

User's Guide

HDLC LAPB Protocol Software for Solaris Systems

Solaris 2.5 , 2.5.1, 2.6 and Solaris 7 for
SPARC, SPARC V9 and Intel Platforms

Copyright © 2000, CNT and Aurora Technologies, Inc., a Carlo Gavazzi Group Company
All Rights Reserved
Printed in the United States of America

This publication is protected by Federal Copyright Law, with all rights reserved. This online version may be freely printed and distributed internally, but cannot be modified, in whole or in part, or included in any other work without prior written consent from Aurora Technologies, Inc.

Limitation of Liability

Aurora Technologies, Inc. makes NO WARRANTY, EXPRESSED or IMPLIED, with respect to this manual, and any related items, its quality, performance, merchantability, or fitness for any particular use. It is solely the purchaser's responsibility to determine its suitability for any particular use.

Information contained in this document is subject to change without notice.

Trademark Credits

Aurora Technologies, the Aurora logotype, Apollo Multiport, Aries Multiport, Control-Tower, Explorer Multiport, LanMultiServer, Saturn Multiport, SBox, and WanMultiServer are trademarks of Aurora Technologies, Inc., a Carlo Gavazzi Group company.

All other registered trademarks and salesmarks are the proprietary property of their respective owners.

About this Manual

The *HDLC LAPB User's Guide* describes how to install and use the Aurora Technologies LAPB software for Solaris systems.

Who Should Use This Book

This book is a guide for anyone who wants to install, configure, and use the Aurora Technologies LAPB subset of the HDLC protocol software. It is assumed that the reader is familiar with Solaris device drivers, STREAMS I/O, and data link protocols.

System Requirements

Aurora's controllers are designed to work with a wide range of Sun Solaris systems on either the SPARC or X86 platforms. Your Solaris system must meet the following system requirements:

Workstation:	Any Solaris system
Operating System:	Solaris 2.5 through Solaris 7
CPU:	SPARC, SPARC V9, x86
Memory:	32 Mbytes minimum, 64 Mbytes recommended
Disk Drive:	2 Mbyte free in /opt (Solaris 2.5 through Solaris 7)
Media Drive:	CD-ROM


Related Documentation

For additional information, refer to the following manuals:

- Your Solaris system's Installation Guide
- Your Solaris system's Owner's Guide
- Your Solaris documentation
- Your peripheral's documentation.

Documentation Conventions

Unless otherwise noted in the text, this document uses the following symbolic conventions.

<code>screen text</code>	ASCII text that the system displays, pathnames, and commands that appear in the text appear in <code>plain typewriter font</code> .
screen display	Graphic text that appears on menus and dialog boxes appears in sans serif font.
<code>literal input</code>	Bold words or characters in formats and command descriptions represent commands that you must type literally. Literal values are in the <code>bold typewriter font</code> .
<i><non-literal input></i>	<i><Bold, italic, bracketed></i> words or characters in formats and command descriptions represent values that you must supply. Do not type the brackets.
<i>emphasis</i>	<i>Italics</i> are used in the text for emphasis, titles, and variables.
□	This symbol indicates the end of the chapter text.
✍	This symbol marks procedures.
	This symbol marks cautionary notes about possible damage to your equipment or data.

Getting Help

If you need to reach us, you can contact us in the following ways:

- World Wide Web: <http://www.auroratech.com>
- Phone: Mon–Fri, 8:30–6:00 Eastern Time
- FAX: Attn: Customer Service and Support
- Email: support@auroratech.com
- Mail: Attn: Customer Service and Support

Warranty Registration

To receive warranty coverage on your Aurora hardware, you must fill out and mail back the Aurora Warranty Registration Card that is located in the back of this manual. Phone support can only be provided after product registration is complete. Hardware and Software Maintenance Agreements can be provided for extended customer support.

Sending in this card also lets us keep you up-to-date on the complete line of Aurora Technologies' products.

If you have any questions or comments on your Aurora Technologies' product, contact your reseller. If you cannot contact your reseller, feel free to contact us directly. We're always listening to you! □

Contents

PREFACE	<i>About this Manual</i>	iii
	Who Should Use This Book	iv
	System Requirements	iv
	Related Documentation	v
	Documentation Conventions	vi
	Getting Help.....	vii
	Warranty Registration.....	vii
CHAPTER 1	<i>Aurora LAPB Overview</i>	1
	HDLC Networking	1
	Operational Modes	2
	Link Stations	3
	Frame Types	3
	Frame Format	4
	Flag Framing and “Bit Stuffing”	5
	Address Field.....	5
	Control Field.....	5
	Information Field.....	6

Frame Check Sequence	6
Included Software.....	7
CHAPTER 2 <i>Software Installation for Solaris Systems</i>	9
Before Installing the LAPB Software	10
Installation.....	11
Miscellaneous Installation Procedures	15
Removing an Old Version of HDLC/LAPB.....	15
Obtain and Install License Key.....	17
Building the Example Program	18
Checking the LAPB Software Installation	19
Bypassing vold.....	20
LAPB Device File Names	21
CHAPTER 3 <i>Basic Link Operation</i>	23
System Setup.....	24
Link Setup.....	26
Link Normal Data Flow.....	27
Link Shutdown	28
CHAPTER 4 <i>Message Formats</i>	29
Control Portion.....	30
Data Portion.....	31
Message Header	32
Activate Link	33
Activate Station.....	34
Configure Link	34
Configure Station.....	37
Deactivate Link.....	40
Deactivate Station	40
Link Inoperative	40
Request Measurements	41

Report Measurements	43
Station Activated	46
Station Inoperative	46
CHAPTER 5 <i>Example Program Sources</i>	47
Example Program Description	48
Configuring the Example	49
Running the Example	49
APPENDIX A <i>Configuration Example</i>	51
APPENDIX B <i>Warranty</i>	53
Warranty Information	53
Hardware	53
Return Policy	54
90 Day Technical Support	54
Software License Agreement	56
Aurora Technologies Software License	56
INDEX	59
PRODUCT INFORMATION WORKSHEET	61

Aurora LAPB Overview

The Aurora LAPB (Link Access Procedure, Balanced) subset of the High-level Data Link Control (HDLC) software package provides you with tools for creating Solaris system applications that use LAPB-protocol data links. The software supports Solaris 2.5 through Solaris 7, on either SPARC or X86 platforms.

HDLC Networking

Aurora's HDLC LAPB implementation is one of a family of synchronous, bit-oriented data link protocols derived from SDLC (Synchronous Data Link Control). Related protocols include SDLC, LAP and LAPD. HDLC is a superset of LAPB and SDLC. LAPB is the data link layer of the CCITT/ITU X.25.

All of these synchronous protocols share a common frame format, which features flag framing and bit stuffing, (see page 5) instead of control characters, to mark the frame and the fields within it. The LAPB protocol can transmit a large data field and incorporates flow, as well as, and error control mechanisms as part of the frame.

Operational Modes

SLDC related bit-oriented synchronous protocols can be grouped into three operational modes, which define a basic set of commands and responses: normal response, asynchronous response, and asynchronous balanced. Optional commands can be added to this core command/response repertoire to enhance the protocol's functionality.

The LAPB implementation of HDLC is commonly used only for the X.25 data link.¹

¹. There is a DOD specification for the transport of IP packets within LAPB frames. Some routers and WAN bridges use LAPB to carry network layer IP, IPX, DECNET or APPLE TALK packets, where backward error correction on a per link basis can improve end-to-end performance. Many legacy applications use LAPB for transport.

