

Dialogic[®] D/42 Series Software API Library Reference

Copyright © 2000-2008 Dialogic Corporation

05-1158-003

Copyright and Legal Notice

Copyright © 2000-2008 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT, EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC. DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by 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 provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Brooktrout, Cantata, SnowShore, Eicon, Eicon Networks, Eiconcard, Diva, SIPcontrol, Diva ISDN, TruFax, Realblobs, Realcomm 100, NetAccess, Instant ISDN, TRXStream, Exnet, Exnet Connect, EXS, ExchangePlus VSE, Switchkit, N20, Powering The Service-Ready Network, Vantage, Making Innovation Thrive, Connecting People to Information, Connecting to Growth and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and product mentioned herein are the trademarks of their respective owners.

Publication Date: November 2008

Document Number: 05-1158-003

Table of Contents

1. How To Use This Manual	9
1.1. Audience	9
1.2. Voice Hardware Covered by This Manual.....	9
1.2.1. Voice Hardware Model Names	10
1.3. When To Use This Manual	11
1.4. Documentation Conventions	11
1.5. How This Manual Is Organized	12
2. Using the PBX Functions	13
2.1. The Dialogic® Unified API	13
2.2. Switch-Specific Support	14
3. Dialogic® Unified API Function Reference	17
ATD4_BDTYPE() - returns the D/42-xx board type.....	18
ATD4_CHTYPE() - returns the D/42-xx channel type	20
d42_brdstatus() - retrieves the current D/42-xx board status.....	22
d42_chnstatus() - retrieves the current D/42-xx channel status	24
d42_closefeaturesession() - closes an open feature session.....	26
d42_display() - retrieves the current LCD/LED display	28
d42_flags() - retrieves current D/42D-SX LCD Features Display data	32
d42_getparm() - retrieves a D/42-xx channel or board parameter	36
d42_getver() - retrieves the D/42-xx board firmware or library version	39
d42_gtcalled() - retrieves the called/calling number ID	42
d42_indicators() - retrieves the status of LCD/LED indicators	45
d42_lcdprompt() - retrieves the current LCD prompt data of the D/42D-SX.....	57
d42_openfeaturesession() - opens a phone extension feature session.....	62
d42_setparm() - sets a D/42-xx board or channel parameter	65
d42_writetodisplay() - writes to the phone set display	71
4. Programming Considerations	73
4.1. Opening a Channel on a Dialogic® D/42-xx Board	73
4.2. Accessing PBX Features on a PBX Using Dial Strings.....	75
4.2.1. Turn On the Message Waiting Indicator	76
4.2.2. Turn Off the Message Indicator	79
4.2.3. Dial Programmable Keys	82
4.2.4. Transferring a Call.....	96
4.2.5. In-Band/Out-of-Band Signaling	97
4.3. Disconnect Supervision.....	98

Dialogic® D/42 Series Software API Library Reference

4.4. Converting Existing Dialogic® D/4x Applications	99
Appendix A - Dialogic® D/42 Series Software Quick Reference	101
Appendix B - Dialogic® D/42 Series Software Demonstration Program	109
Requirements.....	109
Setup.....	109
Documentation Conventions	110
Running the Demo.....	110
Appendix C - Error and Event Definitions.....	119
Glossary	121
Index.....	129

List of Tables

Table 1. MITEL SUPERSET 4 Features Display Descriptions	33
Table 2. D/42-NE2 Indicator Status Definitions	52
Table 3. MITEL SUPERSET 4 Prompt Descriptions	59
Table 4. Dialogic® D/42 Board Parameters for d42_getparm() and d42_setparm() Functions	67
Table 5. Dialogic® D/42 Channel Parameters for d42_getparm() and d42_setparm() Functions	68
Table 6. MITEL Direct Key Dialing Sequences	84
Table 7. Northern Telecom SL-1 Direct Key Dialing Sequences	87
Table 8. Northern Telecom Norstar Direct Key Dialing Sequences	89
Table 9. NEC KTS/PBX Direct Key Dialing Sequences	92
Table 10. Setting In-Band and Out-of-Band Signaling	98
Table 11. Demo Indicator Definitions	117
Table 12. List of Error Codes	119
Table 13. List of Event Codes	120

Dialogic® D/42 Series Software API Library Reference

List of Figures

Figure 1. Contents of the Features Display Application Buffer.....	33
Figure 2. MITEL SUPERSET 4 Telephone Indicators.....	47
Figure 3. Northern Telecom Digit Display Telephone Indicators.....	49
Figure 4. Northern Telecom Model 7310 Telephone Indicators.....	51
Figure 5. NEC Dterm Series III Telephone Indicators.....	54
Figure 6. MITEL SUPERSET 4 Prompt Display	58
Figure 7. MITEL SUPERSET 4 Telephone.....	83
Figure 8. Northern Telecom Digit Display Telephone.....	86
Figure 9. Northern Telecom Model 7310 Telephone.....	88
Figure 10. NEC Dterm III Telephone	91
Figure 11. Dialogic® D42 Demo Window	111
Figure 12. D42 Options Window	111
Figure 13. Select Your D/42 Channel.....	112
Figure 14. Select a D/42 Channel	113
Figure 15. Northern Telecom M7310 Window.....	114
Figure 16. NEC Dterm Series III Window.....	115

Dialogic® D/42 Series Software API Library Reference

1. How To Use This Manual

1.1. Audience

This manual is written for programmers and engineers who are interested in using the Dialogic® D/42 Series Software, together with standard Dialogic® D/4x Voice Software, to develop voice and call processing applications 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 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 *Dialogic® Voice API Programming Guide*. The *Dialogic® Voice API Programming Guide* provides an introduction to the Dialogic® voice products, with explanations and help beyond a strictly technical level so that you can quickly learn the Dialogic® 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.

1.2. Voice Hardware Covered by This Manual

The Dialogic® D/42 Series voice hardware (also referred to as Dialogic® D/42-xx) is designed to provide a set of cost-effective tools for implementing computerized, voice and call processing applications for private branch exchange (PBX) systems and key telephone systems (KTSs). It provides the basic voice and call processing capabilities of Dialogic® D/4x voice hardware and adds hardware and firmware required to integrate with PBXs and KTSs. Refer to the *Dialogic® Voice API Programming Guide* for more information about voice and call processing. For convenience, the terms private branch exchange (PBX), key system unit (KSU), and key telephone system (KTS) will be referred to as PBX.

Dialogic® D/42 Series Software API Library Reference

The Dialogic® voice hardware models covered by this manual include the following:

NOTE: Although the **Dialogic® D/42D-SX** and **Dialogic® D/42D-SL** Boards are documented in this manual, they are not supported in this release.

Dialogic® D/42D-SX— a 4-channel voice board with station interfaces for connecting directly to a MITEL SUPERSET 4 Line Circuit card in a MITEL SUPERSWITCH PBX.

Dialogic® D/42D-SL— a 4-channel voice board with station interfaces for connecting to a Northern Telecom Digit Display (QPC 451 or QPC 61) Line Circuit card in a Northern Telecom SL-1 PBX.

Dialogic® D/42-NS— a 4-channel voice board with station interfaces for connecting to a Northern Telecom NORSTAR key system unit (KSU).

