



**Communication  
Automation  
Corporation**

---

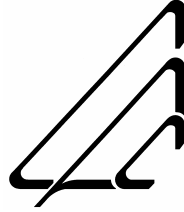
1180 McDermott Dr ♦ West Chester, PA 19380-4022

Tel: (610) 692-9526 ♦ Toll-free: (800) 367-6735 ♦ Fax: (610) 436-8258 ♦ <http://www.cacdsp.com> ♦ Email: [sales@cacdsp.com](mailto:sales@cacdsp.com)

# DPT4 HDLC API

Application Note:  
Version 1.1

email: [support@cacdsp.com](mailto:support@cacdsp.com)



© 2001-2004 Communication Automation Corporation  
(West Chester, PA (USA))

## License Agreement

The international copyright laws that pertain to computer software and hardware protect this Software/Hardware. It is illegal to duplicate the design and implementation of the Hardware and/or to make copies of the Software except as provided in this license agreement. It is illegal to give copies of CAC Software to another person, or to duplicate the Software by any other means, including electronic transmission, except as provided in this license agreement. CAC Software and Hardware contains trade secrets and, to protect them, you may not decompose, reverse engineer, disassemble, or otherwise reduce the applicable object code or binary portions of the Software or Hardware to human perceivable form.

Our software is a product of Communication Automation Corporation (CAC) and is licensed for unrestricted use WITH CAC HARDWARE PRODUCTS ONLY. CAC software may be reproduced and used by the customer only if this legend is included on all distribution media and this legend is included as a part of the software comments, whether the CAC software is used in whole or in part.

Users may copy or modify CAC software without royalty, but are not authorized to license, sublicense, or distribute this copied or modified CAC software to any other person or organization except as part of a hardware product or software developed by the user that incorporates CAC hardware products. You are permitted, however, to freely distribute your own derived software that communicates to the CAC boards through these software drivers and libraries, free of any royalty to CAC.

## Warranty

Communication Automation Corporation reserves the right to make changes to these products, including any software and/or hardware described herein, without notice. No warranty of merchantability or fitness for a particular purpose is expressed or implied. CAC shall not be held liable for incidental or consequential damages in connection with, or arising out of, the use of this Software. CAC does not recommend the use of any of its products, Software or Hardware, for medical or life support applications wherein a failure or malfunction of the product may threaten life or cause injury and will not knowingly license or sell its products for either such use. No rights under any patent accompany the sale of any such products.

## Trademarks

MS-DOS, Windows95, WindowsNT and Windows are registered trademarks of Microsoft Corporation in the United States of America and other countries.

QuickKit Telephony® and smPCI® are registered trademarks of Communication Automation Corporation.

SPARC, SunOS and Solaris are registered trademarks of Sun Microsystems Computer Corporation and SunSoft.

Unix is a registered trademark of Santa Cruz Operations.

Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark.

# Table of Contents

<b>1</b>	<b>Host DPT4 HDLC Library .....</b>	<b>1-1</b>
1.1	HDLC Library Overview .....	1-1
1.2	Initialize HDLC Library.....	1-2
1.3	Open HDLC Channels .....	1-2
1.4	HDLC Messages .....	1-5
1.4.1	Write Messages to HDLC Channel.....	1-5
1.4.2	Read Messages from HDLC channel.....	1-6
1.5	Close Channel.....	1-7

## 1 Host DPT4 HDLC Library

### 1.1 HDLC Library Overview

The Host DPT4 HDLC library provides functions for interpreting HDLC data streams received from E1 or T1 channels and for generating HDLC data streams transmitted on an E1 or T1 channel. The HDLC library uses the standard DPT4 library to read from and write to an E1 or T1 channel or bonded set of channels. The API provided by the standard DPT4 library can be used along with the HDLC library.

Prototypes and defines for the HDLC library are contained in the include file `dphdlc.h`.

Definitions in the standard DPT4 include file `dputil.h` are also required.

See the “Programmer’s Manual for Desktop PCI Telephony”.

## 1.2 Initialize HDLC Library

The HDLC library needs to be initialized with the `pci_initialize_hdlc` procedure before other functions in the library can be used.

Its prototype is:

```
void pci_initialize_hdlc (void);
```

## 1.3 Open HDLC Channels

### Summary

Opens an HDLC telecom channel.

### Function Prototype

PCI_HDLC_CHAN*	<code>pci_open_hdlc_chan(</code>	
	<code>int pci_id,</code>	Board number
	<code>char framer_id,</code>	Connector or framer ID
	<code>int nslots,</code>	Number of timeslots
	<code>int * slotlist,</code>	List of timeslots
	<code>uint32_t flags,</code>	Open mode flags
	<code>uint32_t framer_mode,</code>	Desired framer mode
	<code>int idleFlag)</code>	value sent on write when no data is available

### Description

This function opens and initializes an HDLC telecom channel on a DPT device.

The `pci_id` parameter gives the index of the desired board (the numeric portion of the board name).

The `framer_id` parameter is 0-3, ASCII '0'-'3', ASCII 'a'-'d', or ASCII 'A'-'D'. If `flags` contains `CHAN_NOMAP` `framer_id` is used directly. Otherwise it is interpreted as a connector ID using `pci_chan_log2phys()`. If `flags` contains `CHAN_WRITE` `framer_id` is taken to be a transmit connector. Otherwise it is taken to be a receive connector. If `framer_mode` specifies a local loopback mode either connector may be specified.

A positive `nslots` parameter specifies the number of timeslots in the channel. The `slotlist` parameter is an array of `nslots` integers specifying the timeslot numbers (starting from zero) to be included in the channel. Timeslots must be in ascending order and no timeslot may occur more than once.

