PBX Integration Software Reference

for Linux and Windows

05-1278-006

COPYRIGHT NOTICE

All contents of this document are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation. Every effort is made to ensure the accuracy of this information. However, due to ongoing product improvements and revisions, Dialogic Corporation cannot guarantee the accuracy of this material, nor can it accept responsibility for errors or omissions. No warranties of any nature are extended by the information contained in these copyrighted materials. Use or implementation of any one of the concepts, applications, or ideas described in this document or on Web pages maintained by Dialogic-may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not condone or encourage such infringement. Dialogic makes no warranty with respect to such infringement, nor does Dialogic waive any of its own intellectual property rights which may cover systems implementing one or more of the ideas contained herein. Procurement of appropriate intellectual property rights and licenses is solely the responsibility of the system implementer. The software referred to in this document is provided under a Software License Agreement. Refer to the Software License Agreement for complete details governing the use of the software.

Publication Date: October, 2001

Part Number: 05-1278-006

Dialogic, an Intel company
1515 Route 10
Parsippany NJ 07054
USA

Technical Support
http://support.dialogic.com

For **Sales Offices** and other contact information, visit the main Dialogic website at:
http://www.dialogic.com

Table of Contents

List of Tables

List of Figures

1. How To Use This Manual

1.1. Audience

This manual is written for programmers and engineers who and are interested in using the D/42 series software (the Dialogic standard for PBX Integration series boards), together with standard D/4x voice software, to develop voice and call processing applications on PBX Integration boards for a PBX system.

When this manual addresses "you," it means "you, the programmer," and when this manual refers to the "user," it means the end-user of your application program.

If you are experienced with voice technology and Dialogic products, you may prefer to deal strictly with information found in Sections 3 and 4 in this manual. These sections contain comprehensive and detailed technical information for programming an application with C language library functions and data structures.

If you are new to Dialogic products and voice technology, you may prefer to start with the *Features Guide*. The *Features Guide*, contained in the *Voice Software Reference*, provides an introduction to the voice products, with explanations and help beyond a strictly technical level so that you can quickly learn the voice software. This includes descriptions of how to use the voice processing, signaling, and Call Progress Analysis features and how to design a multi-line voice application.

NOTE: PBX Integration boards only support CPA when used in the default routing configuration. For instance, if a voice resource of a D/82JCT-U is listening to a front end other than the default (its own), it may return a disconnected result. This is because these boards support the call progress analysis feature of **dx_dial( )**, only when a board is using the default TDM routing. In other words, PBX Integration board voice resources cannot be used to provide CPA capability for other boards.

1.2. Voice Hardware Covered by This Manual

The PBX Integration board is designed to provide a set of cost-effective tools for implementing computerized voice and call processing applications for several different private branch exchange (PBX) systems and key telephone systems (KTSs). It provides the basic voice and call processing capabilities of D/4x voice hardware and adds hardware and firmware required to integrate with PBXs and KTSs. Refer to the *Voice Software Reference* for more information on voice and call processing. For convenience, the term PBX is used to refer to any private branch exchange (PBX), key system unit (KSU), or key telephone system (KTS).

The PBX Integration hardware models covered by this manual include the following:

D/42JCT-U™ an 4-port voice-processing solution from the Dialogic PBX Integration family of products. It has downloadable firmware and a universal digital station set interface that can emulate a number of phones from different vendors. The trunk interface section of the board uses special digital PBX signaling link technology to interface with the entire range of supported PBXs. The D/82JCT-U is in the PCI form factor, and it provides SCbus and H.100 connectivity. The board uses Dialogic R4 firmware and the Voice and Unified APIs. Support for host-assisted FAX is also provided.

D/82JCT-U™ an 8-port voice-processing solution from the Dialogic PBX Integration family of products. It has downloadable firmware and a universal digital station set interface that can emulate a number of phones from different vendors. The trunk interface section of the board uses special digital PBX signaling link technology to interface with the entire range of supported PBXs. The D/82JCT-U is in the PCI form factor, and it provides SCbus and H.100 connectivity. The board uses Dialogic R4 firmware and the Voice and Unified APIs. Support for host-assisted FAX is also provided.