Link Stations

An HDLC/LAPB link converts a physical link between adjacent network nodes or stations into a perfect logical error free link. Unlike SDLC and complete HDLC, which can support multi-point networks with three kinds of stations (primary, secondary, and combined), the LAPB subset of HDLC provides a point-to-point full duplex (two-way simultaneous) link between two peer stations, a DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). Both peer stations can send or receive commands and responses.

SDLC and complete HDLC implementations have more flexibility serving end-user needs. For example, SDLC has commands and responses in its command set which make it possible to set up a network loop topology. LAPB does not support this type of SDLC loop.

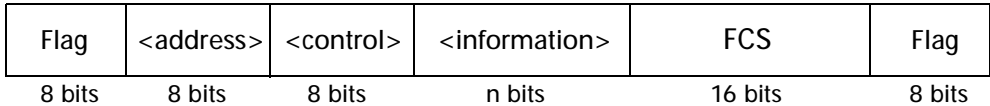
Frame Types

There are three types of HDLC/LAPB frames: *information frames*, *supervisory frames*, and *unnumbered frames*.

- Information frames transport the data across the link.
- Supervisory frames provide flow control and error recovery for information frames.
- Unnumbered frames are used to establish and terminate connections across a physical link.

Frame Format

The general format of the HDLC/LAPB frame structure is shown in Figure 1. This format applies to all three frame types used in LAPB: information frames, supervisory frames, and unnumbered frames.



Flag = 01111110
<address> = sender or receiver of the frame
<control> = includes flow control information and identifies frame type,
<information> = data to be sent to network layer.
FCS = Frame Check Sequence

FIGURE 1. *HDLC/LAPB frame format*

Flag Framing and “Bit Stuffing”

A *frame* is a variable-length package of data which starts and ends with a *flag*. The flag defines the location of the other fields in the frame. Because the flag provides synchronization, it is important to ensure that the flag’s special bit pattern is always distinctive among the other fields in the frame, and that no false flags accidentally intrude into the rest of the frame’s data stream.

To ensure this, LAPB, like the other bit-oriented protocols, uses a technique called *bit-insertion* during transmission of the other fields in the frame. During bit-insertion, if a sequence of more than five consecutive “one” bits occurs in the address, control, information, or frame check sequence (FCS) fields during transmission, the bit-stuffing algorithm will insert a zero bit after the fifth “one.” The receiver will delete these inserted zeros during reception.

Address Field

Versions of SDLC and HDLC which serve multiple stations provide for address fields extended beyond the standard single octet. Single byte address fields are sufficient for LAPB, however, since LAPB only communicates point-to-point between DTEs and DCEs.

Control Field

The control field contains information which distinguishes among the three frame types; for example, whether the frame is an information frame carrying user data, or a supervisory frame providing link control.

Information Field

The information frame is transparent and passes the user data to the next functional level.

The information field can be any length. Some versions of SDLC and HDLC, including Aurora's, require that the information field be organized in octets.

Frame Check Sequence

This two octet field verifies the integrity of the transmitted frame by checking for data errors during transmission.

Included Software

Included in the package is the Aurora LAPB subset of the HDLC STREAMS pseudo-device driver, source code for an example program, and a makefile for building the example program.

The example program (and its makefile) can be used for testing the pseudo-device driver and as a departure point for developing your applications.

The LAPB STREAMS pseudo-device driver is a clonable device driver. The application does not care which minor device is allocated on an open request. All are equivalent in providing functionality. Parameters used to control the operation of a LAPB link are passed to the LAPB STREAMS pseudo-device driver via configuration messages following pseudo-device open (see “Link Setup” on page 26). By default, the first LAPB pseudo-device opened is designated the system control point (see “System Setup” on page 24). The BIOGETMINDEV and BIOSETADMIN commands enable more complex configurations.

The LAPB STREAMS pseudo-device driver uses the services of the Aurora synchronous device driver. The Aurora synchronous driver is configured through the Aurora HDLC/LAPB driver.).

The pseudo-device driver can be installed in one of two modes: standard and passive. The standard mode supports all typical LAPB functionality. The passive mode will not initiate a data link by sending out an SABM/SABME (Set Asynchronous Balanced Mode) frame but will wait to receive a SABM/SABME frame from its peer. Note that all links are either exclusively active or passive.

Software Installation for Solaris Systems

This chapter includes the following sections:

- Before Installing the LAPB Software
- Installation Instructions
- Miscellaneous Installation Procedures
- Checking the LAPB Software Installation
- Bypassing vold
- LAPB Device File Names

The CD-ROM shipped with your controller card is labeled with the appropriate commands for starting the driver installation.

Before Installing the LAPB Software

The Aurora synchronous driver must be loaded before the LAPB software can be run. Be sure to load the Aurora driver when you install your Multiport card. If this driver wasn't loaded when your Aurora card was installed, you will have to load it before using LAPB. You can continue with this installation, but LAPB will not run until the driver is loaded.

Note that the Aurora LAPB software will be installed into either `/opt/AURAbhdlc` or `/opt/AURAbhdlp`.

Before beginning the installation, please record the requested product information on the "Product Information Worksheet" at the back of this *User's Guide*, and mail your Product Registration Card, found in the back of this manual.

Installation

Experienced installers may find enough information in this section that they will not need to read further. Before beginning the installation, please record the requested product information in the back of this manual and mail your Product Registration Card.

There are two LAPB packages on the CD-ROM. AURAbhdlc is the standard, full functionality LAPB package. AURAbhdlp is a *passive* LAPB package. It is a non-standard implementation that does not initiate a link by sending out an SABM(E) frame. You can only install one package.

✍ *Preparatory Steps to Installation*

1. Verify that the Aurora synchronous driver is loaded.
2. Remove existing Aurora LAPB packages using `pkgrm`. (Disable processes on ports if necessary.)
3. To properly install the driver you need to know three attributes of your system. If you can answer the following questions you are ready to install the driver.
 - 1 Is Volume Manager (`vold`) running on your system? Your possible answers are Yes or No.
 - 2 What is the system architecture? Your possible answers are SPARC, SPARCV9 or x86.
4. What Solaris operating system version of the driver do you need to install? This release supports Solaris 2.5, 2.5.1, 2.6 and Solaris 7. Therefore, to install the driver your answer is limited to those listed.

If the Solaris volume management daemon `vold` is running on your system, it automatically mounts the Aurora CD-ROM when you insert it in the CD-ROM drive. If `vold` is not running, you will have to mount the CD-ROM manually.

To check if `vold` is running, type:

```
system# ps -elf | grep vold
```

If you see a line with `/usr/sbin/vold` in the far right column, `vold` is running on the system.

If you do not see a line with `/usr/sbin/vold` in the far right column, `vold` is not running on the system.

Installation

1. Login as, or become, root:

```
system% /usr/bin/su
Password: <root_password>
```

2. Insert the CD-ROM into the CD-ROM drive.
3. Install using with or without Volume Manager running.

- If Volume Manager (`vold`) **IS** running, then type:

```
system# volcheck
```

Mount the CD-ROM as `/cdrom/hdlc_5_00_ed1`. Proceed to step 4.

- If your system does not have Volume Manager (`vold`) enabled, you must mount the disk manually. To do this, you must know the device pathname for the CD-ROM drive. On SCSI-connected SPARC systems, it is typically `/dev/dsk/c0t6d0s0`. If you do not know your CD-ROM pathname, see your system administrator. To proceed when Volume Manager **IS NOT** running, create the mount point.

