query configuration info for a connection
     [o]pen    - open connections
          [e]tc/atm/isarconf.x - parse config file for <PVC> entries
          [f]ile - enter path/filename to parse for <PVC> entries
          [m]anual
             Current parameters:
             QOS: UBR
             TYPE: AAL5
             PCR: 352768
             MCR: 0
             TQD: 0
             ENABLE NOW, NO OAM LOCK...change(y/n)?
             [f]lag
                [d]eferred [i]mmediate [l]ock oam [o]am enable
             [q]os
                [c]br [u]br vbr[0] vbr[1] vbr[2]
             [r]ate
                xbr[1] xbr[2] [t]qd
             [t]ype
                aal[0] aal[5]
          Base VPI:
          Count   :
          Base VCI:
          Count   :
     [s]tats  - query statistics for a connection
     [t][T]   - transmit a frame for one or all connections
     [x]it    - leave this menu section
[x]it     - exit this program and vclose all vc's
```

# Requirements

`isartest` is designed to have either a 4575 or a 4576 adapter on the target machine with `isartest` running or with a loopback cable installed.

If the target machine is not running `isartest` then certain options such as automatic data compare, automatic packet echo, packet sequence number checking, and target machine information or statistics will not be available. To disable receive checking, modify the [`p]arameters-[f]lag` variable to set bit 8 (256). In this mode only the VPI, VCI, and length from the receive indication will be displayed, if verbose is enabled.

# Getting Started

Decide if you want a log file and where the log file is to be located. (i.e. `/tmp/testlog`). `isartest` will prompt you for this information. The logging information is intended to include errors and test results. However, if the verbose option is enabled while in the application, additional information will be logged.

For the 4576 and the initialization function, you have to decide if you are going to point the `isartest` to a configuration file or enter the parameters in the `isartest` application. If you are using a configuration file it must be constructed with the following key words (reference the `isarconf.0` file in the package):

| | |
|---|---|
| Link 0 | # link number |
| MTU 4096 | # maximum CPCS-PDU size |
| SchPCR | Value to set the line cell rate for the adapter and CBR scheduling |
| SchMCR | Value to use for minimum cell rate for CBR scheduling |
| SegVccs | 32768 # range from 64 to 65536, even numbers |
| SegQSize | 1024 # valid values are 64, 256, 1024, 4096 |
| RsmQSize | 1024 # valid values are 64, 256, 1024, 4096 |
| InitFlags | 0 # sonet, idle CLP = 0, txclk source = local |

# Construct the VPI and VCI counts for the driver to set up on the 4576 card

# VPI <value> <count> (count must be modulo 128)

VPI 0 1024

VPI 1 128

VPI 2 128

The application usage is "`isartest`".

```
>isartest
Enter board number: 0
Set up Log File(y/n)?n
isartest for product Version x.y platform.z
```

Where `product` is either 4576 or 5576, `x` is the major release version of the package, `y` is the minor release and `z` is the build number. The `platform` notations are `sun`, `vxw`, and `linux` for Solaris, VxWorks, and Linux.

`isartest` will display the adapter number, current host information, and board type before going to the main menu.

```
Bd#0 SunOS eng-lab-062 5.8 Generic_108528-11 sun4u sparc SUNW,Ultra-60 4576
```

## Main Menu

The following menu is displayed at the beginning and end of any function:

```
isartest for product Version x.y platform.z
[a]utoTest [c]ontrol [d]isplay [h]elp [o]am [p]arams [q]uery [t][T]x [v]c [x]it
```

## Description of Basic Operation

The default flag settings are to enable general verbose mode along with data compare and generation, no echo, and a 1024 byte frame size.

The basic flow of operations is to configure (`init` function in the `[c]ontrol menu`) the driver, open a connection (`open` function in the `[v]c menu`) on the host and target machine, enable the echo function, and then transmit a frame. The sending machine will display the transmit parameters followed by the results, including timing.

```
[a]larm [c]ontrol [d]isplay [h]elp [o]am [p]hy [P]arams [q]uery [t]ests [v]c [x]it
v
VCC Menu
[a]ctvc [c]lose [e]nable [d]isable [i]nfo [o]pen [s]tats [t][T]x [x]it:
T
[a]ll [s]elect: a
TX 1024 bytes...vpi 0 vci 32 seq 0 len 1024 flag 0x205 seed 0x49534152
rx 1024 bytes...vpi 0 vci 32 seq 3 len 1024 flag 0x205 seed 0x49534152
```

The target machine will display the receive indication and, if echo enabled, will show a transmit.

```
rx 1024 bytes...vpi 0 vci 32 seq 3 len 1024 flag 0x205 seed 0x49534152
TX 1024 bytes...vpi 0 vci 32 seq 0 len 1024 flag 0x204 seed 0x49534152
```

All frame data includes a message header that identifies the connection parameters; VPI and VCI, frame length, operation flags, sequence number, and data pattern seed.

# Menu Selections

This section describes in detail, each of the Main Menu selections in alphabetic order.

## [a]larm

This section is used to execute any of the Alarm APIs. The alarm handler in this application only reports events. For a full alarm interface with sanity checking, logging, and history functions, run the package application "`isaralarm`".

Alarm Menu

```
[C]lear [c]lose [d]isable [e]nable [i]nfo [o]pen [t]une [x]it:
```

### [C]lear

This selection is used to manually clear an alarm event.

```
Enter State value to issue(0=none):
```

### [c]lose

This selection will close down the alarm interface.

### [d]isable

This selection will suspend alarm functions.

### [e]nable

This selection will wake up alarm functions.

### [i]nfo

This selection will query the driver for the current alarm parameters. For the entries ending with ".a.v" the values indicate the last open or tune parameters for action and value.

```
State     : 0x3
RxCrcs    : 0x0
RxLens    : 0x0
RxTmos    : 0x0
Mask.a,v  : 0x01, 0x0ffffed0
RxCrc.a,v : 0x01, 0x00000001
RxLen.a,v : 0x01, 0x00000001
RxTmo.a,v : 0x01, 0x00000001
K1K2 bytes: 0x00, 0x00000000
ApsSigF.a,v: 0x00, 0x00000000
Oflag     : 0x3
IndArg    : 0x616c726d
IndTag    : 0x414c524d
IndFunc   : 0x13ca0
PhyState0 : 0x296d
PhyState1 : 0x0
VpiVci    : 0x0
VCID      : 0x0
```

### [o]pen

This selection is used to open and set the tuning parameters for the alarm interface. The oflag is bit oriented.

```
(0)Global (1)enable now (2)ind clear
oflag: 0...modify?(y/n): y
oflag = 3
Tune Param : value    mode
[d]efault   :
[m]ask     : ffffed0 0
rx[c]rcThresh: 1        0
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
d
Tune Param : value    mode
[d]efault  :
```

```
[m]ask     : ffffed0 1
rx[c]rcThresh: 1        1
rx[l]enThresh: 1        1
rx[t]moThresh: 1        1
e[x]ecute cmd
x
```

### [t]une

This selection is used to modify existing alarm parameters.

```
Tune Param : value    mode
[d]efault  :
[m]ask     : ffffed0 0
rx[c]rcThresh: 1        0
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
c
[i]gnore [s]et: s
value = 2
Tune Param : value    mode
[d]efault  :
[m]ask     : ffffed0 0
rx[c]rcThresh: 2        1
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
x
```

### [x]it

Leave this menu section.

## [c]ontrol

This section is used to execute any of the control APIs.

```
Control Menu

[i]nit [l]oopback [r]sTune [t]race [u]nld [x]it
```

### [i]nit

This selection is used to initialize the interface and the driver. The user is prompted to either point to a configuration file or select default settings. In either case the user can still modify the parameters before the I/O is issued.

```
[d]efault [e]tc/atm/isarconf.x [f]ile : f

Path/name of isarconf file: isarconf.tst

Display parameters(y/n)?:y

VPI Entries

vpi 0 vci count 1024
```

```
vpi 1 vci count 128
Current Initialization Parameters:
MTU       : 4096
Flags     : 0
SegVccs   : 128
SegQsize  : 256
SchPCR    : 354000
SchMCR    : 50
RsmQsize  : 256
VccEntry  : 2

Change (y/n)?n
```

If the card has already been initialized the following will be displayed:

```
Adapter already initialized...moving info
CTRL 513 initialize link bad status...IA_ERR_INITED(2055): Link already init'ed
```

This is not an error as it is mandatory for applications to issue the init in order to synchronize with the API and driver.

## [l]oopback

This selection allows loopback mode to be enabled or disabled.

## [r]sTune

This selection is used to modify the interrupt interface for the hardware. The following shows a set up for one interrupt per 8 status queue entries with a backup timer of 1 millisecond and a transmit fifo size of 8.

```
TUNE
[c]ount [f]ifo [t]imer e[x]it: c
last_value: 0
[d]efault [o]ff [s]et: s
cnt = 8
TUNE
[c]ount [f]ifo [t]imer e[x]it: t
last_value: 0
[d]efault [o]ff [s]et: s
tmr = 1000
TUNE
[c]ount [f]ifo [t]imer e[x]it: f
last_value: 0
[d]efault [o]ff [s]et: s
cnt = 8
TUNE
[c]ount [f]ifo [t]imer e[x]it: x
```

### [t]race

This selection allows trace mode to be enabled or disabled.

### [u]nld

This selection is used to cause the driver to return to an un-initialized state. All resources are released and the hardware is reset.

### [x]it

Leave this menu section.

## [d]isplay

This selection is used to display the current transmit parameters, the receive data buffer, the saved receive data buffer, or the transmit data buffer. The receive data buffer is only valid for the 4575 as the 4576 buffers are owned by the API. The saved data buffer is used by the application to save buffers that have encountered error conditions such as data miscompares.

```
Display

[c]onn [p]armeters [r]xbuf [s]avebuf [S]eqList [t]xbuf [T]imers [v]clist
```

### [c]onn

This selection displays all the open connections in the driver.

```
Connection Information
vpi     vci     state   type    mode    txQd    txInuse
0       32      0x3     5       1       0       0
0       33      0x3     5       1       0       0
```

### [p]aramters

This selection displays the current `isartest` parameters.