1.2.1. Voice Hardware Model Names

Model names for Dialogic voice boards are based upon the following pattern:

D/NNNoRBB-TT-VVV

where:

D/	identifies the board as Dialogic voice hardware
NNN	identifies the number of channels (2, 4, 8, 12, etc.), or relative size/power measure
o	0 indicates no support for Call Progress Analysis; 1 indicates support for Call Progress Analysis; and 2 indicates PBX support
R	if present, represents board revision (D, E, J, etc.)
BB	bus type (SC or CT)
TT	telephony interface type (if applicable; valid entries include LS, T1, E1, BR, U {for universal PBX Interface)
VVV	ohm value (if it applicable; valid entries are 75 and 120)

Sometimes it is necessary in this document to refer to a group of voice boards rather than specific models, in which case an “x” is used to replace the part of the model name that is generic. For example, D/xxx refers to all models of the voice hardware, and D/8x refers to all 8-channel models.

1.3. When To Use This Manual

This *PBX Integration Software Reference* contains programming information for developing applications in the Windows and Linux operating system environment using the Unified API™ and D/42 runtime library. The Unified API provides a

single, basic set of high-level calls used to develop applications across a variety of manufacturer's switches. The D/42 runtime library supports the Unified API and works in conjunction with the standard voice runtime library to enable applications to set up calls and perform PBX call functions using the PBX Integration board.

The sequence for installing software and hardware to develop application programs is as follows:

- Install the PBX Integration hardware in a PC according to the *PBX Integration Quick Install Card*.
- Install the System Release software for your system following the procedures in the *System Release Software Installation Reference* to include D/42 and voice support.
- Download the PBX Integration firmware to the boards in your system using the Dialogic Configuration Manager (DCM).

Refer to this manual, the *PBX Integration User's Guide*, and the *Voice Software Reference* to develop application programs.

1.4. Documentation Conventions

The following documentation conventions are used throughout this manual:

- When terms are first introduced, they are shown in *italic text*.
- When a word or phrase is emphasized, it is underscored.
- Data structure field names and function parameter names are shown in boldface, as in **maxsec.**
- Function names are shown in boldface with parentheses, such as **d42_display( ).**

Names of defines or equates are shown in uppercase, such as T_DTMF. File names are also shown in uppercase and italics, such as *D42DRV.EXE*.

1.5. How This Manual Is Organized

Chapter 1 – How To Use This Manual describes the *PBX Integration Software Reference*.

Chapter 2 – Using PBX Functions provides fundamental information on using the voice library functions with the PBX Integration board product.

Chapter 3 – Function Reference provides comprehensive and detailed technical information on the voice software C language voice library functions.

Chapter 4 – Programming Considerations contains programming information about developing applications for the supported PBXs

Appendix A – Unified API Quick Reference provides concise information on the voice software C language voice library functions.

Appendix B – Demonstration Programs for Windows

Appendix C – Error Definitions

Glossary contains a comprehensive list of definitions for commonly used terms.

Index contains an alphabetical index of features and topics.

2. Using the PBX Functions

The PBX circuitry on the PBX Integration boards provides functions specific to several different PBXs. These functions are implemented using the D/42 runtime library (.dll). The D/42 runtime library is used in addition to the standard Dialogic voice runtime library when tight integration and control of the PBX and D/42-xx and PBX Integration boards are required.

The standard voice runtime library acts as an interface between the application program and the PBX Integration board hardware. The voice runtime library is used to access standard voice functions such as voice play/record and call progress analysis. Refer to the *Voice Software Reference* for detailed explanations on using voice functions.

2.1. The Unified API

The Unified API (Application Programming Interface) enables the development of applications across a variety of manufacturers switches (both Key and PBX systems) through a single interface. The Unified API provides a single set of basic functions (refer to *Chapter 3*) that can be used for any supported switch and are sent directly to the switch through the PBX Integration board, without additional hardware. Functioning as an extension to The Dialogic standard voice API, the Unified API offers a single design model that allows developers to take advantage of advanced PBX features (such as called/calling number ID and ASCII display information).

Using the Unified API shortens development time by eliminating the need to learn separate APIs for each switch. It enables you to create applications with a common set of functions, which operate with switches produced by different manufacturers, thereby widening your product's support beyond the traditional single-switch focus.

Utility functions included in the Unified API allow programmers to control the PBX Integration board. The application can retrieve the channel type, obtain and set channel parameters, retrieve firmware/driver/library version numbers, and retrieve error information.

The D/42 runtime library works in conjunction with the standard voice runtime library to enable applications to set up calls and perform PBX call functions using PBX Integration boards. In addition, the D/42 runtime library supports the Unified API.

The functions called by the Unified API are synchronous. This means that when a function is called in a thread, it is performed immediately and blocks until the operation is complete. Functions can be called at any time to execute on a channel that is idle or busy, and do not affect the idle or busy state of the channel.