- 1 Type:

```
system# mkdir /mnt
```

2 Mount the CD-ROM:

```
system# mount -rF hsfs <cdrom-device-name> /mnt
```

Where <cdrom-device-name> is the device name of your CD-ROM drive. For example:

```
system# mount -rF hsfs /dev/dsk/c0t6d0s0 /mnt
```

See the `mount(1M)` man page for additional information. If you are unable to mount the CD-ROM, ask your System Administrator for help.

4. To install the driver, use the following command template as appropriate.

```
system# pkgadd -d <MOUNTPOINT><ARCHITECTURE><SOLARIS_OS>
```

where <MOUNTPOINT> is one of:

- /cdrom/hdlc_5_00_ed1 when VOLD is running
- /mnt when VOLD is not running

where <ARCHITECTURE> is one of:

- /sparc for SPARC 32-bit architecture
- /sparcv9 for SPARC 64-bit architecture
- /i386 for x86 architecture

where <SOLARIS_OS> is one of:

- /SunOS5.5 for Solaris 2.5
- /SunOS5.51 for Solaris 2.5.1
- /SunOS5.6 for Solaris 2.6
- /SunOS5.7 for Solaris 7

As an example, to install the *Solaris 2.6* driver on a 32-bit *SPARC* system with *Volume Manager* running use the command:

```
system# pkgadd -d /cdrom/hdlc_5_00_ed1/sparc/SunOS5.6
```

See the `pkgadd(1M)` man page for additional information.

5. When you are prompted for the specific HDLC implementation, choose `AURAbhdlc` or `AURAbhdlp` as appropriate.

6. To verify that the driver is properly installed, type:

```
system# pkginfo -l <PKG>
```

where <PKG> is one of:

- /AURAbhdlc for the standard HDLC package
- /AURAbhdlp for the passive HDLC package

7. To detach from the cdrom device, type:

```
system# cd /
```

8. If Volume Manager **IS NOT** running, you must unmount the CD-ROM. Type:

```
system# umount /mnt
```

See the `umount (1M)` man page for additional information.

9. Eject the CD-ROM.

- If Volume Manger **IS NOT** running, press the eject button
- If Volume manager **IS** running, type:

```
system# eject hdlc_5_00_ed1
```

10. Obtain and install your EOD string. See “Obtain and Install License Key” on page 17.

11. Mail or FAX in your Product Registration Form.

Miscellaneous Installation Procedures

Use the procedures in this section to install the LAPB software on workstations running the qualified Solaris version. The first procedure removes old versions of LAPB (if you are installing LAPB for the first time you can skip this procedure). The second procedure installs the new LAPB software.

Removing an Old Version of HDLC/LAPB

If there is an old version of Aurora HDLC/LAPB software already installed on the workstation, you must use the following procedure to remove it before installing the new ones. If you are installing the LAPB software for the first time, skip this procedure and go on to *Obtain and Install License Key* on page 17.

✍ *To remove existing LAPB packages*

1. Log in as root by typing

```
login: root
password: <root_password>
```

2. Check for existing Aurora packages by typing

```
system# pkginfo | grep AURA
```

The system will display all the Aurora packages it finds. Look in the list for one of the following packages:

```
AURAbhdlc
AURAbhdlp
```

If the system does not find either of these packages, skip the rest of this procedure and go on to *Obtain and Install License Key*, below. Otherwise continue with Step 3.

3. Disable all processes on the communications controller ports.
4. Type the appropriate command, based on what `pkginfo` finds:

```
system# pkgrm AURAbhdlc
```

or

```
system# pkgrm AURAbhdlp
```
5. Type `y` in response to the resulting prompts.

Obtain and Install License Key

Aurora LAPB requires a license key in the form of an EOS string for correct operation.

To obtain a license, fill out and mail (or FAX) the Aurora Warranty Registration Card located in the back of this manual. Or contact Aurora Customer Support using the information provided in this manual's Preface.

When you contact Customer Support you will need to provide:

- Your Aurora LAPB serial number (from the box in which this manual was packed)
- The `hostid` of the system on which you have installed Aurora LAPB

You can enter this information in the Product Information Worksheet on the last page of this manual.

Building the Example Program

Included in the LAPB Software Package is source code for an example program that you can use to test the LAPB pseudo-device driver, and as a departure point for developing your own applications. Use the following procedure to build the example program.

Note: You must have obtained your EOS (Encoded Option String) before you can build the example program.

✍ To build the example program

1. Create a unique directory for the example with read and write permission.
2. `cd` to the new directory.
3. Copy the example into that directory:

```
system% cp /opt/AURAbhdlc/example/* .
```
4. Give yourself read and write permission for the example makefile:

```
system% chmod 644 Makefile
```
5. Install the license key by editing the makefile; replace the string in the security field of the makefile with the fourteen characters of the EOS.
For SPARC:

```
CFLAGS= -DSECURITY=<EOS> -DSUNOS4 -DSVR4 \  
-DSPARC $(DFLAGS)
```


For X86:

```
CFLAGS= -DSECURITY=<EOS> -DSUNOS4 -DSVR4 \  
-DI86 $(DFLAGS)
```
6. Build the example program (You will need a C compiler):

```
system% make
```

7. List the current directory contents to check the result:

```
system% ls -l
```

You should see an entry for `example` or `example*`.

Checking the LAPB Software Installation

By attaching a cable to two LAPB ports in a loop-back configuration, you can check the LAPB software installation. Use the following procedure. For cabling information, refer to the user manual shipped with your Aurora communications hardware.

✍ *To check the LAPB software installation*

1. Connect a synchronous cable to two of the LAPB ports.
2. Login as root.
3. Run the example program:

```
system# ./example <port1> <port2>
-->
```

where `<port1>` and `<port2>` are the two ports connected in Step 1 and `-->` is the example program prompt. Make sure `<port1>` is connected to the clock side of the loopback cable. See *LAPB Device File Names* on page 21 for proper `<port1>` and `<port2>` values. For example,

```
system# ./example sync/0 sync/1
```

is what you would type if you connected the clock side of the cable to the first Aurora port and the other end to the second Aurora port on workstation.

4. Start sending frames from the first port you specified in Step 3 to the second:

```
--> s s1 <optional text>
sending on s1
received on s2 <optional text>
```

The example program should display the messages `sending on s1` and `received on s2` to confirm data is being sent from the first port (s1) and received at the second port (s2).

5. Now send frames from the second port to the first port:

```
--> s s2 <optional text>
sending on s2
received on s1 <optional text>
```

Again, the appropriate messages should be displayed. If you can send data in both directions between s1 and s2, the installation was successful.

6. Type a question mark to display additional commands:

```
--> ?
```

7. Type `q` to quit:

```
--> q
```

Bypassing vold

Solaris has a utility called `vold` that controls CD-ROM drives. If you get the message `cannot access device` when you try to run `pkgadd`, it could be due to `vold` restricting access to the CD-ROM drive. In this case, please consult your Solaris system administrator for assistance. Bypassing `vold` is a system dependent operation.

LAPB Device File Names

Each LAPB port on which the LAPB protocol is running must be identified by an appropriate device file name. When you install the synchronous driver for your Aurora communications controller, Solaris device files are created for each port on the card.

Table 1 shows the device files created for the Aurora synchronous driver running on an eight-port card installed in the system's first available slot and a four-port card installed in the next available slot.

