

Netaccess Series Release 7.5 Linux Device Driver Programmer's Manual

Driver Release 7.5

Document Number 937-110-60

Printed July 2002



General Notices

Copyright© 2001 - 2002, Brooktrout Technology, a Brooktrout Company.

All rights reserved.

This product may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from Brooktrout Technology.

Brooktrout Technology reserves the right to make improvements and/or changes in the products and programs described in this Type of Manual at any time without notice. Every attempt has been made to insure that the information contained in this document is accurate and complete. Brooktrout Technology will not be responsible for any inaccuracies or omissions in this or any of its other technical publications.

Printed in the United States of America.

Trademarks

Brooktrout Inc., Brooktrout Technology, Netaccess, Instant RAS, Instant ISDN, and TruFax are registered trademarks of Brooktrout, Inc.

TR Series, TR114, TRNIC, Universal Port, TR2000 Series, TR2001, TR1000, RDSP Series, RTNI Series, Ensemble Series, Vantage DSPM, Vantage PCI Series, Vantage Series, AnyCall, AccuCall, AccuSwitch, AccuSpan, AccuLock, AccuTalk, AccuDigit, AccuRate, AccuPitch, AccuTone, AccuPulse, RDSPTest, RFX, RTNI, Prelude, RealCT, CTMedic, Prompt Studio, VEdit, BTGateway, SpeechPac, and BTStack323 are trademarks of Brooktrout, Inc.

Windows, Windows NT, Windows 95, Windows 98, Microsoft SQL Server, Excel, FoxPro, FoxBase, and Visual C++ are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark licensed exclusively through X/Open Company, Ltd.

MVIP is a trademark of Go-MVIP, Inc.

Pentium and Intel are registered trademarks of Intel Corporation.

Adobe and Acrobat are registered trademarks of Adobe Systems Incorporated.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective companies.

International Notice

Due to differing national regulations and approval requirements, certain Brooktrout products are designed for use only in specific countries, and may not function properly in a country other than the country of designated use. As a user of these products, you are responsible for ensuring that the products are used only in the countries for which they were intended. For information on specific products, contact Brooktrout Technology.

18 Keewaydin Drive
Salem, NH 03079
603-898-1800
www.brooktrout.com

Brooktrout Technical Support

For Brooktrout Technical Support, see *Appendix A*.

Brooktrout Technology, Inc.

Software License Agreement

Proprietary Rights

The Software is subject to the protection of the copyright laws of the U.S. and foreign jurisdictions, which prohibit unauthorized copying and distribution of copyrighted works. The Software incorporates proprietary and confidential algorithms and techniques which are subject to legal protection as trade secrets. Brooktrout is the sole owner of all proprietary rights in the Software, except for certain portions which are proprietary to third party licensors of Brooktrout. The User is granted only those rights expressly conferred by this License Agreement.

License

Brooktrout licenses the User to use the Software subject to and in accordance with the following provisions. The software is distributed with network interface boards or boards with network interface functionality that are manufactured and sold by Brooktrout ("Brooktrout Hardware"), and is licensed solely for use in connection with the Brooktrout Hardware. The Software, and modified versions thereof, may be operated only on the central processing unit of any computer served by one or more items of Brooktrout Hardware and may, where appropriate in connection with such use, be downloaded onto memory located on Brooktrout Hardware, and may be modified (if modification is otherwise permitted pursuant to the following provisions), reproduced and distributed only for purposes of such use. Any other use, modification, reproduction or distribution is expressly prohibited.

Licensing provisions applicable to particular Software products are as follows:

1. API, Application and Driver software and Downloadable Firmware distributed in the form of object code:
 - a. The User may incorporate the Software into his own work providing functional and value enhancements and may duplicate and distribute the resulting work as he chooses provided that the resulting work is designed solely for use in connection with Brooktrout Hardware and may be distributed only together with items of Brooktrout Hardware solely for use in connection with such items.
 - b. The User may not modify the Software nor decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human perceivable form.
 - c. When the User incorporates the Software into his product, Brooktrout's copyright notice must be included in the new work.
2. API, Application or Driver software distributed in the form of source code:
 - a. The User may modify the Software and must incorporate it into his own work providing functional and value enhancements. He may duplicate and distribute the resulting work in object code form only provided that the resulting work is designed solely for use in connection with Brooktrout Hardware and may be distributed only together with items of Brooktrout Hardware solely for use in connection with such items. He may not distribute the Software in source form.
 - b. The Software is confidential and proprietary to Brooktrout and the User must protect it in a manner similar to the protection he affords his own confidential and proprietary information.
 - c. When the User incorporates the Software into his product, Brooktrout's copyright notice must be included in the new work.

The reproduction, distribution and modification rights provided above applies to all Software distributed by Brooktrout unless a specific license agreement stating otherwise is attached or part of a contract under which such Software is being provided. In those cases, the specific license agreement will apply.

Distribution

Any distribution of the Software (including modified versions thereof) which is authorized hereby shall be made (a) in object form only; (b) only to purchasers of units of Brooktrout Hardware, or of products including Brooktrout Hardware, and (c) only pursuant to license agreements containing provisions substantially equivalent to those included herein with respect to the Software distribution. Except as expressly permitted hereby, the user may not distribute the Software, or any copy by transfer, lease, loan or any other means.

Termination

The User's license to use the Software may be terminated by Brooktrout in the event of any failure to comply with the above restrictions or any other terms of this License Agreement. In the event of termination of the license, the User must destroy or return to Brooktrout all copies of the Software in his possession.

Limited Warranty

Brooktrout warrants for a period of 90 days following delivery that the disk on which the Software is recorded and which is provided by Brooktrout is free from defects in materials and workmanship. Brooktrout does not warrant that operation of the Software will be uninterrupted or error-free, or that it will satisfy the User's requirements. BROOKTROUT DISCLAIMS ALL OTHER WARRANTIES EXPRESS OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Limitation of Liability

Brooktrout's entire liability and the User's exclusive remedy in connection with the Software will be the replacement of any disk not meeting the above limited warranty upon return of the disk to Brooktrout. In no event will Brooktrout be liable for damages, including any lost profits or other incidental or consequential damages, arising out of or related to the Software and its use, even if Brooktrout has been advised of the possibility of such damages.