NOTE: Synchronous is a term used in the Windows-Dialogic environment. Refer to the *Voice Software Reference* for a detailed explanation of synchronous functions.

The D/42 runtime library treats boards and channels as separate devices, even though channels are physically part of a board. A channel device is an individual PBX line connection, and a board device is a PBX Integration board that contains channels. Most functions are performed at the channel level, such as getting called/calling number ID. Certain functions, such as setting board parameters, can occur at the board level and effect all channels on that board.

NOTE: Since boards and channels are considered separate devices under Windows, it is possible to open and use a channel without opening the board where the channel is located. There is no board-channel hierarchy imposed by the D/42 runtime library.

2.2. Switch-Specific Support

PBX station set phones come with both standard and programmable keys that give access to switch-specific functions. The most common of these features include:

- Transfer
- Hold
- Trunk line select
- Message waiting indication
- Hands-free operation

Refer to the *PBX Integration User's Guide* for detailed information about PBX features. Because the PBX Integration board has the capability to emulate a PBX station set, it can also emulate any standard or programmable function for your application. Applications can take advantage of the most common features listed here, as well as less frequently used features like conference. In addition, your application can reprogram keys as needed. Refer to *Chapter 4* for details about switch-specific programming.

3. Unified API Function Reference

This chapter provides comprehensive and detailed technical information on the PBX interface software, C-language library functions (the Unified API). The library functions are prototyped in *D42LIB.H*.

See the Table of Contents for a list of functions. *Appendix A* provides a Quick Reference containing a compact list of functions that are detailed in this chapter. Only functions compatible with the PBX Integration board are discussed in this document.

Each function is listed in alphabetical order and provides the following information:

Function Header	Located at the beginning of each function and contains the following information: function name, function title, function syntax, input parameters, output or returns, includes (header files required to be include), and mode. The function syntax and inputs include the data type and are shown using standard C language syntax.
Description	Provides a detailed description of the function operation, including parameter descriptions.
Example	Provides one or more C language coding examples showing how the function can be used.
Cautions	Provides warnings and reminders.

Name:	int ATD4_BDTYPE(devh)	
Inputs:	int devh	• board descriptor
Returns:	board type	• returns board type information (see below)
	0	• if success
	-1	• if error; See Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **ATD4_BDTYPE( )** function returns the board type of the queried device.

Board Type	Description
TYP_D/82L4	Lucent Definity 75/85
TYP_D/82L2	Lucent Definity G3
TYP_D/82SR	Siemens ROLM Series
TYP_D/82SH	Siemens Hicom
TYP_D/82SX	MITEL SX Series
TYP_D/82NS	Nortel Norstar
TYP_D/82M1	Nortel Meridian 1

Parameter	Description
devh:	specifies the valid board device descriptor obtained by a call to **dx_open( )**

■ Cautions

None.

■ Example

```
void main(void)
{
          int          devh;
          int          rc = 0;

/* Open Board Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
      printf("Error dx_open()\n");
      exit(-1);
          } /* End dx_open */

/* Check Board Type */
if ( (rc = ATD4_BDTYPE(devh)) == -1)
         {
printf("Error ATD4_BDTYPE()\n");
dx_close(devh);
exit(-1);
         }

printf("Board Type = %d\n",rc);
dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

EDX_TIMEOUT	Firmware does not respond within a specified time
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_UNKNOWNBOARD	Unknown D/42 board type

■ See Also

- **ATD4_CHTYPE( )**

Name:	int ATD4_CHTYPE(devh)	
Inputs:	int devh	• channel descriptor
Returns:	channel type	• returned channel type information (see below)
	0	• if success
	-1	• if error; see Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **ATD4_CHTYPE( )** function returns the channel type of the queried device.

Channel Type	Description
TYP_D/82L4	Lucent Definity 75/85
TYP_D/82L2	Lucent Definity G3
TYP_D/82SR	Siemens ROLM Series
TYP_D/82SH	Siemens Hicom
TYP_D/82SX	MITEL SX Series
TYP_D/82NS	Nortel Norstar
TYP_D/82M1	Nortel Meridian 1

Parameter	Description
devh:	specifies the valid channel device descriptor obtained by a call to **dx_open( )**

■ Cautions

None.

Example

```
void main(void)
{
       int          devh;
       int          rc = 0;

/* Open Channel Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
     printf("Error dx_open()\n");
     exit(-1);
          } /* End dx_open */