TABLE 1. *Solaris Device File Names for Synchronous Ports*

Port Label	LAPB Device Name
<i>Controller card 1 (eight-port)</i>	
0	/dev/sync/0
1	/dev/sync/1
2	/dev/sync/2
3	/dev/sync/3
4	/dev/sync/4
5	/dev/sync/5
6	/dev/sync/6
7	/dev/sync/7
<i>Controller card 2 (four-port)</i>	
0	/dev/sync/8
1	/dev/sync/9
2	/dev/sync/10
3	/dev/sync/11

Basic Link Operation

The following sections describe the link setup, link normal data flow, and link shutdown of an LAPB pseudo-device.

System Setup

The LAPB pseudo-device driver (`/dev/hdlc`) requires the use of an administrative logical channel as shown in Figure 2. The file descriptor obtained the first time the LAPB pseudo-device is opened in each process or process group is designated the administrative logical channel. All administrative requests, except configuration, are sent downstream on this administrative logical channel, and all administrative responses are sent upstream on it.

The administrative logical channel provides an entry point into the LAPB pseudo-device driver for user-controlled administration, while not imposing any restrictions on the STREAMS application construction above it. If multiple processes are going to control different sets of ports, or another process or process group starts, it will set up a separate administrative logical channel. Each process should use the `BIOGETMINDEV` (Get minor device) and `BIOSETADMIN` (Set administrative channel) `ioctl`s to establish the administration channel.

Different processes and process groups can open different LAPB ports. These LAPB ports or groups of ports correspond to different physical ports on the Aurora serial adapters, and groups of ports must be non-intersecting.

Access to the administrative logical channel within a process group is enforced by the LAPB pseudo-device driver. If the administrative logical channel is closed while the LAPB channels that correspond to the physical ports are still open, the open LAPB channels will still try to use the non-existent administrative logical channel; this can cause the system to crash.

Note: It is important to close all LAPB channels that correspond to the physical ports *before* closing the administrative logical channel.

The administrative logical channel is not an active LAPB link (i.e. no bit synchronous device is linked below it). No user data is transmitted or received on the administrative logical channel. Any user output data messages (message_id = XMIT_DATA) sent on the administrative logical channel are silently discarded.

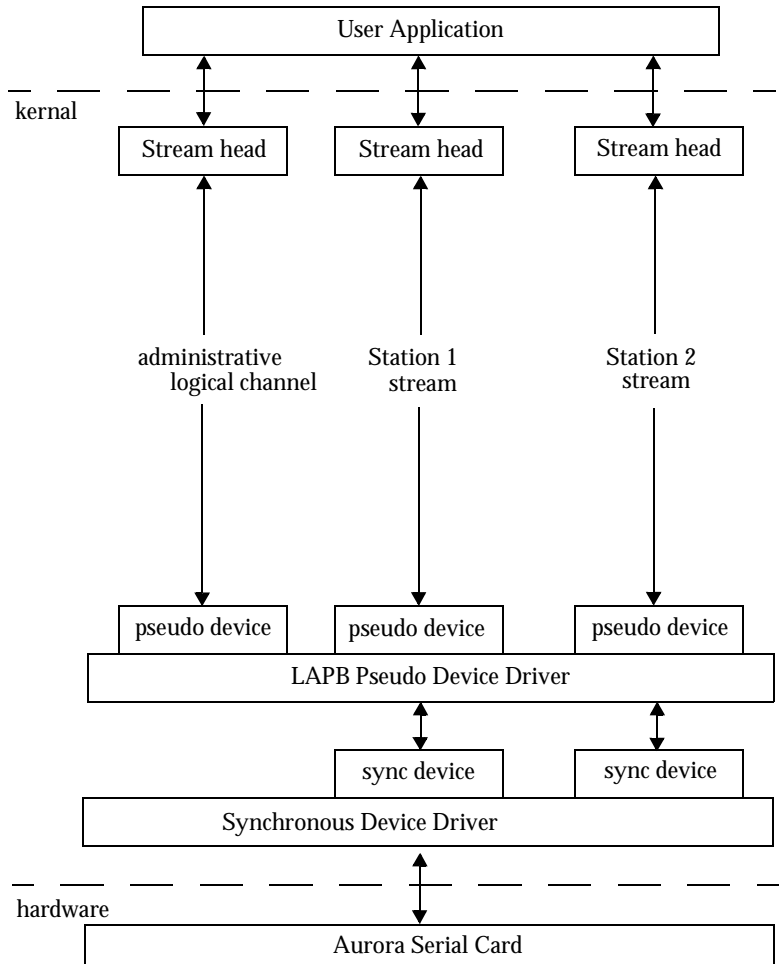


FIGURE 2. Aurora LAPB Link Architecture

Link Setup

The device driver link is set up as follows:

A.) The user application opens a bit synchronous device (for example, `/dev/sync/0`).

B.) The user application opens a LAPB pseudo-device using the LAPB clonable device open call (i.e. `/dev/hdlc`).

C.) The user application issues a `configure link` message on the LAPB stream.

The response to this message is read on the administrative logical channel (refer to “System Setup” on page 24).

D.) The user application issues a `configure station` message on the LAPB stream. The response to this message is also read on the administrative logical channel.

E.) The user application issues an “I_LINK” STREAMS `ioctl` command to link the bit synchronous device driver below the LAPB driver. The user application may now close the bit synchronous device stream and free the file descriptor.

F.) The user application issues an `activate link` request on the administrative logical channel.

The LAPB pseudo-device driver responds to this message on the administrative logical channel. If all is well, the modem control signals are brought up.

G.) The user application issues an `activate station` request on the administrative logical channel. The response to this message is read on the administrative logical channel.

H.) Upon a successful SABM(E)/UA exchange with the remote station, the LAPB pseudo-device sends two messages upstream:

- A `station activated` message is sent on the administrative logical channel.
- A `hdlc-msg` with `message_id = ACTIVE` is sent on the LAPB pseudo-device upward stream.

Link Normal Data Flow

The LAPB pseudo-device link normal data flow allows asynchronous transmission and reception of data messages between the LAPB pseudo-device driver and the stream head/application. STREAMS flow control is implemented to restrict the user from overrunning the link and exhausting the system buffer pool.

If the user does not accept device input data fast enough, and STREAMS flow control is exerted on the LAPB pseudo-device driver, back flow control to the remote station is introduced via the LAPB protocol RNR command.

The user application may use asynchronous I/O to the LAPB pseudo-device via the STREAMS `poll(2)` system call, `signal(2)` system call, and STREAMS `I_SETSIG ioctl`. Or the application may perform blocking I/O.

Link Shutdown

The user application can perform an orderly shutdown of the LAPB pseudo-device as follows:

A.) The user application issues a `deactivate station` request on the administrative logical channel. The response message is read on the administrative logical channel.

B.) On successful DISC/UA exchange with the remote station, the LAPB pseudo-device sends two messages upstream:

A `station inoperative` message is sent on the administrative logical channel.

An `hdlc msg` with `message_id = INOP` is sent on the LAPB pseudo-device upward stream.

If it is known that the peer station will also deactivate the link from the peer's side, the link may also be shut down as follows:

C.) The user application issues a `deactivate link` request on the administrative logical channel. The response message is read on the administrative logical channel. The modem control signals are dropped at this time.

NOTE: This shut down will be graceful only if both sides issue the `deactivate link` requests simultaneously.

D.) The user application can issue `close(2)` system calls at this time and terminate.