Table of Contents

Chapter 1 – Overview

Introduction	1-1
Netaccess Series 7.5 Driver Overview	1-1
Netaccess Series 7.5 Driver Files	1-3

Chapter 2 – Installation, Initialization & Operation

Introduction	2-1
Netaccess Series 7.5 Driver Installation	2-1
Installing the Driver	2-1
Removing a Previous Version	2-3
Driver module loading/unloading	2-3
Running Demo Programs	2-4
Normal Netaccess Series 7.5 Driver Initialization & Operation	2-7
Allocating Data Channels	2-7
Sending Messages/Data Packets Down a Stream	2-8
Retrieving Messages/Data Packets	2-8
Using the Management Channel	2-9
Tunable Parameters	2-10
Modifying Tunable Parameters	2-10

Chapter 3 – Driver Services

Introduction	3-1
ioctl - WANDRViocTEST	3-3
ioctl - WANDRViocBOOT Download Code to the Netaccess Series 7.5 Controller	3-4
ioctl - WANDRViocENA_MGT_CHAN Enable a Management Channel	3-5
ioctl - WANDRViocRESET_BOARD Reset a Netaccess Series 7.5 Controller	3-6
ioctl - WANDRViocGET_BOARD_TYPE Return the type of Netaccess Series 7.5 Controller ...	3-7
ioctl - Retrieve Board Data Structures	3-8
WANDRViocGET_MSG, WANDRViocPUT_MSG	3-10

Chapter 4 – Hot Swap

Introduction	4-1
Overview Note	4-1
Implementation	4-1
Configuration	4-2
Usage	4-2
Extraction (NS300/NS301)	4-2

Extraction (PCI-cPCI) with the	4-2
Insertion (NS300/NS301)	4-3
Insertion (PCI-cPCI)	4-3

Chapter 5 – High Availability

Introduction	5-1
Overview Note	5-1
Requirements	5-1
Configuration	5-1
New loctl's	5-1
Usage	5-2

Chapter 6 – Driver Utilities

Introduction	6-1
boot_board	6-2
pridump	6-3
s_2_bin	6-4

Appendix A – Brooktrout Customer Support

Customer First	A-1
Before You Call	A-1
Contacting Brooktrout Customer Support	A-1
Additional Brooktrout Support Services	A-2



Overview

Introduction

This *Programmer Manual* describes the functions provided by the Netaccess Series 7.5 Linux™ Driver; this Netaccess Series 7.5 Driver release supports Instant ISDN Software™ Release 7.x.

This document has been designed for system developers who are using Netaccess Series 7.5 T1/E1/PRI boards in their application. It assumes you have experience with the C programming language and with using intelligent controller boards in a PCI/CPCI bus environment. You should also have the following publications on hand when using the Netaccess Series 7.5 Driver:

- *Instant ISDN Software Release Notes*: describes the features of Release 7.x of Instant ISDN Software.
- *Instant ISDN Software SMI Reference*: defines the Simple Message Interface (SMI™) control messages used to implement advanced telecommunications and data networking services using Netaccess Series 7.5 boards.
- Appropriate *Technical Description* for the board used.
- Appropriate manuals from the Linux documentation set.

Netaccess Series 7.5 Driver Overview

The Netaccess Series 7.5 Driver allows the application programmer complete access to all functions of the Netaccess Series 7 family of PCI/CPCI-based ISDN controllers. The Netaccess Series 7.5 Linux Driver supports systems running Linux Kernel 2.0.0 or greater.

The Netaccess Series 7.5 Linux Driver may be configured at installation time to support up to eight Netaccess Series 7.5 T1/E1/PRI boards. For more information on board characteristics, refer to the *Technical Description* that accompanies the board. Refer to *Section 2* of this manual for more information on driver installation.

Figure 1-1 shows a single stream providing a connection between the application and a board via the Netaccess Series 7 Driver. The Netaccess Series 7.5 Driver performs the role of a standard CHARACTER device driver in the following ways:

- Handles interrupts from up to eight T1/E1/PRI Controllers
- Supports multiple connections between the application and the driver
- Responds to **open** and **close** calls to be initialized and deinitialized

The Netaccess Series 7.5 Driver responds to standard **open** and **close** services to establish and remove character connections. The **select**, and **ioctl** services are used to pass SMI control messages and data packets between the host-based application and the Netaccess Series 7.5 Controller. Several Netaccess Series 7.5-specific **ioctl** services are used to select an individual board and retrieve error and status information. The Netaccess Series 7.5 Driver services are defined in *Section 3*. Normal driver operations are outlined in *Section 2*.

The Netaccess Series 7.5 Driver implementation limits the application to one LAP-D virtual circuit per stream to ensure flow control works properly. Each HDLC channel on the board can support up to eight LAP-D virtual circuits. Virtual circuits are specified by a combination of the HDLC channel number (LAP-D ID) and a Logical Link ID (LLI). Each circuit must specify a unique set of LAP-D ID and LLI values; no two connections can be mapped to the same LAP-D ID/LLI combination.

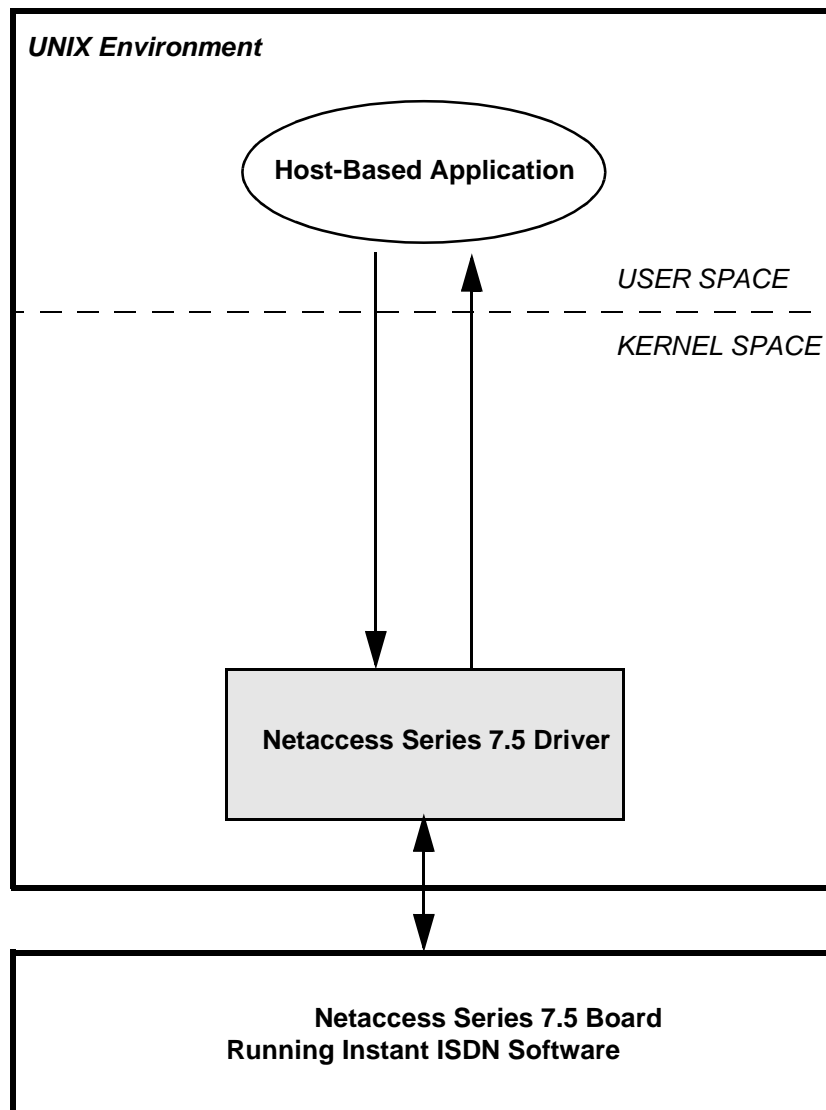


Figure 1-1. Single Character Between an Application and a Netaccess Series 7.5 Board

Three utilities are supplied with the Netaccess Series 7.5 Driver:

- `boot_board` downloads Instant ISDN Software to the boards

- pridump provides diagnostic information following a board failure
- s_2_bin converts Motorola S-record format files to binary versions

These utility programs are described in *Section 6*.