If *nslots* is negative then *slotlist* is interpreted as an integer bitmap indicating the slots to be opened. The least significant bit corresponds to timeslot zero. Any bits that are set to one indicate that the corresponding timeslot should be included in the channel.

The *flags* parameter consists of one or more of: CHAN\_READ or CHAN\_WRITE to specify the direction of the channel, CHAN\_NOMAP to specify that *framer\_id* is not to be interpreted as a connector ID. The HDLC library does not support the CHAN\_NBLOCK flag used to specify that non-blocking I/O.

The *framer\_mode* parameter specifies the desired framer mode. If there are no channels open on the framer the given mode will be set. If there are channels open on the given framer the open will only succeed if the current framer mode matches *framer\_mode*.

The mode is a bit wise combination of the following flags (from `dpchan.h`):

E1_RX_BYP_JAT	Bypass receive jitter attenuator
E1_RX_UNFRAMED	Bypass receive framer
E1_RX_HDB3	Receive using HDB3 line encoding
E1_RX_CRC_MF	Receive using CRC4 multiframing
E1_RX_CAS_MF	Receive using CAS multiframing
E1_RX_CRC_REFR	Reframe on excessive CRC errors
E1_RX_NO_REFR	Do not reframe after initial sync
E1_RX_FAST_CLK	Receiver is candidate for transmit clock source
E1_TX_BYP_JAT	Bypass transmit jitter attenuator
E1_TX_HDB3	Transmit using HDB3 line encoding
E1_TX_CRC_MF	Transmit using CRC4 multiframing
E1_TX_CAS_MF	Transmit using CAS multiframing
E1_LOOP_SDRAM	Enable SDRAM loopback (processor pre-dual-port)
E1_LOOP_DPRAM	Enable DPRAM loopback (processor post-dual-port)
E1_LOOP_SERIAL	Enable serial loopback (local pre-framer)
E1_LOOP_DIG	Enable digital loopback (local post-framer)
E1_LOOP_PAYLD	Enable payload loopback (remote post-framer)
E1_LOOP_LINE	Enable line loopback (remote pre-framer)

The E1\_LOOP\_SERIAL and E1\_RX\_UNFRAMED flags affect all framers.

**Return Values**

<b>Return</b>	<b>ERRNO</b>	<b>Meaning</b>
NULL	EBUSY	One or more specified timeslots are busy or framer is in use with a different <i>framer_mode</i>
NULL	EINVAL	Invalid Parameter
NULL	ENOMEM	Insufficient Memory
NULL	ENXIO	Framer not available
NULL	ENODEV	Device not available
NULL	ETIMEOUT	Embedded processor did not respond
Other	No Change	Pointer to open channel

## 1.4 HDLC Messages

### 1.4.1 Write Messages to HDLC Channel

#### Summary

Writes a single HDLC message on a transmit channel.

#### Function Prototype

```
int pci_write_hdlc_chan (
    PCI_HDLC_CHAN chan,      Pointer to open transmit channel
    size_t nbytes,          Number of bytes to write
    uint8_t* buf)           Data to send
```

#### Description

The `pci_write_hdlc_chan` function writes one complete message to an HDLC channel. If the function is successful it returns *nbytes*. If an error occurs a `-1` is returned.

Since the function does not return until the complete message is sent up to 7 extra bits may be added to the end of a message to pad out the last byte.

#### Return Values

Return	ERRNO	Meaning
-1	EBADF	Invalid <i>chan</i>
-1	EBUSY	I/O already in progress on <i>chan</i> or channel loop source set for <i>chan</i>
-1	E_OVERFLOW	Buffer underflow occurred
-1	EINVAL	<i>chan</i> or <i>buf</i> parameters NULL, read only channel
-1	ETIMEDOUT	Timed out waiting for buffer space to become available
-1	EIO	other IO error
non-negative	No Change	Amount of data written

## 1.4.2 Read Messages from HDLC channel

### Summary

Reads a receive channel.

### Function Prototype

```
int          pci_read_hdlc_chan (
    PCI_HDLC_CHAN  chan,      Pointer to open receive channel
    size_t         nbytes,    Maximum number of bytes to read
    uint8_t *      buf)       Buffer to receive data
```

### Description

This function reads from a receive channel until a complete HDLC message is received.

If the message length is more than `nbytes` the message is truncated and `E_HDLC_LONG_MESSAGE` error is returned.

### Return Values

Return	ERRNO	Meaning
-1	EBADF	Invalid <i>chan</i>
-1	EBUSY	I/O already in progress on <i>chan</i>
-1	ENOLINK	Framer out of sync
-1	E_OVERFLOW	Buffer overflow
-1	ETIMEDOUT	Timed out waiting for DMA or for data to become available
-1	E_HDLC_LONG_MESSAGE	HDLC message longer than <code>nbytes</code>
-1	E_HDLC_PROTOCOL	Bit stream was not a valid HDLC sequence.
-1	E_HDLC_CRC	CRC error encountered
non-negative	No Change	Amount of data read

## 1.5 Close Channel

### Summary

Closes an HDLC channel and frees associated system resources.

### Function Prototype

```
int          pci_close_hdlc_chan (
            PCI_HDLC_CHAN          chan)  Pointer to open HDLC channel
```

### Description

This function closes an open HDLC channel and frees the associated system resources. This function should always be called whenever a channel is no longer needed.

### Return Values

Return	ERRNO	Meaning
-1	EINVAL	<i>chan</i> is not a valid channel
0	No Change	Success