The user application may perform a discourteous disconnect by simply issuing `close(2)` system calls on the open file descriptors and/or terminating.

NOTE: If the peer station does not shut down and a discourteous disconnect is initiated, the peer station may read very large frames with bad CRCs.

Message Formats

The messages passed between the LAPB stream head modules and the user application consist of two parts: a control part and a data part. These messages may be sent and received by the user application using the `getmsg(2)` and `putmsg(2)` system calls. These system calls preserve the distinct parts of the message and message boundaries. The user application specifies the control part via the control pointer and length, and the data part via the data pointer and length.

Control Portion

The control portion of the LAPB message consists of an `hdlc_msg` structure. This structure is used to identify the message as a main path data message or an administrative message. This portion of the message is contained in a STREAMS `mb1k` of type `M_PROTO`. The `hdlc_msg` structure is defined as follows:

```
typedef struct hdlc_msg {
    unsigned short msg_id;
} hdlc_msg_t;
```

where `msg_id` identifies the type of message contained in the Data part of the LAPB message. Possible values are defined as follows:

<code>XMIT_DATA</code>	A user data message to be transmitted on the link.
<code>RECV_DATA</code>	A user data message received on the link.
<code>ACTIVE</code>	Indication from the LAPB pseudo-device driver that the remote station has been contacted and information transfer is available.
<code>INOP</code>	Indication from the LAPB pseudo-device driver that contact with the remote link station has been lost and information transfer is not available.
<code>ADM_IN</code>	An administrative request/response from the LAPB pseudo-device driver to the user.
<code>ADM_OUT</code>	An administrative request from the user to the LAPB pseudo-device driver.

Data Portion

The data portion of the LAPB message consists of the data transmit/received on the link or an administrative request/response. This portion of the message is contained in a linked list of STREAMS `mb1ks` of type `M_DATA`.

User data is not formatted or validated and is simply placed into, or extracted from, the I-field of a valid information frame. Administrative requests and response messages are strictly formatted as described below. The standard set of stream interfaces (`putmsg`, `putpmsg`, `getmsg`, `getpmsg`) should be used to send/receive on the LAPB stream.

Message Header

All administrative messages (control portion `msg_id` set to `ADM_IN` or `ADM_OUT`) begin with a message header. This structure is defined as follows:

```
typedef struct adm_msg {
    unsigned leng          msg_id;
    unsigned leng          cor_id;
    unsigned leng          status;
    unsigned char          data;
} adm_msg_type;
```

`msg_id` is the administrative message identifier. Defined values are as follows:

ACT_LINK	Requests activation of a specific LAPB link.
ACT_STN	Requests activation of a specific LAPB link station.
CFG_LINK	Provides configuration information for a LAPB link.
CFG_STN	Provides configuration information for a LAPB link station.
DACT_LINK	Requests deactivation of a specific LAPB link.
DACT_STN	Requests deactivation of a specific LAPB link station.
LINK_INOP	Informs the application that a specific LAPB link is inoperative.
REQ_MEAS	Requests a measurements report.
RPT_MEAS	Reports measurements for a specific link or station.
STN_ACT	Informs the application that a specific LAPB link station is active and ready for information transfer.
STN_INOP	Informs the application that a specific LAPB link station is inoperative and not available for information transfer

`cor_id` is the user correlation identification. The value specified on the user request is returned, unchanged, in this field.

`status` defines the completion status of the corresponding request. If this field is non-zero, the requested operation was not successful. Refer to specific requests for the possible return values.

`data` is the beginning of the user data. The supplier user data is specific to each request type.

Activate Link

The `activate link` message is used to physically enable a LAPB link. The LAPB link and station must be configured and the bit synchronous device must be linked below the LAPB pseudo-device before this message is issued by the user (refer to “Link Setup” on page 26).

This message causes the current link configuration to be downloaded into the bit synchronous device driver and the modem control signals activated. The format of the activate link message is simply the element address of the desired link.

Possible error returns:

LINK_NOT_FOUND	A link with the specified ea could not be located.
NO_PORT	The bit synchronous device has not been linked below the LAPB pseudo-device.

Activate Station

The `activate station` message is used to place a station in link setup mode. The LAPB link and station must be configured and the bit synchronous device must be linked below the LAPB pseudo-device before this message is issued by the user (refer to “Link Setup” on page 26). The LAPB link does not need to be active prior to issuing this message.

The format of the `activate station` message is simply the element address of the desired station.

Possible error returns:

LSTN_NOT_FOUND	A link station with the specified ea could not be located.
NO_PORT	The bit synchronous device has not been linked below the LAPB pseudo-device.

Configure Link

The `configure link` message passes the user selected operational parameters to the LAPB pseudo-device driver. These parameters are used on receipt of a link activation message. To simplify the configuration process, this message is sent on the LAPB pseudo-device stream, as opposed to the administrative logical channel.

This allows the configuration to be associated with a specific stream without requiring the user to identify the stream to the administrator. From this point on, all administrative messages referring to this link are sent on the administrative logical channel and identify the stream via the element address.

The configuration data is not validated for correctness. It is assumed the application will provide correct configuration data.

The currently defined legal values are below. The format of the configure link message is as follows:

```
/*
 * HDLC Link Configuration
 */

typedef struct {
    unsigned int    authentication;
    int             cd_monitor;
    enum speed_t    speed;
    int             maxdata;
    unsigned int    clock:2;
    unsigned int    nrzi:1;
    unsigned int    duplex:1;
    unsigned int    line:1;
    unsigned int    resv:11;
    element_id_t    ea;
    char             name[NAME_LEN];
    char             device[PATHNAME_ARRAY_LEN];
    int             physical_link_type;
} hdlclink_values_t;

/*
 * HDLC Configuration Constants
 */

#define NO_CD_MONITR           0
#define CD_MONITR             1

#define EXTERNAL_CLOCK        0
#define INTERNAL_CLOCK        1

#define NRZ_ENCODING           0
#define NRZI_ENCODING         1

#define DUPLEX_FULL            0
#define DUPLEX_HALF           1

#define LINE_LEASED            0
#define LINE_SWITCHED         1

/*
 * db 05/15/91 - note that the addresses are the WRONG
 * way round!
 */
#define DCE_LINK_ADDR 3
#define DTE_LINK_ADDR 1

#define DEFAULT_T1             80
#define DEFAULT_T2             30
```

maxdata	maximum # of bytes the current driver can support.
authentication	internal string used to verify user access rights. Generated from vendor based on hostid. (old authentication)
cd_monitor	Use Carrier Detect as physical connect/disconnect indicator.
speed	The speed of the link for DTE provided (internal) clock. Defined speeds are as follows:
	BPS1200 BPS2400 BPS4800 BPS9600 BPS19200 BPS56000 BPS64000 BPS128000 BPS256000
clock	One of the legal values must be specified if you are providing clock.
nrzi	Set to 1 if the data encoding is NRZI.
duplex	Set to 1 if half duplex.
line	Set to 1 if switched line.
ea	System-wide unique element address (identifier).
name	User defined name (ASCII string).
device	EOS. (new security)

Possible error returns:

LINK_NOT_AVAIL

Attempt to configure the administrative logical channel.

Configure Station

The `configure station` message passes the user selected operational parameters to the LAPB pseudo-device driver. These parameters are used on receipt of a station activation message. To simplify the configuration process, this message is sent on the LAPB pseudo-device stream, as opposed to the administrative logical channel. This allows the configuration to be associated with a specific stream without requiring the user to identify the stream to the administrator. From this point on, all administrative messages referring to this station are sent on the administrative logical channel and identify the stream via the element address.