Netaccess Series 7.5 Driver Files

The Netaccess Series 7.5 Driver software is provided as a .tgz file. This file contains Instant ISDN Software Release 7.x as well as the Netaccess Series 7.5 Driver files.

The Netaccess Series 7.5 Driver installation process creates the directories and files on your system as listed in *Table 1-1*. Instructions for installing and initializing the driver are located in *Section 2*.

Table 1-1. Netaccess Series 7.5 Driver Files

Directory	File
/usr/local/wandrv	Change_History Readme.wandrv wandrv.fillst update.fillst
/usr/local/wandrv/demos	boot_board.c check_raw.c id.c lapd_data.c line_test.c makefile mtp2.c naii.0 naii.1 pridump.c q931.c q933.c reset.c s_2_bin.c signaling.c test_boot
/usr/local/wandrv/driver	makefile sysdep.h wandrv.c wandrv_debug.h wandrvnfo.h
/usr/local/wandrv/images	naii_0.bin naii_1.bin
/usr/local/wandrv/include	iisdn.h wandrv.h
/etc/rc.d/init.d	wandrv
/usr/local/wandrv/ha_control	Makefile hsc_state_tr.c

Table 1-1. Netaccess Series 7.5 Driver Files (Continued)

Directory	File
/usr/local/wandrv/ha_demos	hs_enable.c hs_quiesce.c lapd_data_ha.c makefile
/usr/local/wandrv/hs_demos	hs_control.c makefile



Installation, Initialization & Operation

Introduction

This section describes how to install and use the Netaccess Series 7.5 Driver; the information is divided into the following areas:

- Installing Netaccess Series 7.5 Driver software on the host system
- Normal Netaccess Series 7.5 Driver initialization and operation
- Tunable driver parameters

Note: The information in this section assumes the Netaccess Series 7.5 boards have already been installed in the system. Refer to the appropriate Netaccess Series 7.5 Technical Description for board installation instructions.

Netaccess Series 7.5 Driver Installation

Before you begin the installation, do the following:

- Remove any previous versions of the Netaccess Series 7.5 Driver installed on your system; refer to *page 2-3* for instructions.
- Make sure that no other users are logged into the system through a remote terminal during the installation process.

Installing the Driver

To install the Netaccess Series 7.5 Driver files, perform the following steps:

1. If you have not already, log into your system using the root password.
2. Copy the `/<Driver>.tgz` file to the `/` directory.
3. Change directory to the `/` directory.
4. From the command shell, run the following command:

```
tar -xzvf /<Driver>.tgz Install.wandrv
```

5. Then type in to start the installation process.

```
./Install.wandrv /<Driver>.tgz
```

6. Type in **y** to the following question:

```
-----  
Linux 2.x Netaccess Series 7.5 Device Driver Installation Program  
-----
```

```
Install Linux 2.x Netaccess Series 7.5 Device Driver? (y/n) [y]
```

7. The next option allows for Motorola MCG High Availability/Hot Swap support. This should only be enabled on a Motorola CPX 2000 (cPCI) or CPX 8000 (cPCI) with a Linux – HA kernel installed. The Default is **No**.

```
Install with Motorola MCG High Availability/Hot Swap support? (y/n)  
[n]
```

8. This option allows for any cPCI chassis to have the Hot Swap enabled. The Default is **No**.

```
Install with Generic Hot Swap support? (y/n) [n]
```

9. This option allows the user to define whether the driver is going to use Board based buffers (on-board memory) or host based buffers (memory on the computer). The default is board based buffers.

```
Would you like the driver to use data buffers in host or board  
memory? (b/h) [b]
```

10. This feature consists of a buffer being filled with the packets in the receive queue until the maximum size of the buffer is reached or the receive queue is empty. For example, a buffer is allocated with a size of 256bytes. Four packets (size = 100 bytes) are sent to the board and are waiting in the receive queue. A read request comes down from the user application. If the RECV_PKT_OPT option is turned on, the buffer will return with 2 packets (100bytes pkt + 100bytes pkt = 200bytes in a 256 byte buffer). If this option is turned off, the buffer would return with one packet (100byte packet in a 256 byte buffer). This option is useful when reading data fast. One read request is able to get several packets at once. The user application determines whether or not to separate the packets at the host level.

```
Would you like the driver to do Receive Packet Optimization? (y/n)  
[n]
```

11. The script copies the files to the local machine. The option to build the test applications is the last option. The default is yes.

```
Copying Files From Installation Medium To Root Directory ...  
Setting permissions/ownership on files ...  
Determining if rc.d structure is Slackware or Redhat format ...  
Setting up rc.d files for Redhat format ...  
Would you like to build the Netaccess test programs? (y/n) [y]
```

12. Now the script makes the driver and installs it on the machine.

```
Making Device Driver Module ...
```

13. This installation script is finished. All of the parameters above may be changed by editing the file `/usr/local/wandrv/driver/wandrvnfo.h`.

The Linux 2.x Netaccess Series 7.5 Device Driver Has
Installed Successfully to `/usr/local/wandrv`.

Removing a Previous Version

To remove a previous release of the Netaccess Series 7.5 Driver from your system, perform the following steps:

1. *If you have not already*, log into your system as the owner or using the root password.
2. Change directory to the `/` directory.
3. At the command prompt, run the following command

```
./UnInstall.wandrv
```

Driver module loading/unloading

The module's init script is stored in the `/etc/rc.d/init.d` directory.

Loading

Run the following command to load the driver

For Redhat, Mandrake and other like distributions

```
/etc/rc.d/init.d/wandrv start
```

For Slackware

```
/etc/rc.d/rc.wandrv start
```

UnLoading

Run the following command to unload the driver:

For Redhat, Mandrake and other like distributions:

```
/etc/rc.d/init.d/wandrv stop
```

For Slackware

```
/etc/rc.d/rc.wandrv stop
```

Restarting

Run the following command to restart the driver:

For Redhat, Mandrake and other like distributions

```
/etc/rc.d/init.d/wandrv restart
```

For Slackware

```
/etc/rc.d/rc.wandrv restart
```

Running Demo Programs

Table 2.1 lists the test programs supplied with the Netaccess Series 7.5 Driver software. These programs serve as models for Netaccess Series 7.5 Driver operation, and can be used to test Netaccess Series 7.5 boards in various call scenarios. The sample programs should be run only after Instant ISDN Software has been downloaded to the boards using the boot_board utility.

