PBX Integration Software Reference
for Linux and Windows

Copyright © 2001 Dialogic Corporation

05-1278-006
# Table of Contents

1. **How To Use This Manual** ........................................................................................................ 1
   1.1. Audience .............................................................................................................................. 1
   1.2. Voice Hardware Covered by This Manual ........................................................................ 2
       1.2.1. Voice Hardware Model Names ...................................................................................... 3
   1.3. When To Use This Manual .................................................................................................. 3
   1.4. Documentation Conventions .............................................................................................. 4
   1.5. How This Manual Is Organized .......................................................................................... 5

2. **Using the PBX Functions** ...................................................................................................... 7
   2.1. The Unified API .................................................................................................................. 7
   2.2. Switch-Specific Support ..................................................................................................... 9

3. **Unified API Function Reference** ........................................................................................ 11
   ATD4_BDTYPE() - returns the board type ............................................................................... 13
   ATD4_CHTYPE() - returns the channel type ............................................................................. 15
   d42_brdstatus() - retrieves the current board status ............................................................... 17
   d42_chnstatus() - retrieves the current channel status ............................................................ 20
   d42_closefeaturesession() - closes a feature session ............................................................. 23
   d42_display() - retrieves the current LCD/LED display ......................................................... 25
   d42_displayex() - retrieves the current LCD/LED display ..................................................... 30
   d42_getnewmessage() - allows messages to be returned to a board ........................................ 35
   d42_getparm() - retrieves the selected channel or board parameter ........................................ 37
   d42_getver() - retrieves the board firmware or library version .............................................. 42
   d42_gtcallid() - retrieves the called/calling number ID ......................................................... 45
   d42_indicators() - retrieves the current LCD or LED indicators ............................................ 48
   d42_openfeaturesession() - opens a feature session ............................................................... 70
   d42_setparm() - sets a board or channel parameter ............................................................... 73
   d42_writetodisplay() - places information on a phone set’s display ..................................... 76