The configuration data is not validated for correctness. It is assumed the application will provide correct configuration data. Incorrect configuration can cause system crashes.

The format of the configure station message is as follows:

```
/*
 * HDLC Station Configuration
 */

typedef struct {
    int                maxframe;
    int                outframe;
    enum modulo_t     modulo;
    int                t1;
    int                t2;
    int                idleRR;
    int                retry_count;
    element_id_t      ea;
    unsigned char     addr[LSTN_ADDR_LEN];
    char              name[NAME_LEN];
} hdlcst_values_t;
```

maxframe	Maximum frame size for transmit/receive frames on the station.
outframe	Maximum number of unacknowledged information frames allowed.
modulo	Frame sequence number modulus, defined values are: MOD8 and MOD 128.
t1	T1 timer value.
t2	T2 timer value.
retry_count	Maximum retries to transmit an unacknowledged frame.
ea	System-wide unique element address (identifier).
addr	Station address (01 or 03).
name	User defined name (ASCII string).
idleRR	Generate RR if no response in T2 seconds.

The values in the table above must be configured. See “Configuration Example” on page 51 for an example.

Possible error returns:

LINK_NOT_FOUND

The current link has not been configured.

LINK_NOT_AVAIL

Attempt to configure station on the administrative logical channel.

Deactivate Link

The `deactivate link` message is used to gracefully close down a link after a deactivate station has issued. On receipt of this message, the modem control leads are dropped and physical disconnect ensues. If the deactivate station command has not been issued, deactivate link performs an ungraceful close. The format of the `deactivate link` message is simply the element address of the desired link.

Possible error returns:

`LINK_NOT_FOUND`

A link with the specified `ea` could not be located.

Deactivate Station

The `deactivate station` message is used to gracefully close down a station. On receipt of this message, information transfer mode is exited. The format of the `deactivate station` message is simply the element address of the desired station.

Possible error returns:

`LSTN_NOT_FOUND`

A link station with the specified `ea` could not be located.

Link Inoperative

The `link inoperative` message is used to inform the application that a physical disconnect has occurred on the link. The format of the `link inoperative` message is simply the element address of the inoperative link.

Request Measurements

The `request measurements` message is used by the application to solicit report measurements message(s) from the LAPB pseudo-device driver. The application specifies the level of reporting requested via this message. The format of the `request measurements` message is as follows:

```
typedef struct {
    unsigned short    mrcode;
    unsigned short    mrcode;
    unsigned short    md_reset:1;
    unsigned short    ms_reserved:14;
} ms_hdr_type;
```

`ms_code` specifies the measurements requested. Values are defined as follows:

MS_ALL	Report measurements for all links and all link stations.
MS_LINKS	Report measurements for all links.
MS_LINK	Report measurements for a specific link. The specific link is identified via the following <code>na_id_type</code> .
MS_LSTNS	Report measurements for all link stations.
MS_LSTN	Report measurements for a specific link station. The specific link station is identified via the following <code>na_id_type</code> .
<code>ms_last</code>	This field is not used in the request measurements message.
<code>ms_reset</code>	If set to 1, reset the measurements following the report.

```
typedef struct {
    unsigned short    id_len;
    unsigned short    ea;
} na_id_type,
```

`id_len` specifies the length in bytes of this id (i.e. 4).

`ea` identifies the specific element (link or link station) that measurements are requested for.

```
typedef struct {
    ms_hdr_type    rq_bx_hdr;
    na_id_type    na_id;
} rq_meas_type;
```

Possible error returns:

<code>LINK_NOT_FOUND</code>	A link with the specified <code>ea</code> could not be located.
<code>LSTN_NOT_FOUND</code>	A link station with the specified <code>ea</code> could not be located.
<code>MS_CODE_INV</code>	Invalid measurements request code specified.

Report Measurements

The `report measurements` message is used to respond to application requests for measurements. Each link and station measurements report is contained in a separate message. The last or only such measurements reported in response to any given request is indicated via the `ms_last` flag. The format of the `report measurements` message is as follows:

```
typedef struct {
    unsigned short      ms_code;
    unsigned short      ms_last: 1;
    unsigned short      ms_reset: 1;
    unsigned short      ms_reserved: 14;
} ms_hdr_type;
```

<code>ms_code</code>	This code specifies the measurements requested. Refer to the request measurements message for defined values.
<code>ms_last</code>	Set if this is the last report measurements message in response to a request for <code>MS_ALL</code> , <code>MS_LINKS</code> or <code>MS_LSTNS</code> .
<code>ms_reset</code>	This field is not used in the report measurements message.

```
typedef struct {
    unsigned short      obj_type;
    unsigned short      num_meas;
} obj_hdr_type;
```

`obj_type` defines the object type that the following measurements were taken from. Defined values are as follows:

<code>MS_OBJ_LINK</code>	The measurements were taken from a link. The specific link is identified via the following <code>na_id_type</code> .
<code>MS_OBJ_LSTN</code>	The measurements were taken from a link station. The specific link station is identified via the following <code>na_id_type</code> .

```
typedef struct {
    bit16              id_len;
    bit16              ea;
} na_id_type;
```

`id_len` specifies the length in bytes of this id (i.e. 6).

`ea` identifies the specific element (link or link station) that this report measurements message is for.

```
typedef struct f
    unsigned char        meas_code;
    unsigned char        meas_type;
    unsigned char        meas_valid;
    unsigned char        meas_len;
    unsigned char        meas_value[4];
    meas_bin_type;
```

`meas_code` identifies the specific measurement. Defined values are as follows:

MS_LNK_MSGS_RCVD	Number of frames received on this link.
MS_LNK_CHARS_RCVD	Number of characters received on this link.
MS_LNK_RCV_BAD_ADDR	Received frames with invalid link address.
MS_LNK_RCV_BAD_CNTL	Received frames with invalid control field.
MS_LNK_RCV_CRC	Received frames with invalid CRC.
MS_LNK_RCV_SIZE	Frames received exceeding maximum frame size.
MS_LNK_RCV_OVR	Receiver overruns.
MS_LNK_RCV_FA	Received frame aborts.
MS_LNK_MSGS_SENT	Number of frames sent on this link.
MS_LNK_CHARS_SENT	Number of characters sent on this link.
MS_LNK_XMT_UNR	Transmit underruns.
MS_LNK_XMT_WDT	Transmit watchdog time-out.
MS_LNK_RCV_ALLOCB	Receive frames dropped due to STREAMS buffer denial.

MS_LNK_RCV_CANPUT	Receive frames dropped due to STREAMs flow control between the bit synchronous device driver and the LAPB pseudo-device driver.
MS_STN_MSGS_SENT	Number of Information frames sent on this station.
MS_STN_CHARS_SENT	Number of Information characters sent on this station.
MS_STN_MSGS_RCVD	Number of Information frames received on this station.
MS_STN_CHARS_RCVD	Number of Information characters received on this station.
MS_STN_REXMITS	Number of retransmitted Information frames on this station.
MS_STN_RESP_TO	Number of remote response time-outs.
MS_STN_RMT_BUSY	Number of remote busy conditions occurring.
MS_STN_LCL_BUSY	Number of local busy conditions occurring.

`meas_type` identifies the specific type measurement. Defined values are as follows:

MS_COUNTER	This measurement is a simple counter.
MS_HIGH_TIDE	This measurement reports the greatest value since the last reset.
MS_FLAG	This measurement is the current value of a flag.