Table 2-1. Driver Test Programs

Test Name(s)	Purpose/Summary	Requirements
boot_board.c	Downloads the IISDN firmware	<ul style="list-style-type: none"> ■ Compile using "make boot_board" or "make" ■ Run boot_board to download the IISDN firmware
check_raw.c	A RAW data integrity program (also demonstrates the use of forks).	<ul style="list-style-type: none"> ■ Compile using "make check_raw" or "make". ■ Run boot_board to download the IISDN firmware ■ Run check_raw ('check_raw -h' for command line options)
lapd_data.c	A LAP-D data integrity program	<ul style="list-style-type: none"> ■ Compile using "make lapd_data" or "make" ■ Run boot_board to download the IISDN firmware ■ Run lapd_data (lapd_data -h for command-line options)
line_test.c	A program for testing proper alarm conditions and for demonstrating the establishment of a D-Channel	<ul style="list-style-type: none"> ■ Compile using "make line_test" or "make" ■ Run boot_board to download the IISDN firmware ■ A dual (or more) span controller ■ A cross-over cable from Span A to Span B ■ Run line_test

Table 2-1. Driver Test Programs (Continued)

Test Name(s)	Purpose/Summary	Requirements
<p>pridump.c</p>	<p>For dumping out the Interrupt Status.</p> <p>Block on a Netaccess Series 7.5 controller (Especially useful for sending "panic" information to Netaccess Series 7.5 Customer Support). When this program is run successfully with the board not in a panic state the output will look this this:</p> <pre> Board number 0x0 Board number 1 0x0 Version: 0 NAI 0dH r07.00.13 05/06/02 WAN board 0 traceback for vector 0 D: [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] A: [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] [0000.0000] Stack: 0000 Interrupt Status Block: int_rsn = x0 error_code = x0 error_point = x0 error_arg = x0 timestamp = x1EBF spare1 = x0 spare2 = x0 vme_irq_level = x0 When the board is in a panic state useful information will be printed here that will help customer support engineers resolve the issue.</pre>	<ul style="list-style-type: none"> ■ Compile using "make pridump" or "make" ■ Run pridump -b# (where # is the board number).

Table 2-1. Driver Test Programs (Continued)

Test Name(s)	Purpose/Summary	Requirements
q931.c	A program demonstrating proper Q.931 Call Control using the Netaccess Series 7.5 SMI (runs in two windows).	<ul style="list-style-type: none"> ■ Compile using "make q931_net" and "make q931_user" or "make" ■ Run boot_board to download the IISDN firmware ■ Run q931_net in one window ■ Run q931_user in another window
reset.c	A program that puts a Netaccess Series 7.5 controller into reset.	<ul style="list-style-type: none"> ■ • Compile using "make reset" or "make" ■ • Run reset
s_2_bin.c	A program for converting IISDN images from S-record format to binary.	<ul style="list-style-type: none"> ■ Compile using "make s_2_bin" or "make" ■ Run "s_2_bin <S-Record> <binary> or "make naii_0.bin" (PCI) and "make naii_1.bin" (cPCI)
signaling.c	Demonstrates the use of the Netaccess Series 7.5 SIGNALING implementation	<ul style="list-style-type: none"> ■ Compile using "make signaling" or "make" ■ Run boot_board to download the IISDN firmware ■ A cross-over cable from Span A to Span B ■ Run signaling
id.c	Prints some board identification information	<ul style="list-style-type: none"> ■ Compile using "make id" or "make" ■ Run boot_board to download the IISDN firmware ■ Run id
mtp2.c	Tests the SS7 (mtp2 layer) capabilities of the Netaccess Series 7.5 controller.	<ul style="list-style-type: none"> ■ Compile using "make mtp2" or "make" ■ Run boot_board to download the IISDN firmware ■ Run mtp2
q933.c	Tests the Q.933 capabilities of the Netaccess Series 7.5 controller.	<ul style="list-style-type: none"> ■ Compile using "make q933" or "make" ■ Run boot_board to download the IISDN firmware ■ Run q933 b# (where # is the board number)
test_boot	Continually redownloads the IISDN firmware to the Netaccess Series 7.5 controller and keeps track of success and failure.	<ul style="list-style-type: none"> ■ Run test_boot

Normal Netaccess Series 7.5 Driver Initialization & Operation

To initialize the Netaccess Series 7.5 Driver, the application should open a file descriptor for each virtual circuit needed. The file descriptor should identify the Netaccess Series 7.5 Controller, physical HDLC channel, and logical channel used to define each stream. HDLC channels on a board are specified by LAP-D ID. Logical channels are specified by Logical Link ID (LLI); the LLI is equivalent to the ISDN LAP-D/Q.921 Data Link Connection Identifier (DLCI). Each file descriptor must specify a unique set of LAP-D ID and LLI values.

Note: Never attempt to map two file descriptors to the same logical channel. If two or more file descriptors specify the same LAP-D ID/LLI combination, control messages and data packets sent from the Netaccess Series 7.5 Controller to the host may be routed over the incorrect stream.

Using the file descriptor, the application must then issue an L4L3mENABLE_PROTOCOL control message down the stream to establish a link. When the link is established, the Netaccess Series 7.5 Controller returns an L3L4mPROTOCOL_STATUS message back indicating an established link. The Netaccess Series 7.5 Controller attempts to re-establish the link whenever it is down and reports the link's status only when the link state has changed. The Netaccess Series 7.5 Controller sends a channel state of ESTABLISHING every 6 seconds or so until the link comes up. Similarly, if the link goes down, the application receives an L3L4mPROTOCOL_STATUS with state set to ESTABLISHING every 6 seconds until the link comes back up.

Note: Refer to the *Instant ISDN Software SMI Reference* for complete descriptions of SMI control messages.

To communicate with a Netaccess Series 7.5 Controller, the application should use **ioctl** to send control messages and *write* for data packets to specific virtual circuits.

Allocating Data Channels

In a data passing application, the application opens a file descriptor for each data interface required. The data interface is created by sending an L4L3mENABLE_PROTOCOL message with the **enable** byte set to "01" in the **data_interface** structure.

Because data channels are controlled and allocated by the driver, any **data_channel** value passed from the application in an L4L3m message is ignored. Instead, the driver allocates the **data_channel** value based on the stream on which the message is received. This value is returned to the host in the L3L4m common header.

Note: This is not the case if running in a High Availability environment. In such an environment, the host application must specify the data channel and keep track of it in case of a CPU failover.

Sending Messages/Data Packets Down a Stream

To send L4L3 control messages to a Netaccess Series 7.5 Controller, the application uses the `WANDRViocPUT_MSG` command of the `ioctl` system. A sample code fragment for sending a control message is below.

```
L4_to_L3_struct cntl_msg;

cntl_msg.lapdid=0;
cntl_msg.msgtype = L4L3mSET_HARDWARE;
cntl_msg.call_ref = cntl_msg.lli = cntl_msg.L4_ref = 0;
if (ioctl (fd, WANDRViocPUT_MSG, &cntl_msg) < 0)
    printf ("Failed to issue SET_HARDWARE message\n");
```

To send a data packet to a Netaccess Series 7.5 Controller, the application uses the write entry point of the driver. The write (and read) entry points are blocking/interruptible system calls. A sample code fragment for sending a data packet is below.

```
for (i = 0; i < IFRAMESZ; i++) {
    buf[i] = i;
}
if (write(fd,&buf[0],IFRAMESZ) < 0)
    printf("write error (errno=%d)\n", errno);
```

Retrieving Messages/Data Packets

A combination of `select` and the `WANDRViocGET_MSG` command of `ioctl` service calls can be used to retrieve L3L4 control messages and data packets from Netaccess Series 7.5 Controllers. By using `select` to receive all events, there is exactly one place in the code which waits for events to occur and receives these events reliably. When using `WANDRViocGET_MSG` command of `ioctl`, the system can retrieve data and/or control messages. When a control message is present, the message is in the form defined by the `L3_to_L4_struct` in `iisdn.h`.

To retrieve data from a Netaccess Series 7.5 controller, the read entry point is called. The read (like the write) call is blocking/interruptible.