```
Auto flag  0

Burst cnt  1

Echo flag  0

Flag       21

Frame Size 1024

Iteration  1

MTU max    4096

PCR       354000

Seed       0x49534152
```

### [r]xbuf

This selection displays the last receive buffer (only valid for 4575).

### [s]avebuf

This selection displays the last receive that encountered an error.

### [S]eqList

This selection displays the last test missed sequence list.

```
vpi  vci  expected received
0    32   00000005 00000006
```

### [t]xbuf

This selection displays the last transmit buffer.

### [T]imers

This selection displays the last test starting and ending time structures.

### [v]clist

`[v]clist` displays information from the application for all open connections.

```
VPI     VCI     Mode    Type    PCR     TXQ     OAM     VCID
0       32      1       5       0       0       0       49000000
        Api     Tst     Rsrc    Flow    Echo    Ack     OAM     Bytes
TX      15      8       0       0       0       0       0       8192
        Api     Tst     Drop    CmpErr  Echo    Ack     OAM     Bytes
RX      0       0       0       0       0       0       0       0
        Max     Cnt     Hd      Tl
ELST    0       0       0       0
-----------------------------------------------
VPI     VCI     Mode    Type    PCR     TXQ     OAM     VCID
0       33      1       5       0       0       0       49000001
        Api     Tst     Rsrc    Flow    Echo    Ack     OAM     Bytes
TX      0       0       0       0       0       0       0       0
        Api     Tst     Drop    CmpErr  Echo    Ack     OAM     Bytes
RX      0       0       0       0       0       0       0       0
        Max     Cnt     Hd      Tl
ELST    0       0       0       0
-----------------------------------------------
```

These parameters are automatically cleared at the beginning of any autotest test function. The `apitx` and `apirx` values are from the API-issued connection statistics. The `Tst` values are from the application counts.

## [h]elp

This selection prints the basic help information.

## [o]am

This selection is used to execute all of the OAM API functions.

```
OAM Menu
[c]lose [d]isVC [D]isRS [e]nVC [E]nRS [o]pen [t][T]x [x]it:
```

### [c]lose

This selection is used to close out OAM for a connection.

### [d]isVC

This selection is used to suspend receive and transmit processing of OAM frames for a connection.

### [D]isRS

This selection is used to disable OAM processing at the hardware level.

### [e]nVC

This selection is used to wake up receive and transmit OAM functions for a connection.

### [E]nRS

This selection is used to enable OAM processing at the hardware level.

### [o]pen

This selection is used to open OAM processing for a connection.

### [t][T]x

This selection is used to send an OAM frame on a connection. Note that the payload will not contain data of an OAM format, but will contain diagnostic data used by `isartest`.

```
vpi: 0
vci: 32
[1]f4seg [2]f4e2e [3]f5seg [4]f5e2e: 3
TX OAM 48 bytes....vpi 0 vci 32 seq 0 len 48 flag 0x1 seed 0x0
rx OAM 48 bytes...vpi 0 vci 32 seq 0 len 48 flag 0x81 seed 0x0
F5SEG Rx
OAM ECHO Req
```

### [x]it

Leave this menu section.

## [p]hy

This selection is used to execute all of the phy API functions.

```
PHY Menu
[c]ell [e]nable [g]en    [i]nfo   [I]ntr [m]asks
s[o]net [s]tats [S]tatus [t]oggle [x]it:
```

### [c]ell

This selection will display the phy cell registers.

```
Cell Registers
cgen      : 0x60
cval      : 0x70
idlpay    : 0x6a
idlmsk    : 0x1
rxhdr     : 0x0
rxidl     : 0x1
rxmsk     : 0xffffffff
txhdr     : 0x0
txidl     : 0x0
```

### [e]nable

This selection is used to enable a specific phy as the “working” phy and implicitly make the other phy “standby”.

```
Select PHY (0-1)?: 0
```

### [g]en

This selection will display the phy general registers.

```
General Registers
gen       : 0x44
clkrec    : 0x0
outstat   : 0x0
version   : 0x77
utop1     : 0x18
utop2     : 0x0
udf2      : 0x0
```

### [i]nfo

This selection will display the current working phy number, phy count and phy information registers.

```
PHY Information
count      : 1
active     : 0
version(1-0): 0x0 - 0x77
state (1-0): 0x0 - 0x296d
```

### [I]ntr

This selection will display the phy interrupt registers.

```
Interrupt Registers
```

```
sumint    : 0x0
secint    : 0x0
linint    : 0x0
pthint    : 0x0
txcellint : 0x0
rxcellint : 0x0
apsint   : 0x0
```

## [m]asks

This selection will display the phy interrupt mask registers.

```
Mask Registers
ensumint  : 0xe2
ensec     : 0xf8
enlin     : 0xb0
enpth     : 0xc0
encellt   : 0x0
encellr   : 0x80
enaps     : 0x0
```

## s[o]net

This selection will display the phy SONET registers.

```
Sonet Registers
txsec     : 0x0
txlin     : 0x3
txpth     : 0x23
txk1      : 0x0
txk2      : 0x0
txs1      : 0x0
txc2      : 0x13
txz01     : 0x2
txz02     : 0x3
apsthresh : 0x36
rxk1      : 0x0
rxk2      : 0x0
rxs1      : 0x0
rxc2      : 0x13
rxg1      : 0x2
rxz01     : 0x2
rxz02     : 0x3
```

## [s]tats

This selection will display the phy statistic registers.

```
PHY Statistics
locd      : 1
heccor    : 0
hecunc    : 0
oof       : 0
secbip    : 0
linbip    : 0
pthbip    : 0
linrei    : 0
pthrei    : 0
```

```
nomtch    : 0
txcells   : 24
rxcells   : 24
```

### [S]tatus

This selection will display the phy status registers.

```
PHY Status
rxaps     : 0
rxcell    : 12
rxline    : 64
rxpath    : 0
rxsect    : 128
txcell    : 8
```

### [t]oggle

This selection will switch the phy configuration by moving the working phy to standby status and moving the current standby phy to working status.

### [x]it

Leave this menu section.

## [P]arams

Operation Parameters:

```
[e]cho [f]lag [i]nfoTmo [l]en [L]ogfile
[m]tu [s]eed [t]xrxflow [T]xQfull :
```

### [e]cho

This selection toggles the echo function on and off. The new echo state will be displayed. If enabled the `[v]->[t][T]` selection will expect each sent frame to be returned by the target.

### [f]lag

This selection is used to set various operating and transmit functions. The flag is binary in nature and requires the user to add the desired selection values and enter the total as the flag selection.

```
bit 0  (1)  =verbose print
bit 1  (2)  =verbose interface, trace enable
bit 2  (4)  =increment frame size
bit 3  (8)  =random frame size
bit 4 (16)  =generate/compare data
bit 8 (256) =receive raw mode
```

```
     +____
Flag = 21 ..change(y/n)?
```

#### bit 0 - verbose print

This bit is for general purpose verbose printing.

#### bit 1 - verbose interface, trace enable

This bit is used to specify individual connections for transmit or test functions and enables tracing in the 4576 API library.

#### bit 2 - increment frame size

This bit causes the transmit function to increment the frame size for each iteration.

#### bit 3 - random frame size

This bit causes the transmit function to use a random frame size for each iteration.

#### bit 4 - generate/compare data

This bit invokes data pattern generation for each transmit and data pattern compare for each receive. The data pattern algorithm is:

u32 data[0] = u32(u16 vpi - u16 vci) + flag + length + seed;

data[x] = data[x - 1] + seed;

#### bit 8 - receive raw mode

This bit indicates to the receive handler to not evaluate the packet data for the information structure used to communicate between two `isartest` stations. The receive function will simply print the VPI, VCI, and length information, if verbose, and free the buffer.

### [i]nfoTmo

This selection sets the base timeout for the information requests used in the autotest functions. The timeout is modified by the test based on the QoS mode and rate of the connection.

### [l]en

This selection sets the base frame size to use for the transmit and test functions.

### [L]ogfile

This selection is used to change the logfile name.

### [m]tu

This selection sets the maximum frame size boundary for the transmit and test functions.

### [s]eed

This selection is used to specify the seed value to use for data pattern generation and compare.

### [t]xrxflow

This selection sets the limit for the consecutive occurrences of a transmit being rejected in `isartest` because of a flow condition for the autotest echo selections. Flow is implemented based on `rsmQsize`, if the number of transmits minus the number of receives is greater than `rsmQsize - 4` then flow control is entered.

This parameter may need to be modified if using a large number of slow-rate connections.

### [T]xQfull

This selection sets the limit for the consecutive occurrences of a transmit being rejected by the driver because of a queue full condition.

This parameter may need to be modified if using a large number of slow-rate connections.

## [q]uery

This selection provides access to driver information and statistics.

```
Query Menu

[a]dap [A]larm [c]onn [d]river [h]wconf [l]ink [L]ib [p]ort link[s]tat [v]cstat
```

### [a]dap

This selection displays the adapter information.

```
Adapter Information
bd#  : 0
vendor: Interphase
model: 4576
bus  : PMC
name : SX00653-X01
devid : 0x823414f1
subid  : 0x10107e
clsrev : 0x2030005
hwrev  : 0x1080
serial : 0x1add44
ports  : 0x1
sram   : 0x401800
sar    : 0x5
phy    : 0x77
umac   : 00779add440000000000
fmac   : 00779add440000000000
```

### [d]river

This selection displays the driver name and revision information.

```
Driver Information
```

```
vendor : Interphase
desc   : x576 iSAR API driver
env    : SunOS 5.8 sparc 32-bit
version: 3.1 build dww.1561
bd cnt : 1
uptime : 1136532 seconds
```

### [h]wconf

This selection displays the hardware information necessary for application `isarxbr` when calculating VBR connection parameters.

```
RS8234 Configuration
Sysclk : 33000000 mhz
Clk/Slt: 94
TblSize: 2114
Txfifo : 8
IntQtmr: 1000
IntQcnt: 8
```

### [l]ink

This selection displays the link configuration information.