/* Check Channel Type */
if ( (rc = ATD4_CHTYPE(devh))== -1)
         {
printf("Error ATD4_CHTYPE()\n");
dx_close(devh);
exit(-1);
         }

printf("Channel Type = %d\n",rc);
dx_close(devh);
}     /* End main */
```

Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

EDX_TIMEOUT	Firmware does not respond within a specified time
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_UNKNOWNBOARD	Unknown D/42-xx or PBX Integration board type

See Also

- **ATD4_BDTYPE( )**

Name:	int d42_brdstatus(devh, buffstatus, bufferp)	
Inputs:	int devh	• board descriptor
	char *buffstatus	• pointer to buffer containing board status information
	char *bufferp	• reserved for future use
Returns:	0	• if success
	-1	• if error; see Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_brdstatus( )** function retrieves the current board status and places it in an application buffer. The board status is a bit mask representing the status of the board (see below) on a per board basis. Each D/82JCT-U contains two virtual boards of four channels each, for a total of eight channels. Each D/42JCT-U contains one virtual board of four channels. The application buffer (buffstatus) that will contain the board status information must be one byte.

Bit	7	6	5	4	3	2	1	0
Channel	x	x	x	x	4	3	2	1
Example*	0	0	0	0	1	1	1	1

* Data shows that all channels on the board have communication.

bit0	first channel on board	1=OK, 0=no communication
bit1	second channel on board	1=OK, 0=no communication
bit2	third channel on board	1=OK, 0=no communication
bit3	fourth channel on board	1=OK, 0=no communication

Parameter	Description
devh:	specifies the valid board device descriptor obtained by a call to **dx_open( )**
buffstatus:	pointer to the 1-byte application buffer where the board status is placed
bufferp:	pointer to an additional application buffer (reserved for future use)

■ Cautions

The character pointer **bufferp** is required. The associated buffer must be 49 bytes.

■ Example

```
void main(void)
{
     int          devh;
     int          rc = 0;
     char          buffstatus;
     char          bufferp[49];

/* Open Channel Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
printf("Error dx_open()\n");
exit(-1);
          } /* End dx_open */

/* Get the board status Information */
if ( (rc = d42_brdstatus(devh, &buffstatus, bufferp)) == -1)
          {
printf("Error d42_brdstatus()\n");
dx_close(devh);
exit(-1);
          } /* End d42_brdstatus*/

printf("Board Status = %X\n",buffstatus);
dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

See Also

- **d42_chnstatus( )**

Name:	int d42_chnstatus(devh, statusp, bufferp)	
Inputs:	int devh	• channel descriptor
	char *statusp	• pointer to buffer containing channel status information
	char *bufferp	• reserved for future use
Returns:	0	• if success
	-1	• if error; see Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_chnstatus( )** function retrieves the current channel status and places it in an application buffer. The application buffer (statusp) that will contain the channel status information must be one byte. The channel status is a single bit (bit 0) representing the status of the channel device.

Parameter	Description
devh:	specifies the valid channel device descriptor obtained by a call to **dx_open( )**
statusp:	pointer to a 1-byte application buffer. The application buffer will contain a non-zero value if channel is communicating with the switch. non-zero = OK 0 = no communications
bufferp:	pointer to an additional application buffer (reserved for future use)

■ Cautions

The character pointer **bufferp** is required. The associated buffer must be 49 bytes.

■ Example

```
void main(void)
{
     int          devh;
     int          rc = 0;
      char         bufferp[49];
     char         status;

/* Open Channel Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
printf("Error dx_open()\n");
exit(-1);
          } /* End dx_open */

/* Get the channel status Information */
if ( (rc = d42_chnstatus(devh, &statusp, bufferp)) == -1)
          {
printf("Error d42_chnstatus():\n");
dx_close(devh);
exit(-1);
          } /* End d42_chnstatus*/

if (status)
          {
printf("Channel Communication OK\n");
          }
else
          {
printf("No Channel Communication\n");
          }

dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

■ See Also

- **d42_brdstatus( )**

Name: int d42_closefeaturesession(devh)
Inputs: int devh — • channel device
Returns: 0 — • if success
-1 — • if error; see Errors below
Includes: D42LIB.H
Mode: immediate

■ Description

The **d42_closefeaturesession( )** function closes a feature session on a specified channel. Once the feature session is closed the special functions that require a feature session to be open may not be used, for example, **d42_writetodisplay( )**.

Parameter	Description
channel	specifies the channel number.

■ Cautions

This function is valid only with the Nortel Norstar PBX.

This function sets the parameter values for the channel parameters D4CH_SOFTKEYINPUT and D4CH_TERMINATEFEATURE to 0 for disabled.

■ Example