A sample code fragment for receiving events is provided on the following page.

```

fd_set rfds, efds;
struct timeval tv;
L3_to_L4_struct msg;
int fd, inlen;
unsigned char buf[IFRAMESZ];
extern int errno;

tv.tv_sec = 1;
tv.tv_usec = 0;

FD_ZERO (&efds);
FD_ZERO (&rfds);

for (;;) {

    FD_SET (fd, &efds);
    FD_SET (fd, &rfds);

    if (select (fd+1, &rfds, NULL, &efds, &tv) < 0)
        printf("Select failed\n");
    if (FD_ISSET(fd, &eset)) {
        result = ioctl (fd, WANDRViocGET_MSG, &inmsg);
        // Check message type here
    }
    if (FD_ISSET(fd, &rset)) {
        inlen = read (fd, buf, IFRAMESZ);
        // Check data here
    }
}
}

```

Using the Management Channel

Only one management channel can be opened between a Netaccess Series 7.5 driver and an application per board. This management channel is used to pass SMI control messages between the Netaccess Series 7.5 Controller and the application that are not specific to a certain channel on the board. For example, L3L4mLINE_STATUS messages that report alarms on the board's T1/E1 interfaces are sent on the management channel.

The management channel can also be used to pass "lost" SMI control messages. During normal operation, when the application issues an L4L3mENABLE_PROTOCOL message to configure a protocol for a specific stream, the Netaccess Series 7.5 Driver stores the LAP-D ID and LLI. When the Netaccess Series 7.5 Controller generates a control message, the driver searches all open connections to match the LAP-D ID and LLI indicated in the control message header

to a particular file descriptor. If the driver cannot find a match, it directs the lost message up the management channel. Refer to *Section 3* for more information on enabling a management channel.

Tunable Parameters

You may need to “tune” certain driver parameters to optimize your application’s performance.

- `WAN_HANGUP_RED_ALARM` generates a HANGUP on each data stream when the Netaccess Series 7.5 Controller detects a Red Alarm condition. The default value for this feature is 0 (OFF); possible values are 0 (OFF) and 1 (ON).
- `WAN_AUX_ENAB_CHK` adds error checking for LAP-D ID/LLI (DLCI) usage. When this feature is enabled, L4L3mENABLE_PROTOCOL messages that specify a LAP-D/LLI combination already in use are rejected. The default value for this feature is 0 (OFF); possible values are 0 (OFF) and 1 (ON).
- `WAN_NSTREAMS` determines the maximum number of connections that can be opened to each Netaccess Series 7.5 Controller. The default value is 256.
- `WAN_RECV_PKT_OPT` determines if the driver implements receive packet optimization 1 at the driver level for data buffers. The default is 1 (ON); possible values are 0 (OFF) and 1 (ON).
- `WAN_HOST_BUFFERS` determines where the data channels buffers will be allocated and stored. Possible values are 1 (in HOST memory) or 0 (on board shared memory). The default value is 1 (HOST memory).
- `WAN_NTX_BUFS` specifies the number of transmit buffers per stream. The default value is 8; possible values are 1 to 64.
- `WAN_NRX_BUFS` specifies the number of receive buffers per stream. The default value is 8; possible values are 1 to 64.
- `WAN_NUM_BOARDS` specifies the number of PRI boards supported by the system. The default is 8; the maximum should be determined by the overall system performance.
- `WAN_TX_BUFFSZ` specifies the size of the on-board transmit control buffers. The default value is 256; possible values are 2 to 4528.
- `WAN_RX_BUFFSZ` specifies the size of the on-board receive control buffers. The default value is 256; possible values are 2 to 4528.

Modifying Tunable Parameters

In order to modify tunable parameters, the valued must be modified from default in the `/usr/local/wandrv/driver/wandrvnfo.h` file. After the modifications have been made the following commands will rebuild and install the new driver.

1. `cd /usr/local/wandrv/driver`
2. `make clean install`
3. `/etc/rc.d/init.d/wandrv restart`

Driver Services

Introduction

This section describes the Netaccess Series 7.5 Linux Driver services available for application processes through UNIX system calls. Instances of *errno* set by the driver are listed. These values are defined by UNIX in the file *errno.h*. *Table 3-1* lists each service and describes its purpose; the services are listed in alphabetical order in the subsections that follow.

Table 3-1. Netaccess Series 7.5 Linux Driver Services

Service	Purpose
open	Open a connection to the driver for reading or writing
close	Dismantle a connection to the driver
select	Indicates when a file descriptor is available for access.
ioctl	Retrieve board data structures

Note: The **open** and **close** services are common to all UNIX environments. The subsections that follow provide basic information on these services as they relate to Netaccess Series 7.5 T1/E1/PRI board applications only. For complete descriptions and usage guidelines for these services, refer to the standard UNIX man pages.

Table 3-2 lists the appropriate system call errors in the Linux driver and their respective file names. These call errors are used in conjunction with the driver services.

Table 3-2. Netaccess Series 7.5 Linux System Call Errors

	ENOSPEC	ENOMEM	EIO	ERESTARTSYS	EAGAIN	ENOTCONN	ENXIO	EOPNOTSUPP	EMSGSIZE
open	X	X							
close									
read			X	X	X				
write				X	X	X	X	X	X
select									

Table 3-3 lists the possible error codes that may be returned for each Netaccess Series 7.5 service call. Table 3-3 lists codes that are specifically defined for the Netaccess Series 7.5 Driver in the wandrv.h file.

Table 3-3. Netaccess Series 7.5 Driver-Specific Errors For IOCTL Services

	ERESTARTSYS	ENOMEN	EINVAL	ENXIO	EBUSY	EACCESS	EIO	EUNATCH	ENODEV	EMSGSIZE	EFAULT	EAGAIN
WANDRViocBOOT								X			X	X
WANDRViocENA_MGT_CHAN				X	X							
WANDRViocGET_ISB				X							X	
WANDRViocGET_LOGOUT				X				X			X	
WANDRViocGET_MSG	X										X	X
WANDRViocGET_VERSION				X							X	
WANDRViocPUT_MSG	X	X		X					X	X	X	X
WANDRViocRESET_BOARD				X								
WANDRViocREBOOT_BOARD				X								
WANDRViocTEST				X			X					
WANDRViocGET_BOARD_TYPE												

ioctl - WANDRViocTEST

Structure

```
#include <linux/wandrv.h>
int ioctl (fd, WANDRViocTEST, 0);
```

Usage The ioctl WANDRViocTEST command performs a 256 byte I/O test on a portion of the shared memory between the host and the board.

Return

0	successful
ENXIO	Board represent by fd is not present.
EIO	Test found a bad memory call.

ioctl - WANDRViocBOOT Download Code to the Netaccess Series 7.5 Controller

Structure

```
#include <wandrv.h>
int ioctl (fd, WANDRViocBOOT, &cntl_ptr)
int fd;
WAN_BOOTPARAMS cntl_ptr;
```

Usage This service resets a board and downloads Instant ISDN Software. The `boot_board` utility can also perform this function, as well as configure the board's signaling and clocking characteristics. If you choose to write your own download utility, use `boot_board` as a template; refer to *Section 4* for more information on this utility.

fd is the file descriptor for the device returned by `open`.

bootparams is a structure defined to carry the user area address for the download file and its length.