```
Link Information
bd#    : 0
port   : 0
state  : 0x1fff
mtu    : 4096
SchPcr : 351063
SchMcr : 166
SchBand: 0
SegVcc : 4096
RsmVcc : 1024
ActVcc : 2
SegQsz : 64
RsmQsz : 64
uptime : 1136590 seconds
```

### [L]ib

This selection displays the library revision information.

```
Library Information
vendor : Interphase
desc   : x576 iSAR API driver
env    : SunOS 5.8 sparc 32-bit
version: 3.1 build dww.1561
```

### [p]ort

This selection displays the port physical characteristics.

```
Port (0-1)? 0
Port Information
bd#    : 0
port   : 0
type   : 119
speed  : 155 Mbits/sec
```

```
conn   : SC-MM
state  : 0x296d
```

### link[s]tat

This selection displays the statistics for the link.

```
Link Stat Information
aal0In   : 0x0
aal0Out  : 0x0
pcktsIn  : 0xcca
pcktsOut : 0xc47
octetsIn : 0x0
octetsOut: 0x0
oamIn    : 0x0
oamOut   : 0x0
discards : 0x0
losCount : 0x0
uptime   : 1136748 seconds
```

### [x]it

Leave this menu section.

## [t]ests

This section provides a selection of tests to run for desired transfer sizes. The user-defined value for frame length and mtu size are used as baseline values. The current user frame size is displayed at the end of each selection. For all tx and echo selections, the user is prompted for overall transfer size, data compare and generation enable, and frame size manipulation function (fixed, incrementing, or random). The maximum frame size is determined by the user selection in the main menu for `[m]tu`.

NOTE: Setting a transfer length of 1 enables continuous mode, a ^c will cause a break into a support menu when in this mode.

The basic flow for the transmit and echo tests are as follows:

- Use the first VC opened as the information control VC (must be AAL5)
- Use the first 24-32 bytes of the data frames to communicate between machines
- Request target machine information and display
- For transmit only, request starting time from target
- Transmit and receive, if echo, test data
- For transmit only, request ending time
- Display overall statistics with throughput values
- Display individual connections if selected

At the completion of the transmit and echo tests, the host and target machine information is displayed along with total byte counts and time duration (throughput) information.

The following are the available selections:

```
TEST SELECTIONS
Use the upper case selection for Verbose
[b][B] tx   xMB on all vc's frame 1024
[c][C] echo xMB on all vc's frame 1024
[d][D] tx   xMB on random vc's frame 1024
[e][E] echo xMB on random vc's frame 1024
[o][O] open-send-close
[v][V] vc enable/disable
```

### [b][B] - Transmit only, sequential connections

Upper case selection enables verbose mode.

This selection is a transmit-only test for all open connections.

### [c][C] - Echo, sequential connections

Upper case selection enables verbose mode.

This selection transmits frames to the target and expects the target to echo the frames back. Each frame is checked for proper sequence number upon receipt at the target and host.

### [d][D] - Transmit only, random connections

Upper case selection enables verbose mode.

This selection is a transmit-only test on a random selection of open connections.

### [e][E] - Echo, random connections

Upper case selection enables verbose mode.

This transmits frames to the target and expects the target to echo the frames back for random selections of open connections.

Example for selection `[b]`:

```
Set Transfer size
i.e 16KB=16384, 32KB=32768, 64KB=65536
i.e 128KB=131072, 256KB=262144, 512KB=524288
i.e 1MB=1048576, 10MB=16777216, 100MB=268435456
8192
do data (y/n)?n
frame size [f]ixed, [i]incr [r]andom : f
------------------Test Parameters-----------------------------
Host:  Bd#0 SunOS lab-dhcp-104 5.9 Generic sun4u sparc SUNW,Ultra-60 4576
Target:Bd#0 SunOS lab-dhcp-162 5.9 Generic sun4u sparc SUNW,Ultra-60 4576

Base Frame Size: 1024 - Fixed
Data Compare   : Disabled
Test Data Size : 8192...0x2000
Duplex Mode    : Transmit Only
```

```
VC Count       : 1
VC Selection   : Sequential

------------------Test Results---------------------------------
Time: 0.003
        TxB         RxB       Total     MBsec    Mbsec    IOsec
        8192        0         8192      16.0     128      8

        TxFrm    TxOam    TxRty    Flow     RxFrm    RxOam    RxDrp    BadData
Target  8        0        0        0        8        0        0        0
Host    8        0        0        0        8        0        0        0

Display the individual connection results (y/n)?n
-------------------------------------------------------------
```

## [t][T]x

This selection will transmit one frame on either all of the connections or a selected one. The frame size and echo function are set in the parameters menu. The upper case selection is for verbose.

```
[a]utoTest [c]ontrol [d]isplay [h]elp [o]am [p]arams [q]uery [t][T]x [v]c [x]it
T
loop count set to 1
All connections(y/n/q)?y
------------------Test Parameters-----------------------------
Burst 1 iterations 1 flag 23 frame 1024 ttool_byte_cnt 1024

For all 2 VC's sequential
TX 1024 bytes....vpi 0 vci 32 seq 0 len 1024 flag 0x204 seed 0x49534152
TX 1024 bytes....vpi 0 vci 33 seq 0 len 1024 flag 0x204 seed 0x49534152

------------------TX Results----------------------------------
        TxFrm    TxOam    TxRty    Flow     RxFrm    RxOam    RxDrp    BadData
Host    2        0        0        0        0        0        0        0

Display the individual connection results (y/n)?n
-------------------------------------------------------------
```

## [v]c

This selection is used to execute all of the VCC API functions.
VCC Menu

```
[a]ctvc [c]lose [d]isable [e]nable [i]nfo [o]pen [s]tats [t][T]x [x]it:
```

### [a]ctvc

This selection will query for all active connections in the driver and display them. The txc entry represents the current number of transmits still in the queue.

```
index  vcid      vpi   vci   slot state  type  mode  tqt   txc
  0   49000000     0    32     0     7     5     1     0     0
  1   49000001     0    33     1     3     5     1     0     0
```

## [c]lose

This selection is used to close any or all currently opened connections. The user is prompted for base values and counts. Therefore, if the user wanted to close the first five VCs on the first VP.

```
[a]ll open VC's  [s]elect VC's : s

Base VPI: 0

Count   : 1

Base VCI: 1

Count   : 5
```

## [d]isable

This selection is used to suspend receive and transmit functions for a connection.

## [e]nable

This selection is used to wake up receive and transmit functions for a connection.

## [i]nfo

This selection will query the driver for connection configuration information and display.

```
vpi: 0
vci: 32
Connection Information
index     : 0
vcid      : 0x49000000
slot      : 0
state     : 0x7
vpi       : 0
vci       : 32
type      : 5
mode      : 1
txQd      : 0
txInuse   : 0
rcvInd    : 0x1c6a8
rcvArg    : 32
oamInd    : 0x15980
oamArg    : 32
uptime    : 644 seconds
```

## [o]pen

This selection is used to specify connections to open. The selections can be entered manually or by selecting a file to parse <PVC> entries from. The format is explained in the `/etc/atm/isarconf.x` file.

```
[e]tc/atm/isarconf.x [f]ile [m]anual :
```

If the manual selection is used, the user is prompted for a base VPI and VCI value followed by a count. Therefore, if the user wanted to open the two connections on VPI 0 starting with VCI 32:

```
Current parameters:
QOS: UBR
TYPE: AAL5
PCR: 354000
MCR: 50
TQD: 0
ENABLE NOW - NO OAM LOCK...change(y/n)? n
Base VPI: 0
Count   : 1
Base VCI: 32
Count   : 2
VCOPEN
vcopen: vpi 0 vci 32 vcid=0x49000000 status=0x0
      : mode 1 type 5 xbr1 0 xbr2 0
-----------------------------------------------
vcopen: vpi 0 vci 33 vcid=0x49000001 status=0x0
      : mode 1 type 5 xbr1 0 xbr2 0
-----------------------------------------------
```

## [s]tats

This selection will query for statistics on a connection and display.

```
vpi: 0
vci: 32
VCStat Information
vcid      : 0x49000000
aal0In    : 0x0
aal0Out   : 0x0
pcktIn    : 0x0
pcktOut   : 0x2
byteIn    : 0x0
byteOu    : 0x0
oamIn     : 0x2
oamOut    : 0x2
discard   : 0x0
rxcrc     : 0x0
rxlen     : 0x0
rxtmo     : 0x0
uptime    : 1279 seconds
```

## [t][T]x

This selection is used to transmit a single frame on one or all of the connections. The frame length is set in the [P]arameters menu.