```
void main(void)
     {
     int              devh;
     int              rc = 0;
char       szDnNumber = “221”;
     int              iTerminalType;
     int              iEvtMask =  D42_EVT_SOFTKEY    | D42_EVT_ASYNCCLOSEFEATSESSION


     /* Open Channel Device */
     if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
               {
               printf("Error dx_open()\n");
               exit(-1);
```

```
            } /* End dx_open */

/* Open a feature session */
if ( (rc = d42_openfeaturesession (devh, szDnNumber,  &iTerminalType, iEvtMask )) == -1)
      {
      printf("Error d42_closefeaturesession():\n");
      dx_close(devh);
      exit(-1);
      } /* End d42_brdstatus*/

/*something is done */

/* close the feature session */
if ( (rc = d42_closefeaturesession (devh)) == -1)
      {
      printf("Error d42_closefeaturesession():\n");
      dx_close(devh);
      exit(-1);
      } /* End d42_brdstatus*/

      dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDDEVICE	Invalid or wrong device handle sent to the function
ED42_NOFEATURESESSION	No feature session has been opened on the channel.
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

■ See Also

- **d42_openfeaturesession( )**
- **d42_writetodisplay( )**

Name:	int d42_display(devh, bufferp)	
Inputs:	int devh	• channel descriptor
	char *bufferp	• pointer to an application buffer. The buffer will contain display data for the selected channel.
Returns:	0	• if success
	-1	• if error; see Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_display( )** function retrieves the current LCD/LED display (alphanumeric) data and places it in an application buffer. The application buffer must be 49 bytes, and will hold an entire data string up to 48 bytes (see below) plus a null. The length of the data string is 32 or 48 bytes for the supported PBXs. Byte 0 of the display data corresponds to the top, left-most display element. The display data is stored as a null-terminated ASCII string. Refer to the *PBX Integration User's Guide* for more information specific to your PBX. Examples showing the contents of the application buffer for each supported switch with a display less than or equal to 48 bytes are shown below:

■ Siemens Hicom - 48-digit display

	[illegible]
data	4E 6F 65 6C 20 4D 63 4C 6F 75 67 68 6C 69 6E 20
byte	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

	[illegible]
data	20 20 20 20 20 20 20 20 43 6F 6E 73 75 6C 74 61
byte	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

	[illegible]
data	74 69 6F 6E 3F 20 20 20 20 20 20 20 20 20 20

	20
byte	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

■ MITEL SUPERSET 420 - 32-character display

	[illegible]
data	43 41 4C 4C 46 4F 52 44 57 41 52 49 4E 47 3F 20
byte	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

	[illegible]
data	59 65 73 20 20 20 20 20 20 20 20 20 20 4E 6F 20
byte	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

data	xx *xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx*
byte	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

■ Nortel Norstar - 32-character display

	[illegible]
data	54 72 61 6E 73 66 65 72 20 20 20 20 20 20 20 20
byte	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

data	xx *xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx*
byte	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

■ Nortel Meridian 1 - 48-character display

	a — 1 ○ ↙ 5
data	61 32 01 00 04 05 20 20 20 20 20 20 20 20 20 20
byte	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

Parameter	Description
devh:	specifies the valid channel device descriptor obtained by a call to **dx_open( )**
bufferp:	pointer to the application buffer. The buffer will contain the display data in ASCII format.

■ Cautions

The application buffer must be 49 bytes. The length of the LCD display data is 48 bytes for the supported PBX listed above. All other supported PBXs have longer-length LCD display data, so **d42_displayex( )** must be used. The data is stored as a null-terminated ASCII string. An application that passes anything smaller will not be backward compatible.

If you execute a function that updates the display (e.g., set the message waiting indicator, or show the calling number ID), ensure that you allow time for the switch to update the display before using **d42_display( )**, or you can call the **d42_display( )** function until valid display data is returned.