Example

```
bootparams.useraddr = bootfile;
bootparams.userlen = stab.st_size;
if(ioctl (fd, WANDRViocBOOT, &bootparams) (0);
    error ("WANDRViocBOOT");
    exit (ERROR_EXIT);
}
close (fd);
```

Returns Upon successful completion, a 0 is returned; otherwise, a value of -1 is returned and *errno* is set to indicate the error. Possible error numbers are listed below:

- | | |
|--------|--|
| ENXIO | Hangup received on <i>fd</i> ; indicates that no board in the system uses the board ID referenced in the service call. |
| EINVAL | Invalid value; indicates the size of the download image is greater than the buffer can accept. |

ioctl - WANDRViocENA_MGT_CHAN Enable a Management Channel

Structure

```
#include <wandrv.h>
int      ioctl (fd, WANDRViocENA_MGT_CHAN, 0)
int      fd;
```

Usage Certain messages generated by the board pertain to the board rather than a particular connection to on the board. To receive these messages, the application must enable a management channel for the board. You should maintain a separate management channel stream for each board. Messages directed to a management channel include all L3L4mLINE_STATUS alarm messages and any other messages the driver cannot deliver to any other streams.

fd is the file descriptor returned by **open**.

Example

```
if (ioctl (fd, WANDRViocENA_MGT_CHAN, 0) < 0
      exit_errno ("Failed to enable management channel
= %sin", boot_path);
```

Returns Upon successful completion, a 0 is returned; otherwise, a value of -1 is returned and *errno* is set to indicate the error. Possible error codes are listed below:

EBUSY	Channel busy; Indicates a management channel already exists for the WAN Controller.
ENXIO	Hangup received on <i>fd</i> ; indicates that no board in the system uses the board ID referenced in the service call.

ioctl - WANDRViocRESET_BOARD Reset a Netaccess Series 7.5 Controller

Structure

```
#include <wandrv.h>
int ioctl (fd, WANDRViocRESET_BOARD, 0)
int fd;
```

Usage This service resets the board.
fd is the file descriptor returned by **open**.

Example

```
if (ioctl (fd, WANDRViocRESET_BOARD, 0) (0);
    exit_errno ("reset failed")
```

Returns Upon successful completion, a 0 is returned; otherwise, a value of -1 is returned and *errno* is set to indicate the error. Possible error codes are listed below:

ENXIO Hangup received on *fd*; indicates that no board in the system uses the board ID referenced in the service call.

ioctl - WANDRViocGET_BOARD_TYPE Return the type of Netaccess Series 7.5 Controller

Structure

```
#include <wandrv.h>
int ioctl (fd, WANDRViocGET_BOARD_TYPE, int)

int fd;
int brd_type;
```

Usage This service gets the board type and returns it to the host.
fd is the file descriptor returned by open .

Example `ioctl (fd, WANDRViocGET_BOARD_TYPE, &brd_type;`

Returns Upon successful completion, a 0 is returned;

ioctl - Retrieve Board Data Structures

Structure

```
#include <wandrv.h>
int      ioctl (fd, WANDRViocGET_LOGOUT, &logout)
int      ioctl (fd, WANDRViocGET_ISB, &isb)
int      ioctl (fd, WANDRViocGET_VERSION, &version)
int      fd;
IISDN_LOGOUT logout;
IISDN_INTERRUPT_STATUS_BLOCK isb;
char version [IISDN_VERSION_STRING_LEN];
```

Usage

This service retrieves diagnostic trace information, interrupt status information, and the Instant ISDN Software revision level from a board.

fd is the file descriptor returned by **open**.

cntl_ptr points to an address of the appropriate data structure as shown in the example on the following page. This module contains the following structures:

WANDRViocGET_LOGOUT is used to retrieve the 240 byte IISDN_LOGOUT structure. This should be accessed after a board failure. Note that this data structure is in Motorola format and therefore is not byte swapped.

WANDRViocGET_ISB retrieves a 16 byte IISDN_INTERRUPT_STATUS_BLOCK structure. This structure can be polled during the download sequence to determine when the board has successfully passed its diagnostics. This data structure also contains useful information when the board generates a software panic or crashes. Note this data structure is in Motorola format and therefore is not byte swapped.

WANDRViocGET_VERSION returns a 32-byte string from the board indicating the Instant ISDN Software revision level.

Example

```
if (ioctl (fd, WANDRVIocGET_LOGOUT, flogout) (0)
    exit_errno ("couldn't get logout info = %sin", boot_path);
if (ioctl (fd, WANDRVIocGET_ISB, fisb) (0)
    exit_errno ("couldn't get interrupt status block in");
if (ioctl (fd, WANDRVIocGET_LOGOUT, flogout) (0)
    exit_errno ("couldn't get version string in");
```

Returns

Upon successful completion, a 0 is returned; otherwise, a value of -1 is returned and *errno* is set to indicate the error. Possible error codes are listed below:

ENOMEM	No memory available; indicates no buffer was allocated to store the data, interrupt status block on version string.
ENXIO	Hangup received on <i>fd</i> ; indicates that no board in the system uses the board ID referenced in the service call.
EINVAL	Invalid value; indicates that the service call specified an invalid data buffer size.

WANDRViocGET_MSG, WANDRViocPUT_MSG

Structure

```

int          ioctl(fd, WANDRViocPUT_MSG, &outmsg);
int          ioctl(fd, WANDRViocGET_MSG, &inmsg);
L3_to_L4_struct &inmsg;
L4_to_L4_struct &outmsg;
```

Usage This service transmits an L4L3 control message to a board or receives an L3L4 control message from the board.

fd is the file descriptor returned by **open**.

This service allows the application to pass control messages to the L4L3 control queue. Length cannot exceed 512 bytes (the maximum size of a control message) but can be reduced to as little as 8 bytes for control messages which do not possess any data (such as the L4L3mDISABLE_PROTOCOL control message).

The buffer pointed to by *buf* contains a LAP-D iframe to be transmitted.

Returns Upon successful completion a 0 is returned; otherwise, a value of -1 is returned and *errno* is set to indicate the error. Possible error codes are listed below:

EBUSY	Mount device busy; indicates the stream is not available for passing the control message.
ENOMEM	No memory available; indicates no buffer was allocated to receive the control message.
ENOTCONN	Not connected; indicates the driver transmitted data before enabling a data channel to the board.
EOPNOTSUPP	Operation not supported; indicates the driver attempted to transmit data over a Q.931 (call control) channel.
EMSGSIZE	Message size; indicates the driver transmitted a data packet greater than 260 bytes, a control packet less than 8 bytes, or a control packet greater than 512 bytes
ENODEV	No device; indicates the driver transmitted a control packet before initializing the board

Note: Driver-specific error codes (ENOTCONN, EOPNOTSUPP, EMSGSIZE, and ENODEV) are sent upstream to the stream head and the stream is marked so that all subsequent system calls issued to the stream, excluding **close** and **poll**, fail with *errno* set to one of the values above. Normally, the system call after the offending **putmsg** returns the failure code and the offending **WANDRViocPUT_MSG** command completes with no error report. To recover from this condition, close the stream and reopen it.



Hot Swap

Introduction

This section describes the Netaccess Series 7.5 Linux Driver Hot Swap (HS) implementations of the driver and host application usage.

Overview Note

You can safely skip this section of the manual if you do not have a hot-swappable board. Currently, only the PRI-cPCI, NS300 and NS301 boards are considered to be hot-swappable.