```
vpi: 0
vci: 32
TX 1024 bytes....vpi 0 vci 32 seq 0 len 1024 flag 0x0 seed 0x0

If the [P]armeters->[e]cho flag is set or loopback cable installed....

rx 1024 bytes...link 0 vpi 0 vci 32 seq 0 len 1024 flag 0x80 seed 0x0
```

### [x]it

Leave this menu section.

## [x]it

This selection will close all open connections and exit the application.

# ATM isaralarm Guide

# B

## Introduction

### Purpose

The purpose of this appendix is to provide information on the usage and functions of the ATM API application `isaralarm`. This application is used to exercise the cards alarm and phy interface. The adapter must be initialized before running this application.

This tool is considered an engineering-level diagnostic and is subject to change without notice.

## isaralarm Menu Tree

The following is a representation of the `isaralarm` menu tree:

```
[a]larm - Alarm API Functions
          [C]lear- clear an alarm state
          [c]lose- close the alarm interface
          [d]isable- suspend the alarm interface
          [e]nable- wake up the alarm interface
          [i]nfo - query the current alarm structure
          [o]pen - open and configure the alarm interface
          [t]une - modify the parameters for the alarm interface
          [x]it  - leave this menu section
[d]isplay - parameters or rxbuf or txbuf
          [a]larm- show alarm history
                 [a]larms - select a specific historic alarm
                 [s]tates - show the last 16 alarm state fields
                 [v]cid - show the last 16 alarm vcid fields
          [c]onn - current open connections in driver
          [e]rrbuf- last rx buffer with error
          [f]lag - isaralarm flag variable
          [h]ost - display host information
          [p]hy  - detailed, bit level, display of current and last phy
states
          [s]tat - isaralarm statistics
          [t]xbuf- last tx buffer
          [v]cs  - isaralarm connection structures and statistics
[h]elp    - print this list
[o]am
          [c]lose- close oam for a connection
          [d]isVC- suspend OAM on a connection
          [D]isRS- suspend OAM at the hardware
          [e]nVC - wake up OAM for a connection
          [E]nRS - wake up OAM at the hardware
          [o]pen - open OAM for a connection
          [t][T] - transmit an OAM frame on a connection
          [x]it  - leave this menu section
```

```
[p]hy - Phy API functions
          [c]ell - query cell registers
          [e]nable- enable a specific phy
          [g]en  - query general registers
          [i]nfo - query information registers
          [I]ntr - query interrupt registers
          [m]ask - query interrupt mask registers
          s[o]net- query sonet registers
          [s]tats- query statistic registers
          [S]tatus- query status registers
          [t]oggle- switch the phy's
          [x]it  - leave this menu section
[P]arams - modify operation parameters
          [f]lag - set operations flag
                 bit 0  (1)  =verbose print
                 bit 1  (2)  =verbose interface, trace enable
          [l]ogfile- change current logfile name
          [v]cdebounce- set base limit for handling connection alarm
                 [d]uration - total time to debounce
                 [i]nterval - how long between each check of the driver
[q]uery - Query API functions
          [a]dap - get adapter information (serial #, mac, pci, etc.)
          [d]river- get driver revision information
          [h]wconf- get hardware information needed by isarxbr for vbr
calculations
          [l]ink - get driver configuration for a link
          [L]ib  - get library revision information
          [p]ort - get port physical information
          link[s]tat- get statistics for a link
          [x]it  - leave this menu section
[x]it     - exit this program and vclose all vc's
```

# Requirements

`isaralarm` is designed to be run on an adapter that is already initialized. There are no control APIs available to this application. The application `isartest` can be used to initialize the adapter.

# Getting Started

Decide if you want a log file and where the log file is to be located. (i.e. `/tmp/testlog`). `isaralarm` will prompt you for this information. The logging information is intended to include errors and test results. However, if the verbose option is enabled while in the application, additional information will be logged.

The application usage is "`isaralarm`".

```
>isaralarm

Enter board number: 0

Set up Log File(y/n)?n

Verbose (y/n) ?n

isartest for product Version x.y platform.z
```

Where `product` is either 4576 or 5576, `x` is the major release version of the package, `y` is the minor release and `z` is the build number. The `platform` notations are `sun`, `vxw`, and `linux` for Solaris, VxWorks, and Linux.

`isaralarm` will display the adapter number, current host information, and board type followed by the number of connections that it has discovered already open in the driver before going to the main menu.

```
Bd#0 SunOS eng-lab-062 5.8 Generic_108528-11 sun4u sparc SUNW,Ultra-60 4576
```

Created x new vc entries

## Main Menu

The following menu is displayed at the beginning and end of any function:

```
isaralarm for product Version x.y platform.z
[a]larm [d]isplay [h]elp [o]am [p]hy [P]arams [q]uery [x]it
```

## Description of Basic Operation

The basic flow of operations is to open the alarm interface and then wait for asynchronous events.

```
PHY0 ALARM0: state 0x00000043 phy0 0x0003 phy1 0x0000
```

All alarms received will at least print out the above indication, with verbose enabled, the full alarm structure is displayed.

# Menu Selections

This section describes in detail, each of the Main menu selections in alphabetic order.

## [a]larm

This section is used to execute any of the Alarm APIs.

Alarm Menu

```
[C]lear [c]lose [d]isable [e]nable [i]nfo [o]pen [t]une [x]it:
```

### [C]lear

This selection is used to manually clear an alarm event.

Enter State value to issue (**0**=none):

### [c]lose

This selection will close the alarm interface.

## [d]isable

This selection will suspend alarm functions.

## [e]nable

This selection will wake up alarm functions.

## [i]nfo

This selection will query the driver for the current alarm parameters. For the entries ending with "`.a.v`" the values indicate the last open or tune parameters for action and value.

```
State            : 0x3
RxCrcs           : 0x0
RxLens           : 0x0
RxTmos           : 0x0
Mask.a,v         : 0x01, 0x0ffffed0
RxCrc.a,v        : 0x01, 0x00000001
RxLen.a,v        : 0x01, 0x00000001
RxTmo.a,v        : 0x01, 0x00000001
K1K2 bytes       : 0x00, 0x00000000
ApsSigF.a,v      : 0x00, 0x00000000
Oflag            : 0x3
IndArg           : 0x616c726d
IndTag           : 0x414c524d
IndFunc          : 0x13ca0
PhyState0        : 0x296d
PhyState1        : 0x0
VpiVci           : 0x0
VCID             : 0x0
```

## [o]pen

This selection allows the user the open and set the tuning parameters for the alarm interface. The `oflag` is bit oriented.

```
(0)Global (1)enable now (2)ind clear
oflag: 0...modify?(y/n): y
oflag = 3
Tune Param : value    mode
[d]efault   :
[m]ask       : ffffed0 0
rx[c]rcThresh: 1        0
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
d
Tune Param : value    mode
[d]efault  :
[m]ask     :  ffffed0 1
rx[c]rcThresh: 1        1
rx[l]enThresh: 1        1
rx[t]moThresh: 1        1
e[x]ecute cmd
```

```
x
```

### [t]une

This selection is used to modify existing alarm parameters.

```
Tune Param : value    mode
[d]efault  :
[m]ask     :   ffffed0 0
rx[c]rcThresh: 1        0
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
c
[i]gnore [s]et: s
value = 2
Tune Param : value    mode
[d]efault  :
[m]ask     :   ffffed0 0
rx[c]rcThresh: 2        1
rx[l]enThresh: 1        0
rx[t]moThresh: 1        0
e[x]ecute cmd
x
```

### [x]it

Leave this menu section.

## [d]isplay

This selection is used to display the alarm history, driver connections, and bit level phy states.

```
Display Menu

[a]larm [c]onn [e]rrbuf [f]lag [h]ost [p]hy [s]tat [t]xbuf [v]cs
```

### [a]larm

#### [a]larm

This selection is used to specify which historical alarm structure to display.

```
Last Alarm Count: 9
Enter Index to Display: 3
Header
status     : 0x0
type       : 0x503
link       : 0x0
bytesOut   : 0x98
bytesIn    : 0x98
tag        : 0x69534152
State      : 0x2e0003
RxCrcs     : 0x0
RxLens     : 0x0
RxTmos     : 0x0
```

```
Mask.a,v   : 0x01, 0x0ffffed0
RxCrc.a,v  : 0x01, 0x00000001
RxLen.a,v  : 0x01, 0x00000001
RxTmo.a,v  : 0x01, 0x00000001
K1K2 bytes : 0x00, 0x00000000
ApsSigF.a,v: 0x00, 0x00000000
Oflag      : 0x3
IndArg     : 0x616c726d
IndTag     : 0x414c524d
IndFunc    : 0x13ab0
PhyState0  : 0xa12d
PhyState1  : 0x0
VpiVci     : 0x21
VCID       : 0x49000001
```

### [s]tate

This selection will display the states from the last 16 alarm events.

```
Event#  event           phy0    phy1
8       0x08000003      0x2003  0x0000
7       0x02000003      0xa803  0x0000
6       0x00480003      0x2a03  0x0000
5       0x00220003      0xa943  0x0000
4       0x00400003      0x296d  0x0000
3       0x002e0003      0xa12d  0x0000
2       0x0a000003      0x8803  0x0000
1       0x000e0003      0x8003  0x0000
0       0x00000043      0x0003  0x0000
```

### [v]cid

This selection will display the `vcid` from the last 16 alarm events.

```
Event#  vpi     vci     vcid
8       0       33      0x49000001
7       0       33      0x49000001
6       0       33      0x49000001
5       0       33      0x49000001
4       0       33      0x49000001
3       0       33      0x49000001
2       0       33      0x49000001
1       0       33      0x49000001
0       0       33      0x49000001
```

## [c]onn

This selection displays all the open connections in the driver.

```
Connection Information
vpi     vci     state   type    mode    txQd    txInuse
0       32      0x3     5       1       0       0
0       33      0x3     5       1       0       0
```

## [e]rrbuf

This selection will display that last OAM receive buffer that had an error.

## [f]lag

This selection displays the current `isaralarm` flag variable.

```
Flag      0
Frame Size 48
MTU max    4096
```

## [h]ost

This selection displays the host information string.

```
Bd#0 SunOS lab-dhcp-104 5.9 Generic sun4u sparc SUNW,Ultra-60 4576
```

## [p]hy

This selection displays the last and current phy states with bit level descriptions.

```
bit:      <<-0..3->>   <<-4..7->>  <<-8.11->>     <<-12.15->>
          p  s  l  l  l  o  l  l  a  r  a  r  p  d  f  k
          r  i  o  o  o  o  o  o  i  d  i  d  s  e  a  1
          e  g  c  s  l  f  f  p  s  i  s  i  b  g  i  k
             d                    l  l  p  p  f  r  l  2
--------------------+-------------------+--------------------
last0:    1  1  0  0  0  0  0  0  0  0  0  1  0  1  0  1
curr0:    1  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0
```

## [s]tat

This selection displays the `isaralarm` statistics and current phy states.

```
ALARM Statistics
current phy: 0
phy changes: 0
last phy0 s: 0xa803
last phy1 s: 0x0000
curr phy0 s: 0x2003... present, signal detect true
curr phy1 s: 0x0000... Unavailable
no event   : 1
pci        : 0
vc add     : 1
vc delete  : 0
que valid  : 0
seg unflow : 0
rsm unflow : 0
rsm ovflow : 0
rsm crc    : 0
rsm len    : 0
rsm tmo    : 0
sig detect : 3
phy locd   : 2
phy los    : 3
phy lol    : 0
phy oof    : 2
phy lof    : 2
phy lop    : 0
phy ais l  : 0
phy rdi l  : 2
phy ais p  : 0
```

```
phy rdi p  : 2
phy psbf   : 0
phy sig d  : 0
phy sig f  : 0
k1k2 byte  : 0
```

### [t]xbuf

This selection displays the last transmit buffer.

### [v]clist

This selection displays information from the application for all open connections.