Dialogic® D/42-NE2— a 4-channel voice board with digital interfaces for connecting to NEC Electra Professional Level II telephone systems, as well as NEAX 2000 IVS and NEAX 2400 IMS PBX series switches. Throughout this manual, the NEC Electra Professional Level II is referred to as the NEC KTS, while the NEAX 2000 IVS and NEAX 2400 IMS are referred to as the NEC PBX.

1.2.1. Voice Hardware Model Names

Model names for voice boards other than the Dialogic® HD series are based upon the following pattern:

D / x x x y

where:

D/	identifies the board as voice hardware
xxx	identifies the number of channels (2, 4, 8, 12, etc.), followed by a code indicating whether call progress analysis is supported
0	indicates no support for call progress analysis
1	indicates support for call progress analysis
2	indicates PBX support
y	if present, identifies a hardware version (A, B, C, D, etc.)

1. How To Use This Manual

Sometimes it is necessary 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, Dialogic® D/xxx refers to all models of the voice hardware, and Dialogic® D/4x refers to all 4-channel models.

1.3. When To Use This Manual

This *Dialogic® D/42 Series Software API Library Reference* contains programming information for developing applications in the Windows® operating system environment using the Dialogic® Unified API and Dialogic® D/42 Runtime Library. The Dialogic® Unified API provides a single, basic set of high-level calls used to develop applications across a variety of manufacturer’s switches. The Dialogic® D/42 Runtime Library supports the Dialogic® Unified API and works in conjunction with the standard Dialogic® Voice Library to enable applications to set up calls and perform PBX call functions using Dialogic® D/42-xx Boards.

This manual also includes instructions for using the Dialogic® D/42 demonstration program in the Windows® operating system environment.

Refer to this manual, the *Dialogic® D/42 Series Boards User’s Guide*, *Dialogic® Voice API Library Reference*, and *Dialogic® Voice API Programming Guide* 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*.
- 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 **dx_dial()**.

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

1.5. How This Manual Is Organized

Chapter 1 – How To Use This Manual describes the *Dialogic® D/42 Series Software API Library Reference*.

Chapter 2 – Using the PBX Functions provides information on using the voice library functions with Dialogic® D/42-xx products.

Chapter 3 – Dialogic® Unified API Function Reference provides technical information on the voice software C language voice library functions.

Chapter 4 – Programming Considerations contains programming information about developing applications for the MITEL PBX, Northern Telecom SL-1 PBX, Northern Telecom Norstar KSU, and the NEC Electra Professional Level II KTS, NEAX 2000 IVS, and NEAX 2400 IMS PBX.

Appendix A – Dialogic® D/42 Series Software Quick Reference provides summary information on the voice software C language voice library functions.

Appendix B – Dialogic® D/42 Series Software Demonstration Program provides instructions for running this demonstration program.

Appendix C – Error and Event Definitions lists error codes and event codes.

Glossary contains a list of definitions for commonly used terms.

2. Using the PBX Functions

The PBX circuitry on the Dialogic® D/42-xx Boards provides functions specific to a PBX. These functions are implemented using the Dialogic® D/42 Runtime Library (*LIBD42MT.DLL*). The Dialogic® D/42 Runtime Library is used in addition to the standard Dialogic® Voice Library when tight integration and control of the PBX and Dialogic® D/42-xx Boards are required.

The Dialogic® Voice Library is an interface between the application program and the Dialogic® D/42 Series voice hardware. The Voice Library is used to access standard voice functions such as voice play/record and call progress analysis. Refer to the *Dialogic® Voice API Library Reference* for information about using voice functions.

2.1. The Dialogic® Unified API

The Dialogic® 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 Dialogic® 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 Dialogic® D/42 Series Board, without additional hardware. Functioning as an extension to the Dialogic® Voice API, the Dialogic® 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 Dialogic® Unified API can shorten 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.

Developers who wish to continue designing switch-specific applications can continue to do so, as the Dialogic® Unified API also provides access to lower-level function calls made available through each individual switch protocol. And for customers unwilling to shift from older PBX integration development models,

Dialogic® D/42 Series Software API Library Reference

the Dialogic® Unified API provides for backward compatibility, preserving their development investment.

Utility functions included in the Dialogic® Unified API allow programmers to control the Dialogic® D/42 Series Board. The application can retrieve the Dialogic® D/42-xx channel type, obtain and set Dialogic® D/42-xx channel parameters, start and stop the Dialogic® D/42 driver, and retrieve Dialogic® D/42 firmware/driver/library version numbers. The application can retrieve error information using the **ATDV_LASTERR()** and **ATDV_ERRMSGP()** functions in the standard Dialogic® Voice API.

The Dialogic® D/42 Runtime Library works in conjunction with the standard Dialogic® Voice Library to enable applications to set up calls and perform PBX call functions using Dialogic® D/42-xx Boards. In addition, the Dialogic® D/42 Runtime Library supports the Dialogic® Unified API.

NOTE: The Dialogic® Unified API contains both synchronous and asynchronous functions. These terms, along with programming models, callback, and event handlers, are discussed in the *Dialogic® Standard Runtime Library API Programming Guide*.

The Dialogic® 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 Dialogic® D/42-xx 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 affect 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 Dialogic® 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
- Conference

2. Using the PBX Functions

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

Refer to the *Dialogic® D/42 Series Boards User's Guide* for information about PBX and KTS features. Because the Dialogic® D/42-xx Boards have the capability to emulate a PBX station set, they 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 overhead paging. In addition, your application can reprogram keys as needed. Refer to *Chapter 4* for details about switch-specific programming.

Dialogic® D/42 Series Software API Library Reference

3. Dialogic[®] Unified API Function Reference

This chapter provides technical information on the PBX interface software C language library functions (the Dialogic[®] 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 description of the functions that are described in this chapter.

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 syntax, input parameters, output or returns, includes (header files required to be included), and mode. The function syntax and inputs include the data type and are shown using standard C language syntax.
Description	Provides a description of the function operation, including parameter descriptions.
Cautions	Provides warnings and reminders.
Example	Provides one or more C language coding examples showing how the function can be used.
Errors	Lists the error codes that could be returned by the function.