■ Example

```
void main(void)
{
     int          devh;
     int          rc = 0;
     char         bufferp[49];

/* Open Channel Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
printf("Error dx_open()\n");
exit(-1);
} /* End dx_open */

/* Wait for incoming call */
if ( (rc = dx_wtring(devh, 2, DX_ONHOOK, -1))==-1)
          {
printf("Error dx_wtring()\n");
dx_close(devh);
exit(-1);
          }

/* Get the Display Information */
if ( (rc = d42_display(devh, bufferp)) == -1)
          {
printf("Error d42_display()\n");
dx_close(devh);
exit(-1);
           } /* End d42_display */

printf("Display = %s\n",bufferp);
dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle

ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

See Also

- **d42_displayex( )**

Name:	int d42_displayex(devh, bufferp)	
Inputs:	int devh	• channel descriptor
Returns:	char *bufferp	• pointer to an application buffer. The buffer will contain display data for the selected channel.
	buflen	• length of buffer
Includes:	0	• if success
	-1	• if error; see Errors below.
Category:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_displayex( )** function retrieves the current LCD/LED display (alphanumeric) data and places it in an application buffer. Unlike **d42_display( )**, this function can retrieve display data larger than 49 bytes. The buffer must be at least 49 bytes, which would mean a data string of 48 bytes plus a null. The length of the data string is 50 for the Lucent Definity G3 and the 75/85 PBXs 60 for the Siemens ROLM; and 80 for the MITEL SX-200ML or SX-2000 PBXs. Byte 0 of the display data corresponds to the top, left-most display element. The display data is stored as a null-terminated ASCII string. Refer to *the PBX Integration User's Guide* for more information specific to your PBX. An example showing the contents of the application buffer for each of the two supported switches with a display larger than 48 bytes is shown below. **d42_displayex( )** may also be used for display sizes smaller than 48 bytes.

■ Lucent Definity - 50-character display

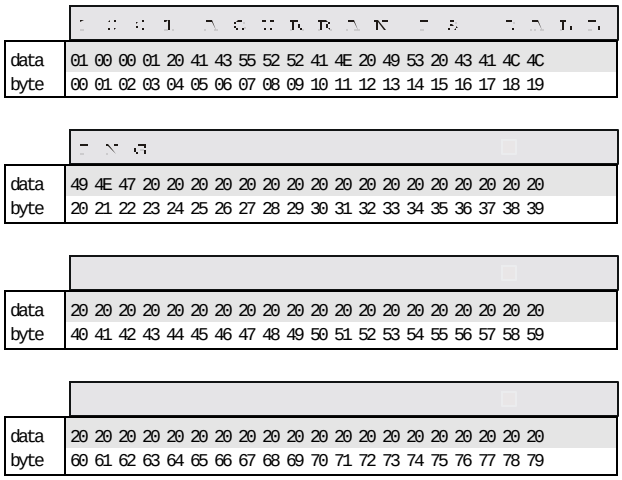
data	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

data	52	45	4C	45	41	53	45	20	41	4E	44	20	54	52	59	20	41	47	41	49	4E	20	20	20	20
byte	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49

■ Siemens ROLM - 60-character display

data	43 4F 4E 46 45 52 45 4E 43 45 20 20 20 20 20 20 01 02 03 20
byte	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
data	20 20 20 20 20 20 20 20 20 20 59 4F 55 52 20 50 4F 53 49 54
byte	20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
data	49 4F 4E 3A 20 20 01 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59

■ MITEL SUPERSET 430 - 80-character display



data	01 00 00 01 20 41 43 55 52 52 41 4E 20 49 53 20 43 41 4C 4C
byte	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
data	49 4E 47 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
byte	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79

Parameter	Description
devh:	specifies the valid channel device descriptor obtained by a call to **dx_open( )**
bufferp:	pointer to the application buffer. The buffer will contain the display data in ASCII format.
buflen:	length of buffer on entry.

■ Cautions

The pointer to the application buffer is assumed to be large enough to hold the entire string plus a null, and the total must be at least 49 bytes.

If you execute a function that updates the display (e.g., set the message waiting indicator, or show the calling number ID), ensure that you allow time for the switch to update the display before using **d42_displayex( )**, or you can call the **d42_displayex( )** function until valid display data is returned.

■ Example

```
void main(void)
{
     int          devh;
     int          buflen = 50;
     int          rc = 0;
     char         bufferp[50];

/* Open Channel Device */
if ( (devh = dx_open("dxxxB1C1",NULL))==-1)
          {
printf("Error dx_open()\n");
exit(-1);
          } /* End dx_open */

/* Wait for incoming call */
if ( (rc = dx_wtring(devh, 2, DX_ONHOOK, -1))==-1)
          {
printf("Error dx_wtring()\n");
dx_close(devh);
exit(-1);
          }

/* Get the Display Information */
if ( (rc = d42_displayex(devh, bufferp, buflen)) == -1)
          {
printf("Error d42_displayex()\n");
dx_close(devh);
exit(-1);
          } /* End d42_displayex */

printf("Display = %s\n",bufferp);
dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