```
VPI     VCI     VCID
0       32      49000000
        Api     OAM     Unfl    Rsrc
TX      0       0       0       0
        Api     OAM     Unfl    Ovfl    CRC     Len     TMO     CmpErr
RX      0       0       0       0       0       0       0       0
-----------------------------------------------
VPI     VCI     VCID
0       33      49000001
        Api     OAM     Unfl    Rsrc
TX      0       0       0       0
        Api     OAM     Unfl    Ovfl    CRC     Len     TMO     CmpErr
RX      0       0       0       0       0       0       0       0
-----------------------------------------------
```

## [h]elp

This selection prints the basic help information.

## [o]am

This selection is used to execute all of the OAM API functions.

```
OAM Menu

[c]lose [d]isVC [D]isRS [e]nVC [E]nRS [o]pen [t][T]x [x]it:
```

### [c]lose

This selection is used to close out OAM for a connection.

### [d]isVC

This selection is used to suspend receive and transmit processing of OAM frames for a connection.

### [D]isRS

This selection is used to disable OAM processing at the hardware level.

### [e]nVC

This selection is used to wake up receive and transmit OAM functions for a connection.

### [E]nRS

This selection is used to enable OAM processing at the hardware level.

### [o]pen

This selection is used to open OAM processing for a connection.

### [t][T]x

This selection is used to send an OAM frame on a connection. Note that the payload will not contain data of an OAM format, but will contain diagnostic data used by `isartest`.

```
vpi: 0
vci: 32
[1]f4seg [2]f4e2e [3]f5seg [4]f5e2e: 3
TX OAM 48 bytes....vpi 0 vci 32 seq 0 len 48 flag 0x1 seed 0x0
rx OAM 48 bytes...vpi 0 vci 32 seq 0 len 48 flag 0x81 seed 0x0
F5SEG Rx
OAM ECHO Req
```

### [x]it

Leave this menu section.

## [p]hy

This selection is used to execute all of the phy API functions.

```
PHY Menu
[c]ell [e]nable [g]en    [i]nfo   [I]ntr [m]asks
s[o]net [s]tats [S]tatus [t]oggle [x]it:
```

### [c]ell

This selection will display the phy cell registers.

```
Cell Registers
cgen             : 0x60
cval             : 0x70
idlpay           : 0x6a
idlmsk           : 0x1
rxhdr            : 0x0
rxidl            : 0x1
rxmsk            : 0xffffffff
txhdr            : 0x0
txidl            : 0x0
```

## [e]nable

This selection is used to enable a specific phy as the “working” phy and implicitly make the other phy “stand by”.

```
Select PHY (0-1)?: 0
```

## [g]en

This selection will display the phy general registers.

```
General Registers
gen              : 0x44
clkrec           : 0x0
outstat          : 0x0
version          : 0x77
utop1            : 0x18
utop2            : 0x0
udf2             : 0x0
```

## [i]nfo

This selection will display the current working phy number, phy count and phy information registers.

```
PHY Information
count      : 1
active     : 0
version(1-0): 0x0 - 0x77
state (1-0): 0x0 - 0x296d
```

## [l]ntr

This selection will display the phy interrupt registers.

```
Interrupt Registers
sumint           : 0x0
secint           : 0x0
linint           : 0x0
pthint           : 0x0
txcellint        : 0x0
rxcellint        : 0x0
apsint           : 0x0
```

## [m]asks

This selection will display the phy interrupt mask registers.

```
Mask Registers
ensumint         : 0xe2
ensec            : 0xf8
enlin            : 0xb0
enpth            : 0xc0
encellt          : 0x0
encellr          : 0x80
enaps            : 0x0
```

## s[o]net

This selection will display the phy SONET registers.

```
Sonet Registers
txsec            : 0x0
txlin            : 0x3
txpth            : 0x23
txk1             : 0x0
txk2             : 0x0
txs1             : 0x0
txc2             : 0x13
txz01            : 0x2
txz02            : 0x3
apsthresh        : 0x36
rxk1             : 0x0
rxk2             : 0x0
rxs1             : 0x0
rxc2             : 0x13
rxg1             : 0x2
rxz01            : 0x2
rxz02            : 0x3
```

## [s]tats

This selection will display the phy statistic registers.

```
PHY Statistics
locd             : 1
heccor           : 0
hecunc           : 0
oof              : 0
secbip           : 0
linbip           : 0
pthbip           : 0
linrei           : 0
pthrei           : 0
nomtch           : 0
txcells          : 24
rxcells          : 24
```

## [S]tatus

This selection will display the phy status registers.

```
PHY Status
rxaps            : 0
rxcell           : 12
rxline           : 64
rxpath           : 0
rxsect           : 128
txcell           : 8
```

## [t]oggle

This selection will switch the phy configuration by moving the working phy to standby status and moving the current standby phy to working status.

### [x]it

Leave this menu section.

## [P]arams

Operation Parameters:

```
[f]lag [l]ogfile [v]cDebounce :
```

### [f]lag

This selection is used to set various operating and transmit functions. The flag is binary in nature and requires the user to add the desired selection values and enter the total as the flag selection.

```
bit 0  (1)  =verbose print

bit 1  (2)  =verbose interface, trace enable

+____

Flag = 21 ..change(y/n)?
```

#### bit 0 - verbose print

This bit is for general purpose verbose printing.

#### bit 1 - verbose interface, trace enable

This bit is used to specify individual connections for transmit or test functions and enables tracing in the 4576 API library.

### [l]ogfile

This selection is used to change the `logfile` name.

### [v]cDebounce

This selection sets the debounce parameters for handling connection alarms.

#### [d]uration

This sets the total time for the debounce.

#### [i]nterval

This sets interval timing for polling the driver during the debounce time.

## [q]uery

This selection provides the user access to driver information and statistics.

```
Query Menu
```

```
[a]dap [A]larm [c]onn [d]river [h]wconf [l]ink [L]ib [p]ort link[s]tat [v]cstat
```

## [a]dap

This selection displays the adapter information.

```
Adapter Information
bd#  : 0
vendor: Interphase
model: 4576
bus  : PMC
name : SX00653-X01
devid : 0x823414f1
subid  : 0x10107e
clsrev : 0x2030005
hwrev  : 0x1080
serial : 0x1add44
ports  : 0x1
sram   : 0x401800
sar    : 0x5
phy    : 0x77
umac   : 00779add440000000000
fmac   : 00779add440000000000
```

## [d]river

This selection displays the driver name and revision information.

```
Driver Information
vendor : Interphase
desc   : x576 iSAR API driver
env    : SunOS 5.8 sparc 32-bit
version: 3.1 build dww.1561
bd cnt : 1
uptime : 1136532 seconds
```

## [h]wconf

This selection displays the hardware information necessary for application `isarxbr` when calculating VBR connection parameters.

```
RS8234 Configuration
Sysclk : 33000000 mhz
Clk/Slt: 94
TblSize: 2114
Txfifo : 8
IntQtmr: 1000
IntQcnt: 8
```

## [l]ink

This selection displays the link configuration information.

```
Link Information
bd#    : 0
port   : 0
state  : 0x1fff
mtu    : 4096
SchPcr : 351063
```

```
SchMcr : 166
SchBand: 0
SegVcc : 4096
RsmVcc : 1024
ActVcc : 2
SegQsz : 64
RsmQsz : 64
uptime : 1136590 seconds
```

### [L]ib

This selection displays the library revision information.

```
Library Information
vendor : Interphase
desc   : x576 iSAR API driver
env    : SunOS 5.8 sparc 32-bit
version: 3.1 build dww.1561
```

### [p]ort

This selection displays the port physical characteristics.

```
Port (0-1)? 0
Port Information
bd#    : 0
port   : 0
type   : 119
speed  : 155 Mbits/sec
conn   : SC-MM
state  : 0x296d
```

### link[s]tat

This selection displays the statistics for the link.