■ Example

```

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

    /* Open Board Device */
    if ( (devh = dx_open("dxxxB1",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, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
DXLIB_INVNRB	Internal voice library error

ATD4_CHTYPE()*returns the D/42-xx channel type*

Name: int ATD4_CHTYPE(devh)
Inputs: int devh • channel descriptor
Returns: channel type • channel type information if success (see below)
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ Description

The **ATD4_CHTYPE()** function returns the D/42-xx channel type of the queried device.

Channel Type	Description
TYP_SL	for the Dialogic® D/42D-SL
TYP_SX	for the Dialogic® D/42D-SX
TYP_NS	for the Dialogic® D/42-NS
TYP_NE2PBX	for the Dialogic® D/42-NE2 used with the NEC NEAX 2000 IVS or NEAX 2400 IMS PBX
TYP_NE2KTS	for the Dialogic® D/42-NE2 used with the NEC Electra Professional Level II KTS
TYP_NE2	for other or unknown Dialogic® D/42-NE2 Boards
TYP_NONE	for non-Dialogic® D/42 Boards

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, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_INVALIDARG	Illegal argument in function
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
DXLIB_INVNRB	Internal voice library error

d42_brdstatus()*retrieves the current D/42-xx board status*

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: ED42_NOERROR • if success
-1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ Description

The **d42_brdstatus()** function retrieves the current D/42-xx 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. The application buffer (**buffstatus**) that will contain the board status information must be 1 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
bits 4-7 reserved for future use

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 48-byte plus 1 NULL byte 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 Board 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, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

d42_chnstatus() ***retrieves the current D/42-xx 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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ Description

The **d42_chnstatus()** function retrieves the current D/42-xx channel status and places it in an application buffer. The application buffer (**statusp**) that will contain the channel status information must be 1 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 the channel is communicating with the switch: non-zero = OK 0 = no communications
bufferp	pointer to an additional 48-byte plus 1 NULL byte 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, &status, bufferp)) == -1)
    {
        printf("Error d42_chnstatus():\n");
        dx_close(devh);
        exit(-1);
    } /* End d42_brdstatus*/

    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, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADDEVICE	Invalid or wrong device handle
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

d42_closefeaturesession()***closes an open feature session***

Name: int d42_closefeaturesession(devh)
Inputs: int devh • channel descriptor
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-NS

■ Description

The **d42_closefeaturesession()** function closes an open feature session on the specified channel, terminating the association between the telephone extension and the channel number. It also disables the asynchronous events that were enabled for the feature session and disallows the use of any functions that require an open feature session, such as **d42_writetodisplay()**. The **d42_closefeaturesession()** function requires that the feature session be previously opened.

Parameter Description

devh	specifies the valid channel device descriptor obtained by a call to dx_open() and on which the feature session is open
-------------	--

■ Cautions

This function requires an open feature session.

■ Example

```
void D42_FeatureSession()
{
char szExt[5];
char *cpNull = NULL;
char cpErrStr[MAX_PATH];
int TermType;
int eventMask;

// extension fpr which we are opening a feature session
sprintf(szExt, "1234");
```

closes an open feature session

d42_closefeaturesession()

```
eventMask = D42_EVT_SOFTKEY | D42_EVT_ASYNC_CLOSEFEATURESESSION;
if (d42_openfeaturesession(G_d42chdev, szExt, &TermType, eventMask) == -1)
{
    printf("Error d42_openfeaturesession()\n");
}

if (d42_writetodisplay(G_d42chdev, "FtrSs Open") == -1)
{
    printf("Error d42_writetodisplay()\n");
}

if (d42_closefeaturesession(G_d42chdev) == -1)
{
    printf("Error d42_closefeaturesession()\n");
}
} // endof D42_FeatureSession
```

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_NOFEATURE SESSION	Function requires an open feature session
ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_UNSUPPORTED	Function not supported on this board
DXLIB_INVNRB	Internal voice library error

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

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 variable (currently 16 or 32 bytes), and is stored as a null-terminated ASCII string. The 49 byte buffer size is for future expansion. An application that passes anything smaller will not be backward compatible in future releases.

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);
}
```

retrieves the current LCD/LED display

d42_display()

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

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADDEVICE	Invalid or wrong device handle
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

d42_flags() *retrieves current D/42D-SX LCD Features Display data*

Name: int d42_flags(devh, bufferp)
Inputs: int devh • channel descriptor
 char *bufferp • pointer to an application buffer. The buffer will contain the Features Display data.
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SX

■ Description

The **d42_flags()** function retrieves current D/42D-SX LCD Features Display data and places it in the application buffer. The application buffer must be 49 bytes, and will contain a bit mask representing the status of each flag in the Features Display. Refer to the *Dialogic® D/42 Series Boards User's Guide* for more information.

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 Features Display data.

The Features Display data stored in the application buffer is 16 bytes long. Each byte represents a specific flag on the SUPERSET 4 Feature Display. Refer to *Table 1* for a description of each Feature Display LCD flag. The value of each byte can be 0x00 (off), 0x01 (on), or 0x02 (flashing). *Figure 1* shows the contents of the application buffer when the Features Display illuminates "3:AUTO ANS".

d42_flags()*retrieves current D/42D-SX LCD Features Display data***■ 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 LCD Flags Information */
    if ( (rc = d42_flags(devh, bufferp)) == -1)
    {
        printf("Error d42_flags()\n");
        dx_close(devh);
        exit(-1);
    } /* End d42_flags*/

    /* Test 1 Flag */
    if (buffer[1] == 1)
    {
        printf("Microphone is ON");
    }

    else if (buffer[1] == 0)
    {
        printf("Microphone is OFF");
    }

    else
        /* must be flashing */
    {
        printf("Microphone is FLASHING");
    } /* End Test 1 Flag */

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

■ **Errors**

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADDEVICE	Invalid or wrong device handle
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

d42_getparm() *retrieves a D/42-xx channel or board parameter*

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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ Description

The **d42_getparm()** function retrieves a D/42-xx 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.

Additional notes and a list of board and channel parameters that can be retrieved for **parmnum** are provided under the **d42_setparm()** function.

Parameter	Description
devh	specifies the valid board device or channel device descriptor obtained by a call to dx_open()
parmnum	specifies the define for the parameter type whose value is to be returned in the variable pointed to by parmvalp (see <i>Table 4</i> and <i>Table 5</i> under the d42_setparm() function)
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 except D4BD_MSGACCESSION and D4BD_MSGACCESSOFF. Both D4BD_MSGACCESSION and D4BD_MSGACCESSOFF should be treated as ASCII strings (char *). The application should cast the **parmvalp** parameter to a (void *) to avoid compiler warnings.

■ Example

```

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

    /* Open Board Device */
    if ( (devh = dx_open("dxxxxB1",NULL))==-1)
    {
        printf("Error dx_open()\n");
        exit(-1);
    } /* End dx_open */

    if ( (ATD4_BDTYPE (devh)) == TYP_SX)
    {
        /* Get the Board Parameter To See if Speakerphone Mode is Enabled */
        if ( (rc = d42_getparm(devh, D4BD_SPMODE, (void *)&parmval)) == -1)
        {
            printf("Error d42_getparm(D4BD_SPMODE)\n");
            dx_close(devh);
            exit(-1);
        } /* End d42_setparm */

        /* Check if Speakerphone is enabled */
        if (parmval == 1)
        {
            printf("Speakerphone Mode is ENABLED");
        }

        else if (parmval == 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, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADPARAM	Invalid value for parameter
ED42_BADDEVICE	Invalid or wrong device handle

d42_getparm() ***retrieves a D/42-xx channel or board parameter***

ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

retrieves the D/42-xx board firmware or library version

d42_getver()

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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: all Dialogic® D/42 Boards

■ Description

The **d42_getver()** function retrieves the D/42-xx board firmware or library version and places it in an application buffer pointed to by **bufferp**. The **flag** specifies what should be returned (firmware or library version number). The application buffer is at least 100 bytes long and returns version number information in the following formats:

Firmware Firmware Version: X.XX <type> YYYY

 or

 Firmware Version: X.XX <type> Y.YY

 where: **X.XX** represents the version number

 <**type**> represents the type of release (Production, Beta,
 Alpha, Experimental, Special, Build, Unknown)

Y.YY or **YYYY** represents a special release number (e.g.,
 experimental number)

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

 where: **YY** represents the year

MM represents the month

XX.XX represents a version number

d42_getver() *retrieves the D/42-xx board firmware or library version*

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 as a null-terminated ASCII string
flag	determines if the firmware or library version number is placed in the application buffer. VER_D42FIRMWARE - returns the Dialogic® D/42-xx firmware version VER_D42LIB - returns the Dialogic® D42 library (<i>LIBD42MT.DLL</i>) version

■ Cautions

The application buffer must be at least 100 bytes long.

■ 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 */
```

retrieves the D/42-xx board firmware or library version

d42_getver()