During installation, be sure to say 'Y' to the question about Generic Hot-Swap. Alternatively, you can recompile the driver with `-DHOT_SWAP` set.

Implementation

For the HS implementation in the Netaccess Series 7.5 driver, a lowest denominator method was used. This implementation will allow the driver to support HS on any chassis without any processor dependancies.

Hot swap can be performed without the knowledge of the host application, this is suitable when the board is not being used. However, more likely than not the host app would like to be notified of board extraction and insertion. To that end, a HS Management Channel has been defined.

The HS Management Channel (HSMC) must be opened on a file descriptor all unto itself. After a successful `open(1)` call, issue **WANDRViocENA_HS_MGT_CHAN** to make that stream a HSMC. All messages received on this stream will be of the type **WAN_HS_STRUCT**.

```
typedef struct {
short board_num;
int opnstmnt;
short msgtype;
short timer;
} WAN_HS_STRUCT;
```

Configuration

When sending the **WANDRViocENA_MGT_CHAN** to enable the HSMC, configuration information can be sent in within a **WAN_HS_CONFIG_STRUCT**.

```
typedef struct {  
    int extract_time;  
    int insert_time;  
    int requireinsauth;  
    int requireextauth;  
} WAN_HS_CONFIG_STRUCT;
```

The extract and insert times could be any arbitrary number. In all cases you would want to enable extraction authorization, but insertion authorization may not be required.

Note: On a PCI-cPCI board, these values are ignored as the board get inserted/extracted immediately without host interaction. With this board, an HSMC can be opened with any values. If an HSMC is opened on a PRI-cPCI board and an HS event occurs, then the host will be notified with a **HS_BOARD_EXTRACTION** or **HS_BOARD_INSERTION** containing a timeout of 0.

Usage

Note: When extracting a PCI-cPCI or NS300/NS301 board, it must be replaced with the same board type (ie: a PCI-cPCI board can not be replaced with a newer NS300 board and vice-versa).

Extraction (NS300/NS301)

Begin the extraction process by lowering the bottom latch of the board, the driver sends a **HS_BOARD_EXTRACTION** message on the HSMC.

Contained within the structure is also a timer value which indicates when forceful extraction will occur. Once the host app has finished using the board and has remapped all its resources to a different board, an authorization by way of a **HS_AUTHORIZE_EXTRACTION** ioctl must be sent. Following this, the blue light on the board will illuminate indicating that it is safe to remove the board from the chassis.

Extraction (PCI-cPCI) with the

Begin the extraction process by lowering the bottom latch of the board, the driver sends a **HS_BOARD_EXTRACTION** message to the HSMC.

Contained within the structure is a timer value which will be set to 0 indicating that the board extraction has already been processed and no authorization from the host is necessary. At this point, the host should close all non-HSMC file descriptors for that board (no **DISABLE_PROTOCOLS** necessary) and continue to monitor for HS events.

Insertion (NS300/NS301)

Begin the insertion process by inserting the board into a previously occupied slot and raise the bottom latch. The driver will detect the insertion and send a **HS_BOARD_INSERTION** on the HSMC if the `requireinsauth` value was set during the creation of the HSMC. The host app can authorize the insertion by sending a **HS_AUTHORIZE_INSERTION** ioctl within the timer or wait for the timer to expire.

Insertion (PCI-cPCI)

Begin the insertion process by inserting the board into a previously occupied slot and raise the bottom latch. The driver will detect the insertion and send a **HS_BOARD_INSERTION** on the HSMC with the timeout value set to 0. When the host application receives the notification of insertion, it is safe to reopen file descriptors to the device, redownload the IISDN firmware and resume normal operation.



High Availability

Introduction

This chapter describes the High Availability (HA) implementations of the driver and host application.

Overview Note

You can safely skip this section of the manual if you do not have a High-Availability Chassis (Motorola 8216 or 8221) with redundant CPUs running Motorola-MCG's High-Availability Linux software. When installing the driver, be sure to say 'y' to the Motorola HA question. Alternatively you could recompile the driver with `-DHIGH_AVAIL` set.

Requirements

Chassis - Currently on the Motorola 8216 and 8221 chassis's are supported.

Operating system - Redhat Linux (distributed by Motorola-MCG) for X86 and LinuxPPC-2000 (distributed by Motorola-MCG) for Motorola MCP750's.

Processor Boards - CPV5XXX for X86 and MCP750 for PPC.

In addition to the specific kernel that is distributed with the Motorola distributions coupled with additional kernel updates by Motorola, a High Availability software package from Motorola is required that controls the HSC (Hot Swap Controller) cards.

Configuration

In order to get the High Availability software to configure the PRI-cPCI or NS300/NS301 boards, it is necessary to add sections to the `em.def` file in `/etc/scem`. For the PCI-cPCI, you must add a section under I/O for the bridge at device type 0x604 (PCI) with vendor/device combo of 0x1011/0x0024. For the NS300/NS301, under I/O at device type 0x608 (BRIDGE) add a section with vendor/device combo of 0x1011/0x0046.

New ioctl's

`HS_QUIESCE` - Quiesce the board.

HS_ENABLE - Enable the board for use.

Usage

Some sample programs have been included to show the usage of the PRI-cPCI and NS300/NS301 boards in a Motorola High-Availability environment. The High-Availability control program is located in the /usr/local/wandrv/ha_control directory. This application must be run on both CPUs (selecting option 1 'CS_to_Active' on the primary indicating Cold Standby to Active and 2 'CS_to_HS' on the secondary indicating Cold Standby to Hot Standby). The basic sequence of commands is as follows.

1. Start the SCEM software
2. Start the Netaccess Series 7.5 driver
3. Run boot_board to download the firmware to the Netaccess Series 7.5 Controller
4. Issue an HS_ENABLE command to enable the controller for use.

If you wish to force a failover to the secondary cpu, number 3 'HS_to_Active' can be selected via the ha_control application. This will forcibly take control of the domains and reactivate the Netaccess Series 7.5 controller under the new driver.

Additional information and demonstrations will be added as the driver's HA capabilities are supplemented.



Driver Utilities

Introduction

This section describes the Netaccess Series 7.5 Driver utilities available for use in application development. *Table 6-1* lists each utility and describes its purpose; the utilities are listed in alphabetically order in the subsections that follow.

Table 6-1. Netaccess Series 7.5 Driver Utilities

Utility	Purpose
boot_board	Download and initialize a specific Netaccess Series 7.5 board
pridump	Obtain crash dump information following a Netaccess Series 7.5 board failure
s_2_bin	Convert Motorola S-Records to binary for Netaccess Series 7.5 board

boot_board

Structure

boot_board

Usage

This utility downloads Instant ISDN Software to a Netaccess Series 7.5 board and configures it for proper operation. The utility is merely an example of the functionality and may require additional configurations of the driver. Contact Netaccess Series 7.5 Customer Support for further details. This utility can be called from the UNIX shell (or in an rc file, preferably), but can be executed by a superuser only. **boot_board** performs the following steps:

- Copies the PRI software object file (naii_0.bin or naii1.bin) from disk to system main memory
- Opens the Netaccess Series 7.5 device driver

Upon completion, **boot_board** reports on the success or failure as reflected by the return value and value of *errno*.

pridump

Structure pridump <board number>

Where board number is an integer from 0 to x. The default is board 0.