```
Link Stat Information
aal0In   : 0x0
aal0Out  : 0x0
pcktsIn  : 0xcca
pcktsOut : 0xc47
octetsIn : 0x0
octetsOut: 0x0
oamIn    : 0x0
oamOut   : 0x0
discards : 0x0
losCount : 0x0
uptime   : 1136748 seconds
```

### [x]it

Leave this menu section.

## [x]it

This selection will close all open connections and exit the application.

# Specifications C

## Interface Specifications

| Item | Specification |
|---|---|
| Host Bus Interface | PCI Local Bus Revision 2.1<br>PMC P1386.1<br>32-bit, 33 MHz |
| PMC Form Factor | Length: 6.5 inches/165.1 mm<br>Width: 3.63 inches/92.2 mm |
| Memory | 4 or 8 MByte control memory |

### Power

| | 3.3 V | 5.0 V | Total |
|---|---|---|---|
| Single PHY | 0.30 A | 0.61 A | 4.5 W |
| Dual PHY | 0.35 A | 1.07 A | 7.0 W |

## Connectors/Cables

| Connector/Medium | Maximum Cable Length |
|---|---|
| SC Duplex (155 Mbps) | Single Mode Fiber (8.5/125)<br>Intermediate Reach<br>Maximum cable length: 9.3 miles/15 km<br>Minimum cable length: 9.8 ft/3 meters<br><br>Multimode Fiber (62.5/125)<br>Maximum cable length: 1.2 miles/2 km |

## Operating Environment

| | |
|---|---|
| Temperature | 32—131° F/0—55° C |
| Relative humidity | 10—95% noncondensing |
| Altitude | 0—15,000 feet/ 0—4.6 kilometers |

# Storage Environment

| | |
|---|---|
| Temperature | –40—185° F/–40—85° C |
| Relative humidity | 10—95% noncondensing |
| Altitude | 0—50,000 feet/0—15.2 kilometers |

# Regulatory Statements D

## FCC

**4576 ATM Communications Interface**

### FCC Part 15 Regulatory Compliance

Consult the dealer or an experienced radio/TV technician for help. This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Quality Manager
2105 Luna Road, Suite 320
Carrollton TX 75006
214-654-5000

This equipment complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This equipment may not cause harmful interference, and

(2) This equipment must accept any interference received, including interference that may cause undesired operation.

## Canadian

### Tested to Comply with Canadian Standards

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

# European

## Regulatory Information for Europe

This equipment displays the CE mark to show that it has been found to be in full compliance with the requirements of the EMC and Low Voltage Directives (89/336/EEC and 72/23/EEC, as amended by Directive 93/68/EEC).

# EN60950–IEC60950 Safety Standard

This equipment complies with the EN60950–IEC60950 safety standard.

# Interphase Fiber Products Compliance, LASER

**4576-006 and -007 ATM Communications Interface**

These Interphase fiber products contain a Class 1 laser device. They comply with IEC60825-1, IEC60825-2, and FDA 21 CFR 1040.10 and 1040.11. These products must be operated under recommended operating conditions.

Ce produit Interphase comporte une interface pour fibre optique utilisant un dispositif laser de classe 1. Il est conforme aux normes CEI 825-1 et 825-2, et FDA 21 CFR 1040,10 et 1040,11. Cet appareil doit être utilisé conformément aux conditions de fonctionnement recommandées.

## Usage Restrictions

The optical ports of these devices shall be terminated with an optical connector or with a dust plug.

## Restrictions d'utilisation

Les branchements optiques de cet appareil doivent toujours être raccordés à un connecteur optique ou porter un bouchon de protection.

# Interphase Fiber Products Compliance, LED

**4576-004 and -005 ATM Communications Interface**

These Interphase fiber products use an LED and complies with IEC60825-1, IEC60825-2, and FDA 21 CFR 1040.10 and 1040.11.

# Glossary

**AAL** ◆ **ATM Adaptation Layer** Service-dependent sublayer of the data link layer. The AAL accepts data from different applications and presents it to the ATM layer in the form of 48-byte ATM payload segments. AALs consist of two sublayers: CS and SAR. AALs differ on the basis of the source-destination timing used, whether they use CBR or VBR, and whether they are used for connection-oriented or connectionless mode data transfer. At present, the four types of AAL recommended by the ITU-T are AAL1, AAL2, AAL3/4, and AAL5.

**AAL1** ◆ **ATM Adaptation Layer 1**One of four AALs recommended by the ITU-T. AAL1 is used for connection-oriented, delay-sensitive services requiring constant bit rates, such as uncompressed video and other isochronous traffic.

**AAL2** ◆ **ATM Adaptation Layer 2** One of four AALs recommended by the ITU-T. AAL2 is used for connection-oriented services that support a variable bit rate, such as some isochronous video and voice traffic.

**AAL3/4** ◆ **ATM Adaptation Layer 3/4** One of four AALs (merged from two initially distinct adaptation layers) recommended by the ITU-T. AAL3/4 supports both connectionless and connection-oriented links, but is primarily used for the transmission of SMDS packets over ATM networks.

**AAL5** ◆ **ATM Adaptation Layer 5** One of four AALs recommended by the ITU-T. AAL5 supports connection-oriented VBR services and is used predominantly for the transfer of classical IP over ATM and LANE traffic. AAL5 uses SEAL and is the least complex of the current AAL recommendations. It offers low bandwidth overhead and simpler processing requirements in exchange for reduced bandwidth capacity and error-recovery capability.

**ABR** ◆ **Available Bit Rate** QoS class defined by the ATM Forum for ATM networks. ABR is used for connections that do not require timing relationships between source and destination. ABR provides no guarantees in terms of cell loss or delay, providing only best-effort service. Traffic sources adjust their transmission rate in response to information they receive describing the status of the network and its capability to successfully deliver data. Compare with CBR, UBR, and VBR.

**AIN** ◆ **Advanced Intelligent Network** In SS7, an expanded set of network services made available to the user, and under user control, that requires improvement in network switch architecture, signaling capabilities, and peripherals.

**AMI** ◆ **Alternate Mark Inversion** Line-code type used on T1 and E1 circuits. In AMI, zeros are represented by 01 during each bit cell, and ones are represented by 11 or 00, alternately, during each bit cell. AMI requires that the sending device maintain ones density. Ones density is not maintained independently of the data stream. Sometimes called binary coded alternate mark inversion.

**API** ◆ **Application Programming Interface** (1) The interface to a library of language-specific subroutines (such as a graphics library) that implement higher-level functions. (2) A set of calling conventions defining how a service is invoked through a software package.

**Apple Talk** ◆ Series of communications protocols designed by Apple Computer consisting of two phases. Phase 1, the earlier version, supports a single physical network that can have only one network number and be in one zone. Phase 2, supports multiple logical networks on a single physical network and allows networks to be in more than one zone.

**ASCII** ◆ **American Standard Code for Information Interchange** The standard binary encoding of alphabetical characters, numbers, and other keyboard symbols.

**ATM** ◆ **Asynchronous Transfer Mode** International standard for cell relay in which multiple service types (such as voice, video, or data) are conveyed in fixed-length (53-byte) cells. Fixed-length cells allow cell processing to occur in hardware, thereby reducing transit delays. ATM is designed to take advantage of high-speed transmission media such as E3, SONET, and T3.

**B8ZS** ◆ **Binary 8-Zero Substitution** Line-code type, used on T1 and E1 circuits, in which a special code is substituted whenever 8 consecutive zeros are sent over the link. This code is then interpreted at the remote end of the connection. This technique guarantees ones density independent of the data stream. Sometimes called bipolar 8-zero substitution.

**B Channel** ◆ **Bearer Channel** In ISDN, a full-duplex, 64-kbps channel used to send user data.

**BIOS** ◆ **Basic Input/Output System** The built-in program that controls the basic functions of communications between the processor and the Input/Output (I/O) devices of a computer.

**BISDN** ◆ **Broadband ISDN** ITU-T communication standards designed to handle high-bandwidth applications such as video. BISDN currently uses ATM technology over SONET-based transmission circuits to provide data rates from 155 to 622 Mbps and beyond.

**bootROM** ◆ **boot Read-Only Memory** Chip mounted on the printed circuit board used to provide executable boot instructions to a computer device.

**BRI** ◆ **Basic Rate Interface** ISDN interface composed of two B Channels and one D Channel for circuit-switched communication of voice, video, and data.

**BSP** ◆ **Board Support Package** A board support package consists of documentation and software used to configure and install a specific operating system on a specific product.

**BUS** ◆ **Broadcast and Unknown Server** Multicast server used in ELANs that is used to flood traffic addressed to an unknown destination and to forward multicast and broadcast traffic to the appropriate clients.

**CAM** ◆ **Content Addressable Memory** Memory that is accessed based on its contents, not on its memory address.

**CBR** ◆ **Constant Bit Rate** QoS class defined by the ATM Forum for ATM networks. CBR is used for connections that depend on precise clocking to ensure undistorted delivery.

**CCS** ◆ **Common Channel Signaling** Signaling system used in telephone networks that separates signaling information from user data. A specified channel is exclusively designated to carry signaling information for all other channels in the system.

**COM** ◆ **Communication or Communications**

**CompactPCI** ◆ CompactPCI is an adaptation of the Peripheral Component Interconnect (PCI) Specification for industrial and/or embedded applications requiring a more robust mechanical form factor than desktop PCI. CompactPCI uses industry standard mechanical components and high performance connector technologies to provide an optimized system intended for rugged applications. CompactPCI provides a system that is electrically compatible with the PCI Specification, allowing low cost PCI components to be utilized in a mechanical form factor suited for rugged environments. CompactPCI is an open specification supported by the PICMG (PCI Industrial Computer Manufacturers Group), which is a consortium of companies involved in utilizing PCI for embedded applications.

**CPCS** ◆ **Common Part Convergence Sublayer** An abstract ATM protocol API defined by the ATM Forum. It forms the boundary interface between the purely software implemented higher layer ATM protocols and the segmentation and reassembly process controlled by hardware.

**CPM** ◆ **Communication Processing Module**

**CRC4** ◆ **Cyclic Redundancy Check**. Error-checking technique in which the frame recipient calculates a remainder by dividing frame contents by a prime binary divisor and compares the calculated remainder to a value stored in the frame by the sending node.

**CS** ◆ **Convergence Sublayer** One of the two sublayers of the AAL CPCS, which is responsible for padding and error checking. PDUs passed from the SSCS are appended with an 8-byte trailer (for error checking and other control information) and padded, if necessary, so that the length of the resulting PDU is divisible by 48. These PDUs are then passed to the SAR sublayer of the CPCS for further processing.