4. **Programming Considerations** ............................................................................................. 79
   4.1. Opening a Channel on the PBX Integration Board ............................................................ 79
   4.2. Accessing PBX Features Using Dial Strings ..................................................................... 81
       4.2.1. On-Hook and Off-Hook Dialing Note ......................................................................... 82
       4.2.2. Turn On the Message Waiting Indicator ...................................................................... 82
       4.2.3. Turn Off the Message Waiting Indicator Dial String ................................................. 85
       4.2.4. Dial Programmable Keys ............................................................................................ 88
       4.2.5. Transferring a Call ....................................................................................................... 118
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Board and Channel Parameters for d42_getparm()</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>Board and Channel Parameters for d42_setparm()</td>
<td>74</td>
</tr>
<tr>
<td>3</td>
<td>Lucent 7434 (4-wire) Direct Key Dialing Sequences</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Lucent 8434DX (2-wire) Direct Key Dialing Sequences</td>
<td>93</td>
</tr>
<tr>
<td>5</td>
<td>Siemens ROLMphone 400 Direct Key Dialing Sequences</td>
<td>96</td>
</tr>
<tr>
<td>6</td>
<td>Siemens Hicom Optiset E Direct Key Dialing Sequences with Hicom 150E</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>Siemens Hicom Optiset E Direct Key Dialing Sequences with Hicom 300E</td>
<td>104</td>
</tr>
<tr>
<td>8</td>
<td>MITEL SX SUPERSET 420 Direct Key Dialing Sequences</td>
<td>107</td>
</tr>
<tr>
<td>9</td>
<td>MITEL SX SUPERSET 430 Direct Key Dialing Sequences</td>
<td>109</td>
</tr>
<tr>
<td>10</td>
<td>Nortel Norstar M7324 Direct Key Dialing Sequences</td>
<td>113</td>
</tr>
<tr>
<td>11</td>
<td>Nortel Meridian 1 M2616 Direct Key Dialing Sequences</td>
<td>116</td>
</tr>
<tr>
<td>12</td>
<td>Setting In-band and Out-of-band Signaling</td>
<td>120</td>
</tr>
<tr>
<td>13</td>
<td>Demo Indicator Definitions</td>
<td>138</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Lucent 7434 (4-wire) Telephone Indicators</td>
<td>51</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Lucent 8434 (2-wire) Telephone Indicators</td>
<td>52</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>ROLMphone 400 Telephone Indicators</td>
<td>55</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Siemens Optiset E Telephone Indicators</td>
<td>56</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>MITEL SUPERSET 400 Series Telephone Indicators</td>
<td>58</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Nortel Model 7324 Telephone Indicators</td>
<td>60</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>Nortel Model 2616 Telephone Indicators</td>
<td>62</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>Nortel Model 7324 Telephone Indicators</td>
<td>67</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>Lucent 7434 (4-wire) Telephone</td>
<td>89</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Lucent 8434 (2-wire) Telephone</td>
<td>92</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>ROLMphone 400 Telephone</td>
<td>95</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>Siemens Optiset E Telephone with a Hicom 150E</td>
<td>100</td>
</tr>
<tr>
<td>Figure 13.</td>
<td>Siemens Optiset E Telephone with a Hicom 300E</td>
<td>103</td>
</tr>
<tr>
<td>Figure 14.</td>
<td>MITEL SUPERSET 420 Telephone</td>
<td>106</td>
</tr>
<tr>
<td>Figure 15.</td>
<td>MITEL SUPERSET 430 Telephone</td>
<td>109</td>
</tr>
<tr>
<td>Figure 16.</td>
<td>Nortel M7324 Telephone</td>
<td>112</td>
</tr>
<tr>
<td>Figure 17.</td>
<td>Nortel Meridian 1 M2616 Telephone</td>
<td>115</td>
</tr>
<tr>
<td>Figure 18.</td>
<td>Dialogic D42 Demo Window</td>
<td>131</td>
</tr>
<tr>
<td>Figure 19.</td>
<td>D42 Options Window</td>
<td>131</td>
</tr>
<tr>
<td>Figure 20.</td>
<td>Input Strings</td>
<td>133</td>
</tr>
<tr>
<td>Figure 21.</td>
<td>Input Strings Warning</td>
<td>134</td>
</tr>
<tr>
<td>Figure 22.</td>
<td>Select Your D/42 Channel</td>
<td>134</td>
</tr>
<tr>
<td>Figure 23.</td>
<td>Select a D/42 Channel</td>
<td>135</td>
</tr>
<tr>
<td>Figure 24.</td>
<td>ROLMphone Window on the D42 Demo</td>
<td>136</td>
</tr>
<tr>
<td>Figure 25.</td>
<td>Siemens Optiset MWI Demo Window</td>
<td>140</td>
</tr>
<tr>
<td>Figure 26.</td>
<td>Enter Channel Number</td>
<td>140</td>
</tr>
<tr>
<td>Figure 27.</td>
<td>Channel Opened Message</td>
<td>141</td>
</tr>
<tr>
<td>Figure 28.</td>
<td>Send Message</td>
<td>141</td>
</tr>
<tr>
<td>Figure 29.</td>
<td>Message Sent</td>
<td>142</td>
</tr>
<tr>
<td>Figure 30.</td>
<td>Delete Message</td>
<td>142</td>
</tr>
<tr>
<td>Figure 31.</td>
<td>Message Deleted</td>
<td>142</td>
</tr>
</tbody>
</table>
PBX Integration Software Reference
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. How To Use This Manual

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
PBX Integration Software Reference

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. How To Use This Manual

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.
PBX Integration Software Reference
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.
**PBX Integration Software Reference**

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.
PBX Integration Software Reference
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.
PBX Integration Software Reference
**ATD4_BDTYPE()**

returns the board type

**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.

<table>
<thead>
<tr>
<th>Board Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP_D/82L4</td>
<td>Lucent Definity 75/85</td>
</tr>
<tr>
<td>TYP_D/82L2</td>
<td>Lucent Definity G3</td>
</tr>
<tr>
<td>TYP_D/82SR</td>
<td>Siemens ROLM Series</td>
</tr>
<tr>
<td>TYP_D/82SH</td>
<td>Siemens Hicom</td>
</tr>
<tr>
<td>TYP_D/82SX</td>
<td>MITEL SX Series</td>
</tr>
<tr>
<td>TYP_D/82NS</td>
<td>Nortel Norstar</td>
</tr>
<tr>
<td>TYP_D/82M1</td>
<td>Nortel Meridian 1</td>
</tr>
</tbody>
</table>

**Parameter**

**devh:**

specifies the valid board device descriptor obtained by a call to `dx_open()`

---

**Cautions**

None.
**Example**

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

### Channel Type Description

<table>
<thead>
<tr>
<th>Channel Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP_D/82L4</td>
<td>Lucent Definity 75/85</td>
</tr>
<tr>
<td>TYP_D/82L2</td>
<td>Lucent Definity G3</td>
</tr>
<tr>
<td>TYP_D/82SR</td>
<td>Siemens ROLM Series</td>
</tr>
<tr>
<td>TYP_D/82SH</td>
<td>Siemens Hicom</td>
</tr>
<tr>
<td>TYP_D/82SX</td>
<td>MITEL SX Series</td>
</tr>
<tr>
<td>TYP_D/82NS</td>
<td>Nortel Norstar</td>
</tr>
<tr>
<td>TYP_D/82M1</td>
<td>Nortel Meridian 1</td>
</tr>
</tbody>
</table>