`meas_valid` Set if measurement invalid (i.e. counter overflow).

`meas_len` The length in bytes of the following `meas_value`.

meas_value

The value of the specified measurement.

```
typedef struct {
    ms_hdr_type          rp_bx_hdr;
    obj_hdr_type        rp_obj_hdr;
    a_id_type           rp_na_id;
    meas_bin_type       rp_meas[201];
} rp_meas_type;
```

Station Activated

The `station activated` message is used to inform the application that a station has entered information transfer mode. The format of the `station activated` message is simply the element address of the activated station.

Station Inoperative

The `station inoperative` message is used to inform the application that a station has exited information transfer mode. The format of the `station inoperative` message is simply the element address of the inoperative station.

Example Program Sources

An example application program is shipped with the Aurora LAPB software. Running this example program allows you to run a simple loopback test of the LAPB installation (see Figure 3 on page 48) and provides a starting point for developing LAPB applications.

Note: While all the structures in the example program say “HDLC,” the program is really testing the LAPB implementation.

Example Program Description

The example program consists of two source files, `example.c` and `hdlc_interface.c`, six header files, `typedefs.h`, `bx_user.h`, `hdlc_user.h`, `brx_scope.h`, `brx_general.h`, and `example.h`, and the make file `Makefile`. During installation of the LAPB software, the example files are placed in `/opt/AURAbhdlc/example`.

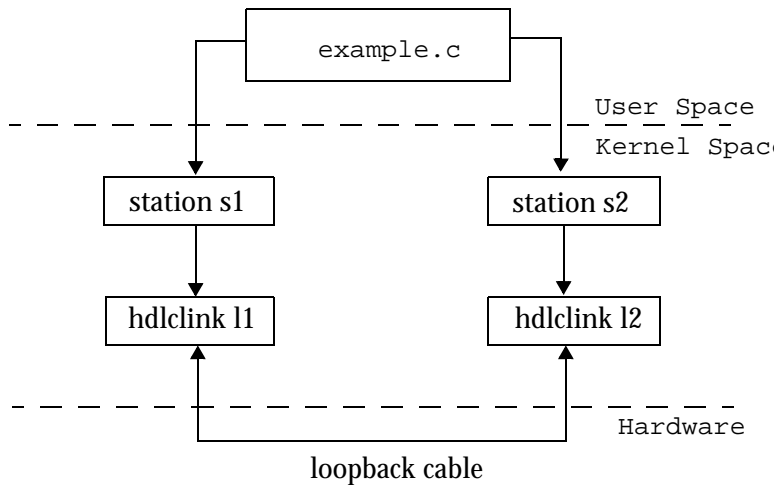


FIGURE 3. Loopback test using the example program

Configuring the Example

Before you can use the example software, it must be configured to match your hardware setup. This is done by editing the station and link values in the file `example.c`.

Running the Example

Once `example.c` is edited and saved, you can make and run the example program. Don't forget to connect the loopback cable first.

✍ To run the example program

1. Change to the example program directory

```
% cd /opt/AURAbhdlc/example
```

2. Make the example program by typing

```
% make
```

3. Run the example program by typing

```
% ./example
```

4. Send data to station 1 by typing

```
% s s1
```

You should see the following message:

```
received 145 bytes of data on station s1
```

If you see the expected message, it means your LAPB installation and default data are working. If you do not get this message, reinstall Aurora LAPB. If the example program still does not run correctly, call Customer Support.

Configuration Example

Configuration examples are available on the distributed CD-ROM. Refer to `example.c` in the following directories: `/opt/AUAbhdlc/example` and `/opt/AURAbhdlp/example`. Various functions illustrate link and station initialization for HDLC/LAPB.

Refer to the release note in the top directory level of the CD-ROM for more details.

Warranty

Warranty Information

Hardware

All Aurora Technologies, Inc. hardware products are warranted against defects for two years from the date of delivery. Buyer agrees that if this product proves defective, Aurora Technologies, Inc. is obligated only to repair, replace or refund the purchase of this product at Aurora Technologies, Inc.'s discretion. The warranty is void if the product has been subjected to alteration, neglect, misuse or abuse; if any repairs have been attempted by anyone other than Aurora Technologies, Inc.; or if failure is caused by accident, acts of God, or other causes beyond the control of Aurora Technologies, Inc.

Aurora Technologies, Inc. reserves the right to make changes or improvements in any product without incurring any obligation to similarly alter products previously purchased. In no event shall Aurora Technologies, Inc. be liable for any direct, indirect, incidental or consequential damages arising out of or in connection with the performance or use of the product or information provided. Aurora Technologies, Inc.'s liability shall in no event exceed the purchase price of the product purchased hereunder.

The foregoing limitation of liability shall be equally applicable to any service provided by Aurora Technologies, Inc.

Application and Protocol Software

Limited Software Warranty. Aurora Technologies, Inc. does not warrant that the functions contained in its Software products will meet your requirements or that the operation of the Software will be uninterrupted or error free. However, Aurora warrants the physical media on which the Software is furnished will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of shipment.

Except as provided above, the software and its written materials are provided “As-Is,” without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the software is with the licensee. Should the software prove to be defective, licensee assumes the entire cost of necessary servicing, repair, or correction.

Return Policy

Products returned for repair must be accompanied by a Return Material Authorization (RMA) number, obtained from Aurora Technologies, Inc. prior to return. Freight on all returned items must be prepaid by the customer, and the customer is responsible for any loss or damage caused by common carrier in transit. Items will be returned to Aurora Technologies, Inc. via a Customs cleared carrier (for example, Federal Express, UPS, DHL), unless prior arrangements are made by the customer for an alternative shipping method. Each product that is returned for repair must include a failure report and must have the RMA number clearly marked on the outside packaging.

Return Address: Aurora Technologies, Inc.
626 Summer Street
Brockton, MA 02302

90 Day Technical Support

Products must be registered with Aurora Technologies, Inc.’s Customer Service and Support (CSS) organization to receive the 90 Day Technical Support. You must fill out and mail or FAX the warranty card that is included with the product before receiving technical assistance.

What you get during the 90 Day Technical Support

90 Day Technical Support is provided by e-mail, FAX, or by telephone. Customers calling in for technical support on current Aurora technologies, Inc. products will receive a response within four (4) business hours. Customer's e-mails or faxed requests will receive a response within twenty-four (24) business hours.

The Technical Support hours in Massachusetts are

8:30 a.m. - 6:00 p.m. Eastern Time,
Monday through Friday, excluding holidays.

Services provided under the 90 Day Technical Support Plan are:

- Help on installation and configuration
- Help diagnosing problems with Aurora hardware and standard released Aurora device drivers.
- Help navigating and locating existing Aurora documentation
- Acceptance of bug reports and providing status updates on any applicable bug fixes.

Policies and pricing are subject to change without notice.

For extended support, please refer to the Aurora Technologies, Inc. web site at www.auroratech.com or call your Sales representative for details. □

Software License Agreement

THIS LEGAL DOCUMENT IS AN AGREEMENT BETWEEN YOU, THE END USER OR “LICENSEE”, AND AURORA TECHNOLOGIES, INC. (“AURORA” OR “LICENSOR”). BY OPENING THIS DISTRIBUTION MEDIA PACKAGE, YOU ARE AGREEING TO BECOME BOUND BY THE TERMS OF THIS AGREEMENT, WHICH INCLUDES THE SOFTWARE LICENSE AND SOFTWARE WARRANTIES (COLLECTIVELY THE “AGREEMENT”).