**CSU** ◆ **Channel Service Unit** A component that terminates a digital circuit, such as T1. A CSU assures compliance to FCC regulations and preforms some line-conditioning functions.

**D Channel** ◆ **Data Channel** Full-duplex, 16-kbps (BRI) or 64-kbps (PRI) ISDN channel.

**DCE** ◆ 1. **Data Communications Equipment** (EIA expansion). 2. **Data Circuit-terminating Equipment** (ITU-T expansion). Devices and connections of a communications network that comprise the network end of the user-to-network interface. The DCE provides a physical connection to the network, forwards traffic, and provides a clocking signal used to synchronize data transmission between DCE and DTE devices. Modems and interface cards are examples of DCE.

**DLCI** ◆ **Data-Link Connection Identifier** Value that specifies a PVC or SVC in a Frame Relay network. In the basic Frame Relay specification, DLCIs are locally significant (connected devices might use different values to specify the same connection). In the LMI extended specification, DLCIs are globally significant (DLCIs specify individual end devices).

**DMA** ◆ **Direct Memory Access** The transfer of data directly into memory without supervision of the processor. The data is passed on the bus directly between the memory and another device.

**DPRAM** ◆ **Dual Port Random Access Memory**

**DS1** ◆ **Digital Signal level 1** Framing specification used in transmitting digital signals at 1.544-Mbps on a T1 facility (in the United States) or at 2.108-Mbps on an E1 facility (in Europe).

**DS3** ◆ **Digital Signal level 3** Framing specification used for transmitting digital signals at 44.736 Mbps on a T3 facility.

**DSX1** ◆ Cross-connection point for DS1 signals.

**DTE** ◆ **Data Terminal Equipment** Device at the user end of a user-network interface that serves as a data source, destination, or both. DTE connects to a data network through a DCE device (for example, a modem) and typically uses clocking signals generated by the DCE. DTE includes such devices as computers, protocol translators, and multiplexers.

**E1** ◆ Wide-area digital transmission scheme used predominantly in Europe that carries data at a rate of 2.048 Mbps. E1 lines can be leased for private use from common carriers.

**E3** ◆ Wide-area digital transmission scheme used predominantly in Europe that carries data at a rate of 34.368 Mbps. E3 lines can be leased for private use from common carriers.

**EEPROM** ◆ **Electrically Erasable Programmable Read-Only Memory** A nonvolatile PROM that can be written as well as read form. Usually used to hold information about the current system configuration, alternate boot paths, etc.

**ELAN** ◆ **Emulated LAN** ATM network in which an Ethernet or Token Ring LAN is emulated using a client-server model. ELANs are composed of an LEC, an LES, a BUS, and an LECS. Multiple ELANs can exist simultaneously on a single ATM network. ELANs are defined by the LANE specification.

**END** ◆ **Enhanced Network Driver**

**EPLD** ◆ **Electrically Programmable Logic Device**

**ES** ◆ **End System** Generally, an end-user device on a network.

**ESF** ◆ **Extended Superframe Format** Framing type used on T1 circuits that consists of 24 frames of 192 bits each, with the 193rd bit providing timing and other functions.

**Ethernet** ◆ Baseband LAN specification invented by Xerox Corporation and developed jointly by Xerox, Intel, and Digital Equipment Corporation.

**FCC** ◆ **Federal Communications Commission** The Government agency responsible for regulating telecommunications in the United States.

**FCC** ◆ **Fast serial Communication Controllers** Used to control the fast Ethernet port.

**FDDI** ◆ **Fiber Distributed Data Interface** LAN standard, defined by ANSI X3T9.5, specifying a 100-Mbps token-passing network using fiber-optic cable, with transmission distances of up to 2 km. FDDI uses a dual-ring architecture to provide redundancy.

**Flash** ◆ Nonvolatile storage that can be electrically erased and reprogrammed so that software images can be stored, booted, and rewritten as necessary.

**Frame Relay** ◆ Industry-standard, switched data link layer protocol that handles multiple virtual circuits using HDLC encapsulation between connected devices. Frame Relay is more efficient than X.25, the protocol for which it is generally considered a replacement.

**FTP** ◆ **File Transfer Protocol** Application protocol, part of the TCP/IP protocol stack, used for transferring files between network nodes.

**GB** ◆ **GigaBytes** $10^9$ bytes per second.

**Gbps** ◆ **Gigabits per second** $10^9$ bits per second.

**HDLC** ◆ **High-Level Data Link Control** Bit-oriented synchronous data link layer protocol developed by ISO. Derived from SDLC, HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums.

**IMA** ◆ **Inverse Multiplexing over ATM** Standard protocol defined by the ATM Forum in 1997.

**IMMR** ◆ **Internal Memory Map Register**

**IP** ◆ **Internet Protocol** Network layer protocol in the TCP/IP stack offering a connectionless internetwork service. IP provides features for addressing, type-of-service specification, fragmentation and reassembly, and security.

**IPv6** ◆ **IP version 6** Replacement for the current version of IP (version 4). IPv6 includes support for flow ID in the packet header, which can be used to identify flows. Formerly called IPng (next generation).

**IPX** ◆ **Internetwork Packet Exchange** NetWare network layer (Layer 3) protocol used for transferring data from servers to workstations. IPX is similar to IP and XNS.

**ISDN** ◆ **Integrated Services Digital Network** Communication protocol, offered by telephone companies, that permits telephone networks to carry data, voice, and other source traffic.

**ISO** ◆ **International Organization for Standardization** International organization that is responsible for a wide range of standards, including those relevant to networking. ISO developed the OSI reference model, a popular networking reference model.

**ITU-T** ◆ **International Telecommunication Union Telecommunication Standardization Sector** International body that develops worldwide standards for telecommunications technologies. The ITU-T carries out the functions of the former CCITT.

**J1** ◆ Japanese transmission standard

**LAN** ◆ **Local-Area Network** High-speed, low-error data network covering a relatively small geographic area (up to a few thousand meters). LANs connect workstations, peripherals, terminals, and other devices in a single building or other geographically limited area. LAN standards specify cabling and signaling at the physical and data link layers of the OSI model. Ethernet, FDDI, and Token Ring are widely used LAN technologies.

**LANE** ◆ **LAN Emulation** Technology that allows an ATM network to function as a LAN backbone. The ATM network must provide multicast and broadcast support, address mapping (MAC Address-to-ATM), SVC management, and a usable packet format. LANE also defines Ethernet and Token Ring ELANs.

**LAPB** ◆ **Link Access Procedure, Balanced**. Data link layer protocol in the X.25 protocol stack. LAPB is a bit-oriented protocol derived from HDLC.

**LEC** ◆ **LAN Emulation Client** Entity in an end system that performs data forwarding, address resolution, and other control functions for a single ES within a single ELAN. An LEC also provides a standard LAN service interface to any higher-layer entity that interfaces to the LEC. Each LEC is identified by a unique ATM address, and is associated with one or more MAC Addresses reachable through that ATM address.

**LECS** ◆ **LAN Emulation Configuration Server** Entity that assigns individual LANE clients to particular ELANs by directing them to the LES that corresponds to the ELAN. There is logically one LECS per administrative domain, and this serves all ELANs within that domain.

**LED** ◆ **Light Emitting Diode** A semiconductor device used to provide visual indications, used in place of an incandescent light. Also a semiconductor device used to transmit light into a fiber.

**LES** ◆ **LAN Emulation Server** Entity that implements the control function for a particular ELAN. There is only one logical LES per ELAN, and it is identified by a unique ATM address.

**LMI** ◆ **Local Management Interface** Set of enhancements to the basic Frame Relay specification. LMI includes support for a keepalive mechanism, which verifies that data is flowing; a multicast mechanism, which provides the network server with its local DLCI and the multicast DLCI; global addressing, which gives DLCIs global rather than local significance in Frame Relay networks; and a status mechanism, which provides an on-going status report on the DLCIs known to the switch. Known as LMT in ANSI terminology.

**MAC Address** ◆ Standardized data link layer address that is required for every port or device that connects to a LAN. Other devices in the network use these addresses to locate specific ports in the network and to create and update routing tables and data structures. MAC addresses are 6 bytes long and are controlled by the IEEE. Also known as a hardware address, MAC-layer address, and physical address.

**MCC** ◆ **Multichannel Communication Controller**

**MiniDIN** ◆ Miniature multi-pin connector.

**MPOA** ◆ **Multiprotocol over ATM** ATM Forum standardization effort specifying how existing and future network-layer protocols such as IP, IPv6, Apple Talk, and IPX run over an ATM network with directly attached hosts, routers, and multilayer LAN switches.

**MUX** ◆ **Multiplexer** Combines multiple signals for transmission over a single line. The signals are demultiplexed, or separated, at the receiving end

**NT1** ◆ **Network Termination 1** A device that provides the interface between customer premises equipment and central office switching equipment.

**NVRAM** ◆ **Nonvolatile RAM** RAM that retains its contents when a unit is powered off.

**OC3** ◆ **Optical Carrier 3** Physical protocol defined for SONET optical signal transmissions. OC3 signal levels put STS frames onto multimode fiber-optic line at 155.52 Mbps.

**OSI** ◆ **Open System Interconnection** International standardization program created by ISO and ITU-T to develop standards for data networking that facilitate multivendor equipment interoperability.

**PARC** ◆ **Palo Alto Research Center** Research and development center operated by XEROX. A number of widely-used technologies were originally conceived at PARC, including the first personal computers and LANs.

**PCI** ◆ **Peripheral Component Interconnect** A high-performance multiplexed address and data bus. Supporting 32-bit with optional 64-bit data transfers, the PCI bus is intended to be an interconnect between peripheral controllers, peripheral add-in boards, and processor/memory systems. The PCI bus operates at up to 33 MHz, providing burst transfer rates up to 132 MBps 32 bits wide, or up to 264 MBps 64 bits wide.

**PDN** ◆ **Public Data Network** Network operated either by a government (as in Europe) or by a private concern to provide computer communications to the public, usually for a fee. PDNs enable small organizations to create a WAN without all the equipment costs of long-distance circuits.