### Parameter Description

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

### Cautions

None.
**ATD4_CHTYPE( )**  
*returns the channel type*

### Example

```c
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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Channel</th>
<th>Example*</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devh:</td>
<td>specifies the valid board device descriptor obtained by a call to <code>dx_open()</code></td>
</tr>
<tr>
<td>buffstatus:</td>
<td>pointer to the 1-byte application buffer where the board status is placed</td>
</tr>
<tr>
<td>bufferp:</td>
<td>pointer to an additional application buffer (reserved for future use)</td>
</tr>
</tbody>
</table>
**d42_brdstatus( )**  
*retreives the current board status*

## Cautions

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

## Example

```c
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()\r\n");
        exit(-1);
    } /* End dx_open */

    /* Get the board status Information */
    if ( (rc = d42_brdstatus(devh, &buffstatus, bufferp)) == -1)
    {
        printf("Error d42_brdstatus()\r\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
retrieves the current board status

See Also

- d42_chnstatus()


**d42_chnstatus()** retrieves the current channel status

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devh:</td>
<td>specifies the valid channel device descriptor obtained by a call to <code>dx_open()</code></td>
</tr>
<tr>
<td>statusp:</td>
<td>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</td>
</tr>
<tr>
<td>bufferp:</td>
<td>pointer to an additional application buffer (reserved for future use)</td>
</tr>
</tbody>
</table>

### Cautions

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

```c
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 */
```
d42_chnstatus() retrieves the current channel status

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()
closes a feature session  

\[ \text{d42\_closefeaturesession( )} \]

- **Name:** \( \text{int d42\_closefeaturesession(devh)} \)
- **Inputs:** \( \text{int devh} \)
  - channel device
- **Returns:**
  - 0
  - -1
  - if error; see Errors below
- **Includes:** D42LIB.H
- **Mode:** immediate

### Description

The \( \text{d42\_closefeaturesession( )} \) 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, \( \text{d42\_writetodisplay( )} \).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>specifies the channel number.</td>
</tr>
</tbody>
</table>

### 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

```c
void main(void) {
    int devh;
    int rc = 0;
    char szDnNumber = "221";

    /* Open Channel Device */
    if ( (devh = dx\_open("dxxxB1C1",\null)) == -1) {
        printf("Error dx\_open()\n");
        exit(-1);
    }
    
    /* Open Feature Session */
```
\textit{d42\_closefeaturesession( )} \hspace{1cm} \textit{closes a feature session}

```c
if ( (rc = d42\_openfeaturesession (devh, szDnNumber, &iTerminalType, iEvtMask)) == -1) {
    printf("Error d42\_openfeaturesession():\n");
    dx\_close(devh);
    exit(-1);
} /* End d42\_brdstatus*/
```

```c
/*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*/
```

### 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 \textit{Appendix C}.

- ED42\_BADDEVICE: 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

- \textit{d42\_openfeaturesession( )}
- \textit{d42\_writetodisplay( )}
retrieves the current LCD/LED display

```
Name:    int d42_display(devh, bufferp)
Inputs: int devh
         char *bufferp
      • channel descriptor
      • 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

```
<table>
<thead>
<tr>
<th>No.</th>
<th>M</th>
<th>L</th>
<th>O</th>
<th>G</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>4E 6F 65 6C 20 4D 63 4C 6F 75 67 6E 69 6E 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Con</th>
<th>sult</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>20 20 20 20 20 20 20 20 43 6F 6E 73 75 6C 74 61</td>
</tr>
<tr>
<td>byte</td>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td>
</tr>
</tbody>
</table>
```