Usage This utility prints out the IISDN_VERSION, IISDN_LOGOUT and IISDN_INTERRUPT_STATUS_BLOCK data structures. This utility is typically called following a Netaccess Series 7.5 Controller failure; refer to the appropriate *Technical Description* for the SYSFAIL LED indications on a Netaccess Series 7.5 board.

Example Output

```
Version :PRII48D Rev6.7.35 12Apr00
```

```
PRI board 0 traceback for vector 0
```

```
D: [0000.0000] [0000.0000] [0000.0000] [0000.0000]
   [0000.0000] [0000.0000] [0000.0000] [0000.0000]
A: [0000.0000] [0000.0000] [0000.0000] [0000.0000]
   [0000.0000] [0000.0000] [0000.0000] [0000.0000]
```

```
Stack:
```

```
20C5  8B B886 20DB 2700  0 27BE  0
BCE   0 3966  0  0 800  50 1300
  50 5100  50 FFF0  0  0  0  0
  0  0  0  0  0  0  0  0
14FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF  0  1  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0
```

```
Interrupt Status Block:
```

```
int_rsn = x3 error_code = x0 error_point = x0 error_arg =
x0
```

```
timestamp = x0 spare1 = x0 spare2 = x0 vme_irq_level = x0
```




Appendix A

Brooktrout Customer Support

Customer First

Brooktrout is committed to delivering complete customer satisfaction. We endeavor to make all of our products reliable, easy to install and easy to use. If you need additional technical assistance after reading the *Technical Description* and *Programmer's Manual*, Brooktrout provides access to a wide range of service and support offerings.

Before You Call

When contacting Brooktrout Customer Support, please call from a location where you can operate your system, including the ability to remove or look at the hardware components of the Brooktrout card.

Contacting Brooktrout Customer Support

Purchase of a Brooktrout Developer's Kit entitles you to 12-months of unlimited technical support during standard business hours. Our Technical Assistance Center also offers a wide variety of additional fee-based support services, including Extended 24-hour Support Plans, Training Classes, Application Developer's Lab, On-Site Support, and more.

Brooktrout Customer Support can be reached via phone, fax, email and postal mail. Our support center is staffed Monday through Friday, 8:30am to 5:30pm, EST.

Please fill out and return the Warranty Registration Card supplied with your Brooktrout product. Your information will be entered into our Support Database for product tracking and will facilitate better customer support to your business center.

Mailing Address	Brooktrout Technology, Inc. 18 Keewaydin Drive Salem, NH 03079	
Support Phone	(781) 433-9600 (781) 449-9009 (Fax)	8:30 am - 5:30 pm EST, Mon - Fri 24 hours
eMail	techsupport@brooktrout.com	

Additional Brooktrout Support Services

Using an Internet connection, you can access our web page on the World Wide Web:

<http://www.brooktrout.com>

From our web page, you can access the latest technical and product information and the latest versions of our device drivers, FAQ files – you can even open a trouble ticket with a Brooktrout Support Engineer.

B

- binary file format 1-3
- boot_board 1-2, 3-4, 6-2, 6-4
- Brooktrout Customer Support A-1
- Brooktrout Driver
 - media format 1-3
 - removing a previous version 2-3
- Brooktrout Driver utilities
 - boot_board 1-2
 - pridump 1-3
 - s_2_bin 1-3
- Brooktrout WAN Driver
 - test programs 2-4

C

- close 1-1
- code fragment
 - receiving events 2-9

D

- data interface channels 2-7
- downloading Instant ISDN Software 3-4, 6-2
- driver services
 - close 1-2
 - ioctl 1-2, 3-4, 3-5, 3-6, 3-8
 - open 1-2
 - select 1-2
 - WANDRviocGET_MSG,
WANDRviocPUT_MSG 3-10
- driver-specific error codes 3-10

E

- EACCESS 3-2
- EAGAIN 3-2
- EBUSY 3-2, 3-10
- EFAULT 3-2
- EINVAL 3-2, 3-4
- EIO 3-2
- EMSGSIZE 3-2, 3-10

- ENODEV 3-2, 3-10
- ENOMEM 3-10
- ENOMEN 3-2
- ENOTCONN 3-10
- ENXIO 3-2, 3-4, 3-6
- EOPNOTSUPP 3-10
- ERESTARTSYS 3-2
- EUNATCH 3-2

F

- file descriptor 2-7

H

- HANGUP event 2-10
- HDLC channels 1-2, 2-7
- High Availability (HA) 5-1
- HS_AUTHORIZE_EXTRACTION 4-2
- HS_AUTHORIZE_INSERTION 4-3
- HS_BOARD_EXTRACTION 4-2
- HS_BOARD_INSERTION 4-3
- HS_ENABLE 5-2
- HS_QUIESCE 5-1

I

- installing the Netaccess Series 7 Driver 2-1
- ioctl 1-2, 2-7, 2-8, 3-4, 3-5, 3-6, 3-8

L

- L3_to_L4_struct 2-8
- L3L4mLINE_STATUS message 2-9, 3-5
- L3L4mPROTOCOL_STATUS message 2-7
- L4L3mDISABLE_PROTOCOL message 3-10
- L4L3mENABLE_PROTOCOL message 2-7, 2-9,
2-10
- LAP-D ID 1-2, 2-7
- LAP-D virtual circuits 1-2
- logical channels 2-7
- Logical Link ID (LLI) 1-2, 2-7

M

management channel 2-9, 3-5
 Motorola S-record format files 1-3

N

Netaccess Series 7 board
 resetting the board 3-6
 Netaccess Series 7 Driver
 installing the files 2-1
 number of Netaccess boards supported 1-1
 Netaccess Series 7 Linux Driver services 3-1

O

open 1-1

P

pridump 1-3, 6-3
 pristat 6-4

R

Red Alarm condition 2-10
 related publications 1-1
 removing a previous version of the WAN driver 2-3
 retrieving messages/data packets from a stream 2-8

S

s_2_bin 1-3, 6-4
 select 1-2, 2-8
 sending messages/data packets down a stream 2-8

T

tunable parameters 2-10

U

utilities
 boot_board 1-2, 6-2
 pridump 1-3, 6-3
 s_2_bin 1-3, 6-4

V

virtual circuits 1-2

W

WAN_AUX_ENAB_CHK 2-10
 WAN_HANGUP_RED_ALARM 2-10
 WAN_HOST_BUFFERS 2-10
 WAN_HS_CONFIG_STRUCT 4-2
 WAN_HS_STRUCT 4-1
 WAN_NRX_BUFS 2-10
 WAN_NSTREAMS 2-10
 WAN_NTX_BUFS 2-10
 WAN_NUM_BOARDS 2-10
 WAN_RECV_PKT_OPT 2-10
 WAN_RX_BUFFSZ 2-10
 WAN_TX_BUFFSZ 2-10
 WANDRViocBOOT 3-4
 WANDRViocENA_HS_MGT_CHAN 4-1
 WANDRViocENA_MGT_CHAN 4-2
 WANDRViocGET_ISB 3-8
 WANDRViocGET_LOGOUT 3-8
 WANDRViocGET_MSG, WANDRViocPUT_MSG
 3-10
 WANDRViocGET_VERSION 3-8