THIS AGREEMENT CONSTITUTES THE COMPLETE AGREEMENT BETWEEN YOU, THE LICENSEE AND AURORA. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT OPEN THE DISTRIBUTION MEDIA PACKAGE. PROMPTLY RETURN THE UNOPENED PACKAGE, HARDWARE (IF ANY) AND THE OTHER ITEMS, INCLUDING WRITTEN MATERIALS, BINDERS AND OTHER CONTAINERS, THAT ARE PART OF THIS PRODUCT TO THE PLACE OF PURCHASE.

Aurora Technologies Software License

1. GRANT OF LICENSE: In consideration of payment of the License fee, which is part of the price paid for this product, Aurora, as LICENSOR, grants to you, the LICENSEE, a non-exclusive right to use and display this copy of an Aurora Software program, (the “Software”) on a single computer at a single location. If the single computer on which you use the Software is a multi-user system, this License covers all users of that system. Aurora reserves all rights not expressly granted to the LICENSEE.

2. DELIVERY, INSTALLATION, ACCEPTANCE AND RISK OF LOSS: Aurora shall deliver the Software to a common carrier, FOB Aurora’s Facilities. LICENSEE shall be solely responsible for installation of the Software on the computer. LICENSEE agrees the acceptance shall occur upon delivery of the Licensed Software to a common carrier. LICENSEE assumes all risk of loss or damage upon delivery of the Software by Aurora to LICENSEE or common carrier.

3. PAYMENTS: In consideration of the License and rights in the Software granted by Aurora and in consideration of Aurora’s performance of its obligations hereunder, LICENSEE agrees to pay Aurora Technologies, Inc. the License fee according to the terms as invoiced. Failure to remit complete payment according to the specified terms will result in revocation of the License upon thirty (30) days’ written notice.

4. OWNERSHIP OF SOFTWARE: As the LICENSEE, you own the magnetic or other physical media on which the Software is originally or subsequently recorded or fixed, but Aurora in no way transfers title or ownership of the Software recorded on the original distribution media and all subsequent copies of the Software, regardless of the form or media in or on which the original and other copies may exist. This License is not a sale of the original Software or any copy.

5. COPY RESTRICTIONS: This Software and the accompanying written materials are copyrighted. Unauthorized copying of this Software, including Software that has been modified, merged or included with other Software, or of the written materials is expressly forbidden. LICENSEE may be held legally responsible for any copyright infringement that is caused or encouraged by LICENSEE's failure to abide by the terms of this License. Subject to these restrictions, and if the Software is not copy protected, LICENSEE may make one (1) copy of the Software solely for backup purposes. LICENSEE must reproduce and include any copyright notice on the backup copy.

6. USE RESTRICTION: LICENSEE may physically transfer the Software from one computer to another provided that the Software is used on only one computer at a time. LICENSEE may not distribute copies of the Software or accompanying written materials to others. LICENSEE may not translate, decompile, disassemble or otherwise reverse engineer, or modify, adapt or create derivative works based on the Software. LICENSEE may not modify, adapt, translate or create derivative works based on the written materials without prior written consent of Aurora.

7. TRANSFER: LICENSEE may sell the License rights in the Software to another party only if that party also agrees to the terms and conditions of this Agreement. In accordance with such sale, the LICENSEE must simultaneously transfer any and all written materials and the backup copy, or destroy the backup copy.

8. TERM AND TERMINATION: The License is effective until terminated. This License will terminate automatically without notice from Aurora if LICENSEE fails to comply with any provision of this License. Upon termination you shall destroy the written materials and all copies of the Software, including modified copies, if any.

9. UPDATE POLICY: Aurora may create, from time to time, updated versions of the Software. At its option, Aurora may make such updates available to the LICENSEE and transferees who have purchased an Extended Support Plan from Aurora.

10. EXPORT RESTRICTIONS: The LICENSEE understands that the United States export control laws may govern the export and re-export of products to be licensed under this Agreement and that individual validated export licenses may be required from the U.S. Department of Commerce prior to the export of products. The LICENSEE agrees to assist the LICENSOR to obtain any required License by supplying appropriate documentation requested by the seller. The LICENSEE agrees to comply with the U.S. Export Administration Regulations in effect from time to time and will not re-export any products without first obtaining approval from the LICENSOR and the U.S. Department of Commerce as required. The re-export of LICENSOR's source code, whether modified or not, is prohibited without prior written approval from the LICENSOR. If the LICENSOR grants the LICENSEE approval to re-export source code, LICENSEE will obtain all required export approvals from the U.S. Department of Commerce prior to sale and shipment. The LICENSEE agrees to indemnify and hold harmless the LICENSOR from all costs and expenses incurred by the LICENSOR as a result of the LICENSEE's breach of this section.

11. RIGHT TO GRANT A LICENSE: Aurora hereby warrants that it has the right to grant a License to use the Software to the LICENSEE and that it has the right and power to enter into this License.

12. LIABILITIES: In no event will Aurora be liable for any lost revenues or profits or other special, indirect or consequential damages, even if Aurora has been advised of the possibility of such damages. Aurora's maximum liability for damage shall be limited to the License fees paid by the LICENSEE under this License for the particular Software which caused the damages.

13. GENERAL: LICENSEE may not sublicense, assign or transfer the License on the Software except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void. This Agreement will be governed by the laws of the Commonwealth of Massachusetts in the United States of America. □

Index

A

Administrative logical
channel 24–25

B

Basic 23
Basic link operation 23–28
Bit stuffing 5
Bit-oriented protocols 1
Bypassing void 20

C

Configuration
example 51
Customer service,
contacting vii

D

Device file names, LAPB 21
Disconnect, discourteous 28
Documentation
conventions vi

E

Example program 7, 17
description 48
directory 48, 49
running 49
sources, running 47–49

F

Flag framing 5
Frame structure 3
Frame types 3

H

Hardware
warranty 53–54
HDLC/LAPB, old versions,
removing 15

I

Installation procedures,
detailed 15

L

LANMultiServers
system requirements iv
LAPB device file names,
table 21
LAPB frame fields
address 5
control 5
frame check sequence 6

information 5

LAPB frame structure 3

LAPB message

- Activate Link 33
- Activate Station 34
- Configure Link 34
- Configure Station 37
- control portion 30
- Data portion 31–46
- Deactivate Link 40
- Deactivate Station 40
- Link Inoperative 40
- Message Header 32
- Report Measurements 43
- Request Measurements 41
- Station Activated 46
- Station Inoperative 46

LAPB overview 1–7

LAPB software

- checking installation 19
- installation procedure 17

Link normal data flow 27

Link setup procedure 26

Link shutdown 28

Link stations 3

Loopback test 47

M

Manuals, related v

Message formats 29–46

O

Operational modes 2

S

Software

- license agreement 56–58
- warranty 54

Software included 7

Software Installation for
Solaris Systems 9–21

Solaris Systems, software
installation 9–21

System Requirements iv

System setup 24

T

Technical support 54

V

vold, bypassing 20

W

WANMultiServers

- system requirements iv

Warranty

- Hardware 53–54
- Software 54

Product Information Worksheet

Please record the following information about your Aurora controller card and workstation:

LAPB software serial number: _____

Workstation/PC model: _____

Operating System version: _____

hostid of the system: _____

PCI card installed in slot number: _____

Peripheral/Port assignments:

Slot	Port	Peripheral
—	0	
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	