```
?  w  n  ?
| data  | 74 69 6E 6F 6F 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |
```
*d42_display(*) retrieves the current LCD/LED display*

**MITEL SUPERSET 420 - 32-character display**

<table>
<thead>
<tr>
<th>Data</th>
<th>43 41 4C 46 4F 52 44 47 3F 59 65 20 20 20 20 20 20 20 20 20 4E 6F 20 20 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
</tbody>
</table>

**Nortel Norstar - 32-character display**

<table>
<thead>
<tr>
<th>Data</th>
<th>54 72 61 6E 73 66 65 72 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>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 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td>
</tr>
</tbody>
</table>
retrieves the current LCD/LED display \( d42\_display() \)

<table>
<thead>
<tr>
<th>data</th>
<th>xx xx xx xx xx xx xx xx xx xx xx xx xx xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47</td>
</tr>
</tbody>
</table>

- **Nortel Meridian 1 - 48-character display**

<table>
<thead>
<tr>
<th>data</th>
<th>61 32 01 00 04 05 20 20 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data</th>
<th>20 20 20 20 20 20 20 20 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data</th>
<th>20 20 20 20 20 20 20 20 20 20 20 20 20 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47</td>
</tr>
</tbody>
</table>

### 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.
**d42_display()** retrieves the current LCD/LED display

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

```c
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
retrieves the current LCD/LED display  

\[ d42\_display() \]

- **ED42\_UNSUPPOR TED**  Function not supported on this board
- **ED42\_SYSTEM**  System level error
- **ED42\_INVALARG**  Invalid argument passed to function

**See Also**
- \[ d42\_displayex() \]
**d42_displayex()** retrieves the current LCD/LED display

**Name:** int d42_displayex(devh, bufferp)
**Inputs:** int devh
**Returns:** char *bufferp

- channel descriptor
- pointer to an application buffer. The buffer will contain display data for the selected channel.

buflen

- length of buffer

**Includes:**
- 0
- -1

**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

| 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| 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| 20|

### Siemens ROLM - 60-character display

| 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 | 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 |
|------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
retrieves the current LCD/LED display  

\textit{d42\_displayex( )}

\section*{Siemens ROLM - 60-character display}

\begin{verbatim}
data 43 4F 4E 46 45 52 45 4E 43 45 20 01 02 03 20 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
byte 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
\end{verbatim}

\section*{MITEL SUPERSET 430 - 80-character display}

\begin{verbatim}
data 01 00 01 20 41 43 55 52 52 41 4E 20 49 53 20 43 41 4C 4C 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 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 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
byte 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
\end{verbatim}

\begin{tabular}{|l|l|}
\hline
\textbf{Parameter} & \textbf{Description} \\
\hline
devh: & specifies the valid channel device descriptor obtained by a call to \texttt{dx\_open( )} \\
bufferp: & pointer to the application buffer. The buffer will contain the display data in ASCII format. \\
buflen: & length of buffer on entry. \\
\hline
\end{tabular}
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 \texttt{d42_displayex()}. or you can call the \texttt{d42_displayex()} function until valid display data is returned.
Example

```c
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
\textit{d42\_displayex()} \hspace{1cm} \textit{retrieves the current LCD/LED display}

ED42\_MEMORY \hspace{1cm} Buffer not large enough

\textbf{See Also}

- \textit{d42\_display()}
allows messages to be returned to a board  \(d42\_getnewmessage()\)

<table>
<thead>
<tr>
<th>Name:</th>
<th>int d42_getnewmessage(channel, bufferp)</th>
</tr>
</thead>
</table>
| Inputs: | unsigned int channel  
| | unsigned char *bufferp |
| Returns: | 0  
| | -1 |
| 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>specifies the channel number.</td>
</tr>
<tr>
<td>bufferp</td>
<td>points to the buffer where messages are placed</td>
</tr>
</tbody>
</table>

**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**

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

/* Get new message */
if ( (rc = d42_getnewmessage(channel, &buffer))
```
d42_getnewmessage( ) allows messages to be returned to a board

```c
if (d42_getnewmessage() == ERR_SUCC) {
    printf("d42_getnewmessage() == %d %s, channel = %d, Message = %s", channel, buffer);
} else {
    printf("d42_getnewmessage() == %d %s", 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( )

36
retrieves the selected channel or board parameter  

\[ d42\_getparm() \]

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devh:</td>
<td>specifies the valid board device or channel device descriptor obtained by a call to <code>dx_open()</code></td>
</tr>
<tr>
<td>parmnum:</td>
<td>contains the parameter name to retrieve</td>
</tr>
<tr>
<td>parmvalp:</td>
<td>pointer to the application variable that will receive the parameter value</td>
</tr>
</tbody>
</table>

---

## 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.
*d42_getparm*() retrieves the selected channel or board parameter

**Table 1. Board and Channel Parameters for d42_getparm()**

<table>
<thead>
<tr>
<th>Board Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4BD_CALLID</td>
<td>Enable Caller ID</td>
</tr>
<tr>
<td></td>
<td>Values: 0 - disable (default)</td>
</tr>
<tr>
<td></td>
<td>1 - enable</td>
</tr>
<tr>
<td></td>
<td>Obtains the switch type</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
</tr>
<tr>
<td></td>
<td>PBX_L4 - Lucent 75/85</td>
</tr>
<tr>
<td></td>
<td>PBX_L2 - Lucent G3</td>
</tr>
<tr>
<td></td>
<td>PBX_SH - Siemens Hicom</td>
</tr>
<tr>
<td></td>
<td>PBX_SR - Siemens ROLM</td>
</tr>
<tr>
<td></td>
<td>PBX_NS - Norstar</td>
</tr>
<tr>
<td></td>
<td>PBX_M1 - Meridian 1</td>
</tr>
<tr>
<td></td>
<td>PBX_SX - MITEL SX-50</td>
</tr>
<tr>
<td></td>
<td>PBX_SX2 - MITEL SX-200ML or SX-2000</td>
</tr>
<tr>
<td>D4BD_GETSWITCHTYPE</td>
<td>Enable report reset</td>
</tr>
<tr>
<td></td>
<td>Values: 0 - disable (default)</td>
</tr>
<tr>
<td></td>
<td>1 - enable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4CH_CHANNELSTATUS</td>
<td>Receive asynchronous channel status messages</td>
</tr>
<tr>
<td></td>
<td>Values: 0 - disable (default)</td>
</tr>
<tr>
<td></td>
<td>1 - enable</td>
</tr>
<tr>
<td>D4CH_LC_LAMP</td>
<td>Lamp to monitor for loop current</td>
</tr>
<tr>
<td>D4CH_CHANNELUPDATE</td>
<td>Enable/Disable asynchronous LCD and indicator updates</td>
</tr>
<tr>
<td>D4CH_CALLERIDAVAILABLE</td>
<td>Enables notification of Caller ID availability using the T_CALLERIDAVAILABLE event.</td>
</tr>
<tr>
<td></td>
<td>Values: 0 - disable (default)</td>
</tr>
</tbody>
</table>
retrieves the selected channel or board parameter  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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