**PDU** ◆ **Protocol Data Unit** A message of a given protocol comprising payload and protocol-specific control information, typically contained in a header.

**PLP** ◆ **Packet Level Protocol** Network layer protocol in the X.25 protocol stack. Sometimes called X.25 Level 3 and X.25 Protocol.

**PMC** ◆ **PCI Mezzanine Card** PCI "daughter" card designed to mount on a "mother card".

**POST** ◆ **Power-On-Self-Test** Test that automatically runs whenever the power is applied to the card.

**PRI** ◆ **Primary Rate Interface** ISDN interface to primary rate access. Primary rate access consists of a single 64-Kbps D Channel plus 23 (T1) or 30 (E1) B Channels for voice or data.

**PROM** ◆ **Programmable Read-Only Memory** ROM that can be programmed using special equipment. PROMs can be programmed only once.

**PVC** ◆ **Permanent Virtual Circuit or Connection** Virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time. In ATM terminology, called a permanent virtual connection.

**QoS** ◆ **Quality of Service** Measure of performance for a transmission system that reflects its transmission quality and service availability.

**RAM** ◆ **Random-Access Memory** Volatile memory that can be read and written by a microprocessor.

**RISC** ◆ **Reduced Instruction Set Computing**

**ROM** ◆ **Read-Only Memory** Nonvolatile memory that can be read, but not written, by the microprocessor.

**RTM** ◆ **Rear Transition Module** A module that provides network connections from the rear of a system.

**Rx** ◆ **Receive or Receiver**

**SAR** ◆ **Segmentation And Reassembly** One of the two sublayers of the AAL CPCS, responsible for dividing (at the source) and reassembling (at the destination) the PDUs passed from the CS. The SAR sublayer takes the PDUs processed by the CS and, after dividing them into 48-byte pieces of payload data, passes them to the ATM layer for further processing.

**SCC** ◆ **Serial Communication Controller**

**SDH** ◆ **Synchronous Digital Hierarchy** European standard that defines a set of rate and format standards that are transmitted using optical signals over fiber. SDH is similar to SONET, with a basic SDH rate of 155.52 Mbps, designated at STM-1.

**SDLC** ◆ **Synchronous Data Link Control** SNA data link layer communications protocol. SDLC is a bit-oriented, full-duplex serial protocol that has spawned numerous similar protocols, including HDLC and LAPB.

**SDU** ◆ **Service Data Unit** A unit of interface information whose identity is preserved from one end of a layer connection to the other.

**SDRAM** ◆ **Synchronous Digital Random Access Memory**

**SEAL** ◆ **Simple And Efficient AAL** Scheme used by AAL5 in which the SAR sublayer segments CS PDUs without adding additional fields.

**SIU** ◆ **Serial Interface Unit**

**SMC** ◆ **Serial Management Controller**

**SMDS** ◆ **Switched Multimegabit Data Service** High-speed, packet-switched, datagram-based WAN networking technology offered by the telephone companies.

**SNA** ◆ **Systems Network Architecture** Large, complex, feature-rich network architecture developed in the 1970s by IBM.

**SONET** ◆ **Synchronous Optical Network** High-speed (up to 2.5 Gbps) synchronous network specification developed by Bellcore and designed to run on optical fiber. STS1 is the basic building block of SONET. Approved as an international standard in 1988.

**SS7** ◆ **Signaling System 7** Standard CCS system used with BISDN and ISDN.

**SSCS** ◆ **Service Specific Convergence Sublayer** One of the two sublayers of any AAL. SSCS, which is service dependent, offers assured data transmission. The SSCS can be null as well, in classical IP over ATM or LAN emulation implementations.

**STM-1** ◆ **Synchronous Transport Module level 1** One of a number of SDH formats that specifies the frame structure for the 155.52-Mbps lines used to carry ATM cells.

**STS** ◆ **Synchronous Transport Signal**

**STS1** ◆ **Synchronous Transport Signal level 1** Basic building block signal of SONET, operating at 51.84 Mbps. Faster SONET rates are defined as STS-n, where n is a multiple of 51.84 Mbps.

**SVC** ◆ **Switched Virtual Circuit** Virtual circuit that is dynamically established on demand and is torn down when transmission is complete. SVCs are used in situations where data transmission is sporadic. Called a switched virtual connection in ATM terminology.

**T1** ◆ T1 transmits DS1-formatted data at 1.544 Mbps through the telephone-switching network, using AMI or B8ZS coding.

**T3** ◆ Digital WAN carrier facility. T3 transmits DS3-formatted data at 44.736 Mbps through the telephone switching network.

**TCP** ◆ **Transmission Control Protocol** Connection-oriented transport layer protocol that provides reliable full-duplex data transmission. TCP is part of the TCP/IP protocol stack.

**TCP/IP** ◆ **Transmission Control Protocol/Internet Protocol** Common name for the suite of protocols developed by the U.S. DoD in the 1970s to support the construction of worldwide internetworks. TCP and IP are the two best-known protocols in the suite.

**TFTP** ◆ **Trivial File Transfer Protocol** Simplified version of FTP that allows files to be transferred from one computer to another over a network.

**Token Ring** ◆ Token-passing LAN developed and supported by IBM. Token Ring runs at 4 or 16 Mbps over a ring topology. Similar to IEEE 802.5.

**TTY** ◆ **Teletypewriter** General term for an input device.

**Tx** ◆ **Transmit or Transmitter**

**UBR** ◆ **Unspecified Bit Rate** QoS class defined by the ATM Forum for ATM networks. UBR allows any amount of data up to a specified maximum to be sent across the network, but there are no guarantees in terms of cell loss rate and delay. Compare with ABR, CBR, and VBR.

**USRBUF** ◆ A driver structure describing the use of a specific buffer containing payload data to be transferred using ATM. They can be linked together to allow non-contiguous areas of memory to be sent as one unit.

**X.25** ◆ ITU-T standard that defines how connections between DTE and DCE are maintained for remote terminal access and computer communications in PDNs. X.25 specifies LAPB, a data link layer protocol, and PLP, a network layer protocol. Frame Relay has to some degree superseded X.25.

**XNS** ◆ **Xerox Network Systems** Protocol suite originally designed by PARC. Many PC networking companies, such as 3Com, Banyan, Novell, and UB Networks used or currently use a variation of XNS as their primary transport protocol.

**VBR** ◆ **Variable Bit Rate** QoS class defined by the ATM Forum for ATM networks. VBR is subdivided into a Real Time (RT) class and Non-Real Time (NRT) class. VBR (RT) is used for connections in which there is a fixed timing relationship between samples. VBR (NRT) is used for connections in which there is no fixed timing relationship between samples, but that still need a guaranteed QoS.

**VCC** ◆ **Virtual Channel Connection** Can be a Permanent Virtual Connection (PVC) or a Switched Virtual Connection (SVC). Any ATM connection between two nodes.

**VCI** ◆ **Virtual Channel Identifier**16-bit field in the header of an ATM cell. The VCI, together with the VPI, is used to identify the next destination of a cell as it passes through a series of ATM switches on its way to its destination. ATM switches use the VPI/VCI fields to identify the next network VCL that a cell needs to transit on its way to its final destination. The function of the VCI is similar to that of the DLCI in Frame Relay.

**VCL** ◆ **Virtual Channel Link** Connection between two ATM devices. A VCC is made up of one or more VCLs.

**VPI** ◆ **Virtual Path Identifier** 8-bit field in the header of an ATM cell. The VPI, together with the VCI, is used to identify the next destination of a cell as it passes through a series of ATM switches on its way to its destination. ATM switches use the VPI/VCI fields to identify the next VCL that a cell needs to transit on its way to its final destination. The function of the VPI is similar to that of the DLCI in Frame Relay.

**WAN** ◆ **Wide-Area Network** Data communications network that serves users across a broad geographic area and often uses transmission devices provided by common carriers. Frame Relay, SMDS, and X.25 are examples of WANs.

# Index

When using this index, keep in mind that a page number indicates only where referenced material begins. It may extend to the page or pages following the page referenced.


**A**

adapter
- connecting to network ..... 11
- driver installation
  - Linux ..... 23
  - Linux Intel API ..... 29
  - Linux PowerPC API ..... 27
  - Solaris ..... 13, 35
  - VxWorks ..... 19
- inspecting ..... 9
- installing ..... 9
- troubleshooting ..... 109
- unpacking ..... 9

adapter features ..... 2

**B**

boot problems ..... 112

**C**

caution
- electrostatic damage ..... 9

configuring network protocols ..... 16
connecting to the network ..... 11
connector
- SC Duplex ..... 11

connectors and cables
- installation ..... 12

**D**

data rates ..... 1
driver
- installation
  - Linux ..... 23
  - Solaris ..... 13, 35
  - VxWorks ..... 19
- removing
  - Solaris ..... 17
- unloading
  - VxWorks ..... 21

**E**

Environment
- Operating ..... 153
- Storage ..... 154

errors, troubleshooting ..... 109

**F**

features ..... 2
fiber connections ..... 12

**G**

grounding strap ..... 9, 10

**H**

hardware
- installation ..... 9

host adapter not found ..... 113

**I**

inspecting the adapter ..... 9
installing the adapter ..... 9
installing the driver
- Linux ..... 23
  - prerequisites ..... 23
- Solaris ..... 13, 35
  - completing installation ..... 17
  - network protocols ..... 16
  - prerequisites ..... 13, 38
  - starting ..... 15
- VxWorks ..... 19
  - prerequisites ..... 19

isar_alarm_ind ..... 66
isar_oam_ind ..... 64
isar_query ..... 59
isar_rcv_ind ..... 62
isartest
- autotest ..... 132
- close/open connections ..... 134
- control ..... 121
- display ..... 123, 143
- exit ..... 137, 152
- help ..... 125, 146
- parameters ..... 128, 150
- query ..... 130, 150
- transmit ..... 134

**L**

LED
- description ..... 109
- faceplate ..... 12
- interpreting ..... 109
- state and function ..... 109

Link LED ..... 12, 109
link manager ..... 6
Linux ..... 32
- command-line installation ..... 32
- driver installation ..... 23

**M**

motherboard removal ..... 10

## N

network connection ..... 11
network problems ..... 114
network protocols for Solaris ..... 16

## P

pkgadd command ..... 15
pkginfo command ..... 15
problems and solutions ..... 110
product features ..... 2

## Q

quick test ..... 53

## R

reboot command ..... 17
removing driver
  Solaris ..... 17

## S

SC Duplex connections ..... 11, 12
shutdown command ..... 17
Solaris
  driver installation ..... 13, 35
  driver removal ..... 17
  rebooting the machine ..... 17
Status LED ..... 12, 109

## T

tools required ..... 9, 10
troubleshooting ..... 109
  boot problems ..... 112
  computer does not boot ..... 113
  link problems ..... 110
  network problems ..... 114
  no host adapter found ..... 113
  status problems ..... 110, 111
TX/RX
  fiber connections ..... 12

## U

unload driver
  VxWorks ..... 21
unpacking adapter ..... 9

## V

VBR support ..... 50
VxWorks
  driver installation ..... 19
  driver unload ..... 21

## W

warnings
  damaged board ..... 9
  electrical shock ..... 10
  *See also* caution