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_RDFWVER	Error reading firmware version
ED42_INVALIDARG	Invalid argument passed to function
EDX_SYSTEM	System level error
DXLIB_INVNRB	Internal voice library error

d42_gtcallid()*retrieves the called/calling number ID*

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: ED42_NOERROR • if success
-1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ 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 *Dialogic® D/42 Series Boards User's Guide* for more information specific to your PBX. Examples showing the contents of the application buffer for each supported switch are as follows:

■ MITEL PBX (Dialogic® D/42D-SX Board)

text	bb 2 2 1 _ C A L L I N G
data	20 32 32 31 5F 43 41 4C 4C 49 4E 47 00 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 23
data	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 47

■ Northern Telecom SL-1 (Dialogic® D/42D-SL Board)

text	bb bb bb bb bb bb bb bb bb bb bb bb 2 2 1
data	20 20 20 20 20 20 20 20 20 20 20 20 20 20 32 32 31 00 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 23
text	
data	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 47

retrieves the called/calling number ID

d42_gtcallid()

■ Northern Telecom Norstar (Dialogic® D/42-NS Board)

text	bb 2 2 1 _ 2 2 4
data	20 32 32 31 5F 32 32 34 00 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 23
text	
data	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 47

■ NEC KTS/PBX (Dialogic® D/42-NE2 Board)

text	bb 2 0 0 _ 2 0 3
data	20 32 30 30 5F 32 30 33 00 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 23
text	
data	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 47

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.

■ Cautions

The application buffer must be 49 bytes. The length of the called/calling number ID data is variable (not exceeding 48 bytes), and is stored as a null-terminated ASCII string (total length 49 bytes).

NOTE: During testing of the Dialogic® D/42D-SX Board, it was determined that called/calling number ID data was not always sent by the PBX prior to the ring event. To ensure that the correct called/calling ID data is obtained, the application should be set up to answer a call only after the second ring.

■ 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 Calling/Caller Id */
    if ( (rc = d42_gtcallid(devh, bufferp)) == -1)
    {
        printf("Error d42_gtcallid()\n");
        dx_close(devh);
        exit(-1);
    } /* End d42_gtcallid */

    printf("Caller Id = %s\n",bufferp);

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

```

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

retrieves the status of LCD/LED indicators

d42_indicators()

Name: int d42_indicators(devh, bufferp)
Inputs: int devh • channel descriptor
 char *bufferp • pointer to an application buffer
 containing the indicators data
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

■ Description

The **d42_indicators()** function retrieves the status of LCD/LED indicators and places it in an application buffer. The application buffer must be 49 bytes, and will hold the entire bit mask (see below) representing the status of each indicator. Refer to the *Dialogic® D/42 Series Boards User's Guide* for more information specific to your switch. Examples showing the contents of the application buffer for each supported switch are as follows:

D/42D-SL	8 indicators
D/42D-SX	16 indicators
D/42-NS	10 indicators
D/42-NE2	24 indicators

Parameter	Description
devh	specifies the valid channel device descriptor obtained by a call to dx_open()
bufferp	pointer to the application buffer; the indicator is placed here

■ Cautions

The application buffer must be 49 bytes. The length of the line indicator data is variable (currently 8, 10, 16, and 24 bytes), and is stored as bit mask. The 49 byte buffer size is for future expansion. An application that passes anything smaller will not be backward compatible in future releases.

■ MITEL PBX (Dialogic® D/42D-SX Board)

There are 16 LCD Line Indicators (each containing two segments; a square and a circle) located on the left side of line keys 00-15 on the MITEL SUPERSET 4 telephone - see *Figure 2*. The line indicator status data stored in the application buffer is 16 bytes long. Byte 0 contains the status for Hold. Bytes 1-15 contain the indicator status for line keys 01-15, respectively. Each byte (8 bits) contains data for both segments of an indicator (bits 0-3 represent the square, bits 4-7 represent the circle). The status data for each byte is defined as follows:

Data for bits 0-3	Description
0x?0	square off
0x?1	square on
0x?2	square flashing 250 ms off, 250 ms on
0x?3	square flashing 500 ms off, 500 ms on
0x?4	square flashing 438 ms off, 62 ms on
0x?5	square flashing 62 ms off, 438 ms on
0x?F (square only)	inverse flash rate of circle
Data for bits 4-7	Description
0x0?	circle off
0x1?	circle on
0x2?	circle flashing 250 ms off, 250 ms on
0x3?	circle flashing 500 ms off, 500 ms on
0x4?	circle flashing 438 ms off, 62 ms on
0x5?	circle flashing 62 ms off, 438 ms on

■ Example

If the data for byte 7 is 0x02, the circle segment for Line Key 7 is off and the square segment is flashing at 250 ms. The contents of the application buffer are shown below.

d42_indicators()

retrieves the status of LCD/LED indicators

■ **Northern Telecom SL-1 (Dialogic® D/42D-SL Board)**

There are eight LED Line Indicators located on the top-right of the Digit Display telephone - see *Figure 3*. There are no indicators for Feature Keys 8 and 9. The line indicator status data stored in the application buffer is 8 bytes long. Bytes 0-7 contain the indicator status of Feature Keys 0-7, respectively. The status data for each byte is defined as follows:

Value (in HEX)	State
0x00	off
0x01	wink (flash 120 Hz)
0x02	flash (flash 60 Hz)
0x03	on

■ **Example**

If the data for byte 1 is 0x03 and byte 2 is 0x02, the indicator for Feature Key 1 is on and the indicator for Feature Key 2 is flashing at 60 Hz. The contents of the application buffer are shown below.

d42_indicators()

retrieves the status of LCD/LED indicators

■ **Northern Telecom Norstar (Dialogic® D/42-NS Board)**

There are 10 LCD Line Indicators located between Programmable Memory Buttons 0-9 on the Model 7310 telephone - see *Figure 4*. The indicator status data stored in the application buffer is 10 bytes long. Bytes 0-9 contain the indicator status of Memory Buttons 0-9, respectively. The status data for each byte is defined as follows:

Value (in HEX)	State
0x00	off
0x01	on
0x02	alerting (flashing)
0x03	Ihold
0x04	Uhold

■ **Example**

If the data for byte 1 is 0x01 and byte 8 is 0x02, the indicator for Memory Button 1 is on and the indicator for Memory Button 8 is alerting. The contents of the application buffer are shown below.

d42_indicators()*retrieves the status of LCD/LED indicators*

- Five 2-color indicators located on the FNC, CNF, LNR/SPD, SPKR, and ANS keys.
- Two indicators, MIC and ICM, located below the line keys .

The indicator status data stored in the application buffer is 32 bytes long. Each byte represents a specific indicator on the Dterm Series III telephone - see *Table 2*.

Table 2. D/42-NE2 Indicator Status Definitions

Byte	Description	Byte	Description
0	line LED 1	16	not used
1	line LED 2	17	message waiting indicator
2	line LED 3	18	CNF key LED
3	line LED 4	19	FCN key LED
4	line LED 5	20	not used
5	line LED 6	21	LNR/SPD key LED
6	line LED 7	22	ANS key LED
7	line LED 8	23	SPKR key LED
8	line LED 9	24	ICM key LED
9	line LED 10	25	MIC key LED
10	line LED 11	26	not used
11	line LED 12	27	not used
12	line LED 13	28	not used
13	line LED 14	29	not used
14	line LED 15	30	not used
15	line LED 16	31	not used

The two-color LEDs can take on one of the states listed below. The MIC and ICM indicators use only the red states.

Binary	Hex	Description
0000 0000	0x00	off
0000 0001	0x01	flutter (red)
0000 0010	0x02	wink (red)
0000 0011	0x03	rapid wink (red)
0000 0100	0x04	interrupted rapid wink (red)

d42_indicators()

retrieves the status of LCD/LED indicators

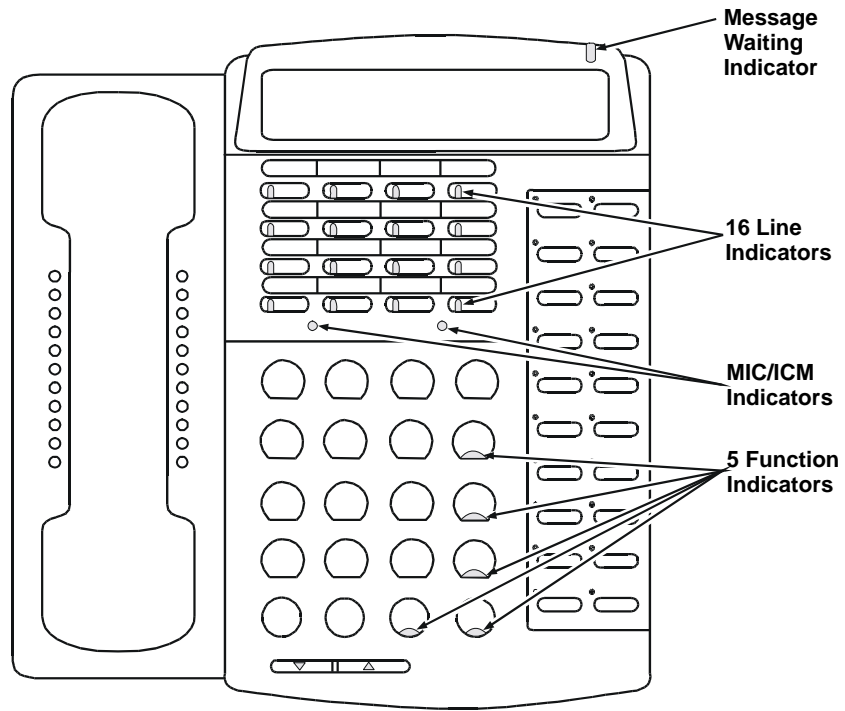


Figure 5. NEC Dterm Series III Telephone Indicators

■ Example