```c
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 */

    /* Get the Board Parameter To See if Speakerphone Mode is Enabled */
    if ((parmnum = d42_getparm(devh, D4B_SPMODE, (void *)&parmvalp)) == -1)
    {
        printf("Error d42_getparm()\n");
        d4c_close(devh);
        exit(-1);
    } /* End d42_getparm */

    /* Check if Speakerphone is enabled */
    if (parmvalp == 1)
```
d42_getparm() retrieves the selected channel or board parameter

{
    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 */
retrieves the selected channel or board parameter

\[ \text{d42_getparm()} \]

### 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

- \[ \text{d42_setparm()} \]
**d42_getver( )** retrieves the board firmware or library version

**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
retrieves the board firmware or library version \textit{d42_getver()} \\

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devh:</td>
<td>specifies the valid board device descriptor obtained by a call to \textit{dx_open()}</td>
</tr>
<tr>
<td>bufferp:</td>
<td>pointer to the application buffer that will contain the version data</td>
</tr>
<tr>
<td>flag:</td>
<td>determines if the firmware or library version number is placed in the application buffer.</td>
</tr>
<tr>
<td></td>
<td>\begin{itemize}</td>
</tr>
<tr>
<td></td>
<td>\item VER_D42FIRMWARE - returns the D/42-xx or PBX Integration board firmware version</td>
</tr>
<tr>
<td></td>
<td>\item VER_D42LIB - returns the D42 library (D42LIB) version</td>
</tr>
</tbody>
</table>

**Cautions**

The application buffer must be at least 100 bytes.

**Example**

```c
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\n",bufferp);
    dx_close(devh);
}  /* End main */
```

\texttt{d42\_getver( )} \textit{retrieves the board firmware or library version}

\section*{Errors}

If this function returns -1 to indicate a failure, one of the following (most common) codes will be contained in \texttt{dx\_errno}. For a complete list of error codes and definitions, refer to \textit{Appendix C}.

- \texttt{ED42\_BADDEVICE} \quad Invalid or wrong device handle
- \texttt{ED42\_UNSUPPORTED} \quad Function not supported on this board
- \texttt{ED42\_SYSTEM} \quad System level error
- \texttt{ED42\_RDFWVER} \quad Error reading firmware version
- \texttt{ED42\_INVALARG} \quad Invalid argument passed to function
d42_gtcallid( )

**Name:** int d42_gtcallid(devh, bufferp)

**Inputs:**
- int devh
- char *bufferp

  - channel descriptor
  - 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:

<table>
<thead>
<tr>
<th>Text</th>
<th>Data</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>Text</td>
<td>Data</td>
<td>Byte</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>

**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