ED42_MEMORY	Buffer not large enough

See Also

- **d42_display( )**

Name:	int d42_getnewmessage(channel, bufferp)	
Inputs:	unsigned int channel	• channel number
	unsigned char *bufferp	• pointer to buffer containing message data
Returns:	0	• if success
	-1	• if error; see Errors below
Includes:	D42LIB.H	
Mode:	immediate	

■ Description

The **d42_getnewmessage( )** function allows messages to be returned to a board from a Norstar PBX. The function retrieves the next message for the specified channel and places it in the user buffer. This feature has to be turned on by setting the parameter **D4CH_MESG_Q** with the **dx_setparm( )** function

Parameter	Description
channel	specifies the channel number.
bufferp	points to the buffer where messages are placed

■ Cautions

This function is valid only with the Nortel Norstar PBX.

The pointer to the user buffer is assumed to be large enough to hold the entire string plus a NULL, which is a total of 49 characters. The associated buffer must be 49 bytes. An application which passes anything smaller will not be backward compatible.

■ Example

```
int  rc =0;
unsigned char buffer[49];
unsigned int channel = 1;

/* Get new message */
if (  (rc = d42_getnewmessage(channel, &buffer))
```

```
== ERR_SUCC)
  {
    printf("d42_getnewmessage() == %d %s, channel
    = %d, Message = %s\n",  channel, buffer);
    }
  else
    {
    printf("d42_getnewmessage() == %d %s\n", rc,
    d42_geterror(rc));
    }
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function
ERR_NOBOARD	No board present
ERR_NODBFW	No firmware loaded
ERR_BADCH	Invalid channel number
ERR_NULLPTR	Null pointer passed to function
ERR_QEMPTY	Message queue is empty
ERR_QOVRFLOW	Message queue is full

The final two messages listed are returned when the host computer PBX message queue is full or empty, respectively. This queue is 8K, so up to 96 messages may be stored before the overflow state occurs. When the queue is full, incoming messages are lost until the application clears the queue.

■ See Also

- **d42_closefeaturesession( )**
- **d42_openfeaturesession( )**
- **d42_writetodisplay( )**

Name:	int d42_getparm(devh, parmnum, parmvalp)	
Inputs:	int devh	• board or channel descriptor
	int parmnum	• parameter name
	void *parmvalp	• pointer to parameter value
Returns:	0	• if success
	-1	• if error; see Errors below
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_getparm( )** function retrieves the selected channel or board parameter and places it in the application buffer (parmvalp). Depending on the parameter retrieved, the data returned can be either a character string or an integer. The board and channel parameter that can be retrieved are listed in *Table 1*.

Parameter	Description
devh:	specifies the valid board device or channel device descriptor obtained by a call to **dx_open( )**
parmnum:	contains the parameter name to retrieve
parmvalp:	pointer to the application variable that will receive the parameter value

■ Cautions

When retrieving a parameter, the application passes a pointer to a variable that will contain the actual parameter value. This variable should be treated as an unsigned integer for all parameters. The application should cast the parmvalp parameter to a (void *) to avoid compiler warnings.

Table 1. Board and Channel Parameters for d42_getparm( )

Board Parameters	Description
D4BD_CALLID	Enable Caller ID Values: 0 - disable (default) 1 - enable
D4BD_GETSWITCHTTYPE	Obtains the switch type Values: PBX_L4 - Lucent 75/85 PBX_L2 - Lucent G3 PBX_SH - Siemens Hicom PBX_SR - Siemens ROLM PBX_NS - Norstar PBX_M1 - Meridian 1 PBX_SX - MITEL SX-50 PBX_SX2 - MITEL SX-200ML or SX-2000
D4BD_REPORT_RESET	Enable report reset Values: 0 - disable (default) 1 - enable

Channel Parameters	Description
D4CH_CHANNELSTATUS	Receive asynchronous channel status messages Values: 0 - disable (default) 1 - enable
D4CH_LC_LAMP	Lamp to monitor for loop current
D4CH_CHANNELUPDATE	Enable/Disable asynchronous LCD and indicator updates
D4CH_CALLERIDAVAILABLE	Enables notification of Caller ID availability using the T_CALLERIDAVAILABLE event. Values: 0 - disable (default)

	1 - enable
D4CH_CHANNELSTATUS	Enables notification of a change in the status of the channel. Values: 0 - disable (default) 1 - enable
D4CH_SOFTKEYINPUT*	Enables notification of SoftKey input using the T_SOFTKEYINPUT event. Values: 0 - disable (default) 1 - enable
D4CH_TERMINATEFEATURE*	Enables notification when a feature session is terminated. Values: 0 - disable(default) 1 - enable

* When d42_openfeaturesession( ) is called for a channel, the value of this parameter is set automatically to 1 (enable) for that channel. When d42_closefeaturesession( ) is called, the value of this parameter is set automatically to 0 (disable) for that channel.

■ Example