```

void main(void)
{
    int         devh;
    int         rc = 0;
    int         count;
    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 Calling/Caller Id */
    if ( (rc = d42_gtcallid(devh, bufferp)) == -1)
    {
        printf("Error d42_gtcallid()\n");
        dx_close(devh);
        exit(-1);
    } /* End d42_gtcallid */

    printf("Caller Id = %s\n",bufferp);

    /* Get the Indicator Information */
    if ( (rc = d42_indicators(devh, bufferp)) == -1)
    {
        printf("Error d42_indicators(): Error Code: %hX\n",ATDV_LASTERR(devh));
        dx_close(devh);
        exit(-1);
    } /* End d42_indicators*/

    for (count = 0; count < 49; count++)
    {
        printf("Indicator %d = %X\n",count, bufferp[count]);
    }

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

```

d42_indicators()

retrieves the status of LCD/LED indicators

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

d42_lcdprompt() retrieves the current LCD prompt data of the D/42D-SX

	PROGRAM	CAMP	SWAP	PAGE	PROGRAM	ON	SPLIT	EXIT	NIGHT	CALL	ON	NAME	ANS	BACK	OFF	SAVE	PICKUP	OVERRIDE	TRANS	CALL	🎵	OVERRIDE	CONF	FWD	
Data	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01
Byte	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	

	HELP	SEND	ADD	REMINER	CALL	MSG	HELD	REMINER	REDIAL	CANCEL	PRIVACY	READ	↓	NO	REL	MSG	RELEASE	HANG	NEXT	SPEED	↑	UP	YES	CALL	
Data	00	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Byte	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	

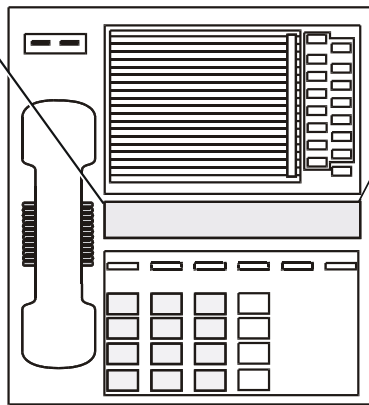
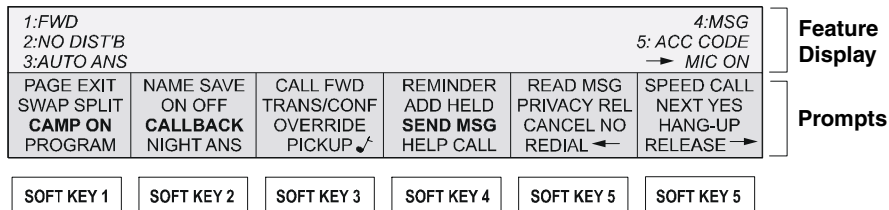


Figure 6. MITEL SUPERSET 4 Prompt Display

retrieves the current LCD prompt data of the D/42D-SX `d42_lcdprompt()`

Table 3. MITEL SUPERSET 4 Prompt Descriptions

Soft Key	Byte	Prompt	Soft Key	Byte	Prompt
1	0	PROGRAM	4	24	HELP
	1	CAMP		25	SEND
	2	SWAP		26	ADD
	3	PAGE		27	REMINDER
	4	PROGRAM		28	CALL
	5	ON		29	MSG
	6	SPLIT		30	HELD
	7	EXIT		31	REMINDER
2	8	NIGHT	5	32	REDIAL
	9	CALL		33	CANCEL
	10	ON		34	PRIVACY
	11	NAME		35	READ
	12	ANS		36	←
	13	BACK		37	NO
	14	OFF		38	REL
	15	SAVE		39	MSG
3	16	PICKUP	6	40	RELEASE
	17	OVERRIDE		41	HANG
	18	TRANS		42	NEXT
	19	CALL		43	SPEED
	20	⌂		44	→
	21	OVERRIDE		45	UP
	22	CONF		46	YES
	23	FWD		47	CALL

d42_lcdprompt() *retrieves the current LCD prompt data of the D/42D-SX*

■ Cautions

This function is only for the Dialogic® D/42D-SX Board. The application buffer must be 49 bytes. The length of the LCD prompt data is 48 bytes and is stored as a bit mask.

■ 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 LCD Prompt Information */
    if ( (rc = d42_lcdprompt(devh, bufferp)) == -1)
    {
        printf("Error d42_lcdprompt()\n");
        dx_close(devh);
        exit(-1);
    } /* End d42_lcdprompt*/
    dx_close(devh);
} /* End main */
```

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNSUPPORTED	Function not supported on this board
ED42_INVALIDARG	Invalid argument passed to function

retrieves the current LCD prompt data of the D/42D-SX *d42_lcdprompt()*

DXLIB_INVNRB Internal voice library error

d42_openfeaturesession() ***opens a phone extension feature session***

Name: int d42_openfeaturesession(devh, bufferp, termtype, evtmask)
Inputs: int devh • channel descriptor
 char *bufferp • pointer to a buffer specifying a
 valid phone extension number in
 ASCII character string format
 int *termtype • pointer to memory location that
 receives the type of phone display
 int evtmask • specifies the events to enable for
 the feature session
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-NS

■ **Description**

The **d42_openfeaturesession()** function opens a phone extension feature session on a specified channel, associating the telephone extension with the channel number. It returns information about the display used by the telephone set. It also enables for the feature session the asynchronous events specified in **evtmask** and allows the use of any functions that require an open feature session, such as **d42_writetodisplay()** and **d42_closefeaturesession()**.

Parameter	Description
devh	specifies the valid channel device descriptor obtained by a call to dx_open()
bufferp	pointer to ASCII character string application buffer where the NULL terminated phone extension number is placed
termtype	pointer to the application variable that will receive information on the type of phone display that is associated with the extension. The information is used in the d42_writetodisplay() function. Values returned are: 0x00 No display available 0x01 No display available 0x02 16-byte display 0x03 32-byte display

Parameter	Description
evtmask	specifies in a bit mask the events to enable for the feature session. Values are (can be ORed): <ul style="list-style-type: none"> D42_EVT_SOFTKEY Enables the asynchronous event TD42_SOFTKEYINPUT, which reports on the completion of softkey input. D42_EVT_ASYNC_CLOSEFEATSESSION Enables the asynchronous event TD42_ASYNC_CLOSEFEATSESSION, which reports on the close of a feature session.

■ Cautions

Only one feature session can be open on a channel at any time.

■ Example

```
void D42_Test_FeatureSession()
{
    char szExt[5];
    char *cpNull = NULL;
    char cpErrStr[MAX_PATH];
    int TermType;
    int eventMask;

    // extension fpr which we are opening a feature session
    sprintf(szExt, "1234");

    eventMask = D42_EVT_SOFTKEY | D42_EVT_ASYNC_CLOSEFEATSESSION;
    if (d42_openfeaturesession(G_d42chdev, szExt, &TermType, eventMask) == -1)
    {
        printf("Error d42_openfeaturesession()\n");
    }

    if (d42_writetodisplay(G_d42chdev, "FtrSs Open") == -1)
    {
        printf("Error d42_writetodisplay()\n");
    }

    if (d42_closefeaturesession(G_d42chdev) == -1)
    {
        printf("Error d42_closefeaturesession()\n");
    }
} // endof D42_Test_FeatureSession
```

d42_openfeaturesession()

opens a phone extension feature session

■ **Errors**

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FEATSESSION ALREADYOPEN	Attempt to open more than one feature session per channel
ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_UNSUPPORTED	Function not supported on this board
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

Name:	int d42_setparm(devh, parmnum, parmvalp)	
Inputs:	int devh	• board or channel descriptor
	int parmnum	• parameter name
	void *parmvalp	• pointer to an application buffer containing the parameter value
Returns:	ED42_NOERROR	• if success
	-1	• if error (see Errors list)
Includes:	D42LIB.H	
Mode:	synchronous	
Supports:	Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2	

■ Description

The **d42_setparm()** function sets a D/42-xx board or channel parameter. Depending on the parameter to be set, the value can be either a character string or an integer. A list of the board and channel parameters that can be set for **parmnum** can be found in *Table 4* and *Table 5*.

Parameter	Description
devh	specifies the valid board device or channel device descriptor obtained by a call to dx_open()
parmnum	specifies the define for the parameter that is to be updated with the value in the variable pointed to by parmvalp (see <i>Table 4</i> and <i>Table 5</i>)
parmvalp	pointer to the application buffer containing the parameter value

NOTE: Setting board parameters affects all the channels on the board, but setting channel parameters affects only the specified channel.

To set board parameters, the following requirements must be met:

- the board must be open
- all channels on the board must be closed

To set channel parameters, the following requirements must be met:

- the channel must be open
- the channel must be idle

This function will return a failure if:

- the board or channel descriptor is invalid
- any channels are open when setting board parameters
- when setting channel parameters, the channel is not open and idle
- a read-only parameter is specified
- the value of **parmnum** is invalid
- **parmnum** is not supported on the specified board
- an MF parameter is specified while MF detection is enabled

■ NEC PBX (Dialogic® D/42-NE2 Board)

The D4BD_MSGACCESSION and D4BD_MSGACCESSOFF parameter values must be character strings. The string cannot exceed 7 characters plus a null. Characters must be 0-9, #, and *.

The D4BD_RESETRINGCNT parameter is used to enable or disable the ring counter. On the Dialogic® D/42-NE2 Board, the ring counter is used to count the number of ring signals received from the PBX. Your application will answer a call after the ring counter reaches a certain value (that you set). The ring counter automatically resets to zero when the Dialogic® D/42-NE2 Board detects that the call is abandoned or there is sufficient delay between ring events (set to 8 seconds by default).

The NEAX 2400 IMS does not send call abandoned messages, therefore the ring counter will only be reset when the ring event delay is exceeded. Be aware that if a call is abandoned and a new call comes in before the 8-second delay, the ring counter will not be reset before the second call and will indicate the number of rings from both calls, possibly resulting in inaccurate data being reported to the application. For example, an application using a NEAX 2400 is set to answer a call after two rings. An incoming call is abandoned after the first ring and a new call comes in immediately following the abandoned call (less than 8 seconds apart). The application will answer the new call after the first ring because the ring counter was not reset after the first call was abandoned (it was incremented from one to two). This may result in inaccurate data being returned to the application, such as calling/called number ID or display data.

When using a Dialogic® D/42-NE2 Board with a NEAX 2400 IMS, the D4BD_RESETRINGCNT parameter must be set to 0 (which is the default). If enabled, the Dialogic® D/42-NE2 Board may intermittently reset the ring counter and your application will not answer calls reliably.

When using a Dialogic[®] D/42-NE2 Board with the NEAX 2000 IVS, you should set the D4BD_RESETRINGCNT parameter to 1. This will enable the Dialogic[®] D/42-NE2 Board to reset the ring counter when it detects that a call is abandoned, allowing your application to retrieve accurate called/calling number ID and display data.

■ Cautions

When setting a parameter, the user passes a pointer to a variable containing the new parameter value. This variable should be treated as an unsigned integer for all parameters except D4BD_MSGACCESSION and D4BD_MSGACCESSOFF. Both D4BD_MSGACCESSION and D4BD_MSGACCESSOFF should be treated as ASCII strings (char *). The application should cast the **parmvalp** parameter to a (void *) to avoid compiler warnings.

Table 4. Dialogic[®] D/42 Board Parameters for d42_getparm() and d42_setparm() Functions

Board Parameters	Description
Dialogic[®] D/42D-SX Board	
D4BD_SPMODE	Set speaker phone mode. Values: 0 - enable (default) 1 - disable
Dialogic[®] D/42D-SL Board	
D4BD_RINGON	Set ring on duration. Values: 0 - 1000 x 10 ms. (default: 400)
D4BD_RINGOFF	Set ring off duration. Values: 0 - 1000 x 10 ms. (default: 200)
Dialogic[®] D/42-NS Board	
(no parameters)	
Dialogic[®] D/42-NE2 Board (PBX only)	
D4BD_MSGACCESSION	Set message access code on. Values: string (default: **9)
D4BD_MSGACCESSOFF	Set message access code off. Values: string (default: ##9)
D4BD_RESETRINGCNT	Controls the automatic reset of the ring counter on the Dialogic [®] D/42-xx Board. Values: 0 - disable (default) 1 - enable

d42_setparm()

sets a D/42-xx board or channel parameter

Table 5. Dialogic® D/42 Channel Parameters for d42_getparm() and d42_setparm() Functions

Channel Parameters	Description
Dialogic® D/42D-SX Board	
(no parameters)	
Dialogic® D/42D-SL Board	
D4CH_PDNKEY	Define PDN key feature Values: 0 - 9 decimal (default: 0)
D4CH_XFERKEY	Define transfer feature key Values: 0 - 9 decimal (default: 1)
D4CH_SENDKEY	Define send message feature key Values: 0 - 9 decimal (default: 2)
D4CH_CANCELKEY	Define cancel message feature key Values: 0 - 9 decimal (default: 3)
D4CH_DNKEY	Define DN feature key Values: 0 - 9 decimal (default: 4)
D4CH_RELEASEKEY	Define release feature key Values: 0 - 9 decimal (default: 9)
Dialogic® D/42-NS Board	
D4CH_ASYNCCALLID	Control caller ID reporting through the asynchronous TD42_ASYNCCALLID event. Values: 0 - disable caller ID reporting (default) 1 - enable caller ID reporting
D4CH_ASYNCCHSTATUS	Control channel synchronization status reporting through the asynchronous TD42_ASYNCCHSTATUS event. Values: 0 - disable channel status reporting (default) 1 - enable channel status reporting Data values returned with the TD42_ASYNCCHSTATUS event are: D42_CH_STATUS_OFF - communication off D42_CH_STATUS_ON - communication on

Channel Parameters	Description
D4CH_ASYNCCLOSEFEATSESSION	Control feature session reporting through the TD42_ASYNCCLOSEFEATURESESSION event. Note that this parameter is not normally manipulated through the d42_setparm() function because it is built into the d42_openfeaturesession() and d42_closefeaturesession() functions, which can automatically enable and disable this parameter, respectively. This parameter requires an open feature session. Values: 1 - enable feature session reporting 0 - disable feature session reporting (default)
D4CH_SOFTKEYINPUT	Control softkey input reporting through the TD42_SOFTKEYINPUT event. Note that this parameter is not normally manipulated through the d42_setparm() function because it is built into the d42_openfeaturesession() and d42_closefeaturesession() functions, which can automatically enable and disable this parameter, respectively. Values: 0 - disable softkey input reporting (default) 1 - enable softkey input reporting
Dialogic® D/42-NE2 Board	
(no parameters)	

■ Example

```
void main(void)
{
    int          devh;
    int          rc = 0;
    unsigned int parmvalp = 1;

    /* 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_SX)
    {
        /* Set the Board Parameter To Enable Calling/Caller Id */
    }
}
```

d42_setparm()

sets a D/42-xx board or channel parameter

```
if ( (rc = d42_setparm(devh, D4BD_SPMODE, (void *)&parmvalp)) == -1)
{
    printf("Error d42_setparm(D4BD_SPMODE)\n");
    dx_close(devh);
    exit(-1);
} /* End d42_setparm */

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

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_FWREQFAILURE	Firmware error
ED42_BADPARM	Invalid value for parameter
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_INVALIDARG	Invalid argument passed to function

writes to the phone set display

d42_writetodisplay()

Name: int d42_writetodisplay(devh, bufferp)
Inputs: int devh • channel descriptor
 char *bufferp • pointer to a buffer containing
 ASCII character string data to be
 displayed
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-NS

■ Description

The **d42_writetodisplay()** function writes to the phone set display the ASCII string contents of **bufferp**. This function requires that a phone extension feature session be previously opened on the channel through the **d42_openfeaturesession()** function, which associates a telephone extension with a channel number and provides information about the display used by the phone set.

Parameter	Description
devh	specifies the valid channel device descriptor obtained by a call to dx_open() and on which a feature session is has been opened
bufferp	pointer to an application buffer containing a NULL terminated ASCII data string that is to be displayed on the telephone set display. The buffer size is determined by the maximum amount of data that can be sent to the telephone set display, which was returned by the d42_openfeaturesession() function, plus one NULL byte.

■ Cautions

Only one feature session can be open on a channel at any time.

■ Example

```

void D42_Test_FeatureSession()
{
    char szExt[5];
    char *cpNull = NULL;
    char cpErrStr[MAX_PATH];
    int TermType;
    int eventMask;

    // extension fpr which we are opening a feature session
    sprintf(szExt, "1234");

    eventMask = D42_EVT_SOFTKEY | D42_EVT_ASYNC_CLOSEFEATSESSION;
    if (d42_openfeaturesession(G_d42chdev, szExt, &TermType, eventMask) == -1)
    {
        printf("Error d42_openfeaturesession()\n");
    }

    if (d42_writetodisplay(G_d42chdev, "FtrSs Open") == -1)
    {
        printf("Error d42_writetodisplay()\n");
    }

    if (d42_closefeaturesession(G_d42chdev) == -1)
    {
        printf("Error d42_closefeaturesession()\n");
    }
} // endof D42_Test_FeatureSession

```

■ Errors

If this function returns -1 to indicate a failure, use **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to retrieve one of the following (most common) errors. For a complete list of error codes and definitions, refer to **Appendix C**.

ED42_NOFEATURE SESSION	Function requires an open feature session
ED42_FWREQFAILURE	Firmware error
ED42_BADDEVICE	Invalid or wrong device handle
ED42_UNKNOWNBOARD	Unknown Dialogic® D/42 board type
ED42_UNSUPPORTED	Function not supported on this board
ED42_INVALIDARG	Invalid argument passed to function
DXLIB_INVNRB	Internal voice library error

4. Programming Considerations

Information about using the Windows® version of the Dialogic® D/42 Runtime Library to perform PBX functions is included in this chapter. A general description of the PBX functions and considerations unique to each PBX system are included for the following:

- Opening a channel on a Dialogic® D/42-xx Board
- Accessing PBX features using dial strings
 - turn on/off message waiting indicators
 - dial programmable keys
 - call transfer
 - in-band/out-of-band signaling
- Disconnect supervision
- Converting existing Dialogic® D/4x Board applications into Dialogic® D/42 Board applications

4.1. Opening a Channel on a Dialogic® D/42-xx Board

■ Description

A Dialogic® D/42-xx Board begins to synchronize with the PBX immediately after the Dialogic® D/42 firmware (D4X.FWL) is downloaded to the board (assuming that the board is physically connected to the PBX). This process can take up to 60 seconds to complete. During this time period, the Dialogic® D/42-xx Board should not receive any calls from the PBX. Before any other functions are accessed, your application should ensure that the Dialogic® D/42-xx Board is synchronized by opening the board device and calling the **d42_brdstatus()** function. Failure to ensure that the connection is established (synchronized) may result in unpredictable results.

After synchronization is complete, the **dx_open()** function is used to open the channel by using a valid device name to identify the channel you wish to open.

■ **Example**

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

    /* open the channel */
    if ((devh = dx_open("dxxxB1C1",NULL)) == -1)
    {
        /* process error */
        exit(1);
    }

    /* Open Board 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);

    /* wait for 60 seconds for switch */
    Sleep(60000L);

    dx_close(devh);
    exit(0);
}
```

4.2. Accessing PBX Features on a PBX Using Dial Strings

You can access PBX features such as turning on and off a message waiting indicator, dialing programmable keys, and transferring calls, using dial strings in the **dx_dial()** function. **dx_dial()** is a Dialogic® D/4x Voice API function. For general information on how to use this function, see the *Dialogic® Voice API Library Reference*.

Input parameters for the **dx_dial()** function are defined as follows:

dx_dial()	
Name: int dx_dial(chdev, dialstrp, capp, mode)	
Inputs: int chdev	• channel descriptor
char *dialstrp	• pointer to ASCIIZ dial string
DX_CAP *capp	• pointer to Call Progress Analysis Parameter structure
unsigned short mode	• asynchronous/synchronous setting and call analysis flag

The dial string will accept escape sequences that are used to access PBX features. Acceptable ASCII characters for each dial string are the standard DTMF dialing and control characters described in the *Dialogic® Voice API Library Reference*, and the additional characters described in the following paragraphs.

The procedure for accessing a feature is as follows:

1. Set the hook state (on-hook or off-hook) required for dialing the feature dial string.
2. Use the appropriate dial string (e.g., <ESC>K).

NOTE: In some cases, a pause (“,”) may be needed after the entire dial string to give the switch enough time to respond to the command before issuing the next command.

4.2.1. Turn On the Message Waiting Indicator

■ Description

A dial string instructs the PBX to light the message waiting indicator on the specific extension. The dial string contains the following components:

<ESC>	the ASCII escape character (0x1B).
command	an ASCII character that identifies the “turn on message waiting indicator” feature.
,	pause
<extension>	the number of the extension whose message waiting indicator is to be lit.

The dial strings for specific PBXs are listed below.

NOTE: The pause in the dial string is sometimes needed to give the PBX time to activate the feature. The *command* character is case sensitive. Characters with the incorrect case will be ignored.

■ MITEL PBX (Dialogic® D/42D-SX Board)

,<extension>,<ESC>O

The Dialogic® D/42D-SX channel must be off-hook when dialing this string.

On a SUPERSET 4 telephone, this dial string activates the message indicator on the specified extension. On a non-SUPERSET 4 telephone, message waiting indication can be controlled if the phone is equipped with a message waiting indicator or if it has been assigned to a COS with the Audible Message Waiting (AMW) feature enabled.

4. Programming Considerations

■ Northern Telecom SL-1 (Dialogic® D/42D-SL Board)

<ESC>O,<extension>,<ESC>O

The Dialogic® D/42D-SL channel must be on-hook when dialing this string.

On an SL-1 phone, the message indicator is next to the Message Waiting Key (MWK), a feature key programmed on the telephone. On a non-SL-1 phone, the message waiting indicator can be turned on if the phone is equipped with a message waiting indicator or if it has been assigned to a COS with the Audible Message Waiting (AMW) feature enabled.

■ Northern Telecom Norstar (Dialogic® D/42-NS Board)

<ESC>O,<extension>,<ESC>O

The Dialogic® D/42-NS channel must be on-hook when dialing this string.

■ NEC KTS (Dialogic® D/42-NE2 Board)

<ESC>O,<extension>,<ESC>O

The Dialogic® D/42-NE2 channel must be on-hook when dialing this string.

■ NEC PBX (Dialogic® D/42-NE2 Board)

<ESC>O,<extension>,<ESC>O

The Dialogic® D/42-NE2 channel must be off-hook when dialing this string.

■ Example

```
#define ESC    0x1b
unsigned int devh;    /* channel descriptor */
char digstr[40];

int turn_on_mwl()
{
    /* set up dial string */
    switch (ATD4_CHTYPE(devh))
    {
        case TYP_SX:
            sprintf(digstr, "%c0,555,%c0",ESC);
            break;
        case TYP_SL:
        case TYP_NS:
        case TYP_NE2KTS:
            sprintf(digstr, "%c0,555,%c0",ESC,ESC);
            break;

        case TYP_NE2PBX:
            sprintf(digstr, "%c0,555,%c0",ESC,ESC);
            /* for the NEC PBX, channel must be off-hook before dialing */
            if(dx_sethook(devh, DX_OFFHOOK, EV_SYNC)==-1)
            {
                /* Process error */
                exit(1);
            }
            Sleep(1000); /* Allow 1 second for offhook to register */
            break;
    }
    /* turn on message waiting indicator on ext. 555 */
    if (dx_dial(devh,digstr,NULL,EV_SYNC) == -1)
    {
        printf("\nDial failed\n");
        exit(1);
    }

    if(ATD4_CHTYPE(devh) == TYP_NE2PBX)
    {
        if (dx_sethook(devh,DX_ONHOOK, EV_SYNC)==-1)
        {
            /* Process error */
            exit(1);
        }
    }

    return (0);
}
```

4. Programming Considerations

4.2.2. Turn Off the Message Indicator

■ Description

A dial string instructs the PBX to turn off the message waiting indicator on the specific extension. The dial string contains the following components:

<ESC>	the ASCII escape character (0x1B).
<i>command</i>	an ASCII character that identifies the “turn off message waiting indicator” feature.
,	pause
<extension>	the number of the extension whose message waiting indicator is to be turned off.

The dial strings for specific PBXs are listed below.

NOTE: The pause in the dial string is sometimes needed to give the PBX time to activate the feature. The *command* character is case sensitive. Characters with the incorrect case will be ignored.

■ MITEL PBX (Dialogic® D/42D-SX Board)

,<extension>,<