```
void main(void)
{
    int          devh;
    int          rc = 0;
    int          parmnum;
    unsigned int  parmvalp;

/* Open Board Device */
if ( (devh = dx_open("dxxxB1",NULL))==-1)
         {
printf("Error dx_open()\n");
exit(-1);
         } /* End dx_open */

if ( (ATD4_BDTYPE (devh)) == TYP_D/82M1)
         {
/* Get the Board Parameter To See if Speakerphone Mode is Enabled */
if ( (rc = d42_getparm(devh, D4BD_SPMODE, (void *)&parmvalp)) == -1)
         {
printf("Error d42_getparm(D4BD_SPMODE)\n");
dx_close(devh);
exit(-1);
         } /* End d42_getparm */

/* Check if Speakerphone is enabled */
if (parmvalp == 1)
```

```
        {
printf("Speakerphone Mode is ENABLED");
else if (parmvalp == 0)
printf("Speakerphone Mode is DISABLED");
}  /* End Check if Speakerphone is enabled */

        } /* end ATD4_BDTYPE */
dx_close(devh);
}  /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_INVALARG	Invalid argument passed to function

■ See Also

- **d42_setparm( )**

Name:	int d42_getver(devh, bufferp, flag)	
Inputs:	int devh	• board descriptor
	char *bufferp	• pointer to an application buffer containing the version information
	int flag	• determines if firmware or library version is retrieved
Returns:	0	• if success
	-1	• if error; see Errors below.
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_getver( )** function retrieves the board firmware or library version and places it in an application buffer. The application buffer is at least 100 bytes long and will contain either the firmware or library version number in the following format:

Firmware `Firmware Version: XX.XX type YY.YY`

where: **X.XX** is the version number
type is the type of release (Alpha, Beta, Experimental, or Production)
Y.YY is the alpha or experimental number

Library `File Version: YY.MM.XX.XX  Product Version: YY.MM.XX.XX`

where: **YY** is the year
MM is the month
X is a number

Parameter	Description
devh:	specifies the valid board device descriptor obtained by a call to **dx_open( )**
bufferp:	pointer to the application buffer that will contain the version data
flag:	determines if the firmware or library version number is placed in the application buffer. VER_D42FIRMWARE - returns the D/42-xx or PBX Integration board firmware version VER_D42LIB - returns the D42 library (D42LIB) version

■ Cautions

The application buffer must be at least 100 bytes.

■ Example

```
void main(void)
{
     int          devh;
     int          rc = 0;
     char         bufferp[100];

/* Open Board Device */
if ( (devh = dx_open("dxxxB1",NULL))==-1)
          {
printf("Error dx_open()\n");
exit(-1);
          } /* End dx_open */

/* Get the Firmware Version */
if ( (rc = d42_getver(devh, bufferp, VER_D42FIRMWARE)) == -1)
          {
printf("Error d42_getver()\n");
dx_close(devh);
exit(-1);
          } /* End d42_getver */

/* Print the Firmware Version /*
printf('%s,bufferp);

dx_close(devh);
}     /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in dx_errno. For a complete list of error codes and definitions, refer to *Appendix C*.

ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_SYSTEM	System level error
ED42_RDFWVER	Error reading firmware version
ED42_INVALARG	Invalid argument passed to function

Name: int d42_gtcallid(devh, bufferp)

Inputs:	int devh	• channel descriptor
	char *bufferp	• pointer to an application buffer containing called/calling number ID data
Returns:	0	• if success
	-1	• if error; see Errors below
Includes:	D42LIB.H	
Mode:	synchronous	

■ Description

The **d42_gtcallid( )** function retrieves the called/calling number ID of the incoming call and places it in an application buffer. The application buffer must be 49 bytes, and will hold the entire data string (see below) plus a null. The length of the data string is variable. Refer to the *PBX Integration User's Guide* for more information specific to your PBX. An example showing the contents of the application buffer for any supported switch is as follows:

text	*bb*	2	2	1	_	2	2	4															
data	20	32	32	31	5F	32	32	34	00	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*
byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
text																							
data	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*	*xx*
byte	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46

Parameter	Description
devh:	specifies the valid channel device descriptor obtained by a call to **dx_open( )**
bufferp:	pointer to the application buffer. The called/calling number ID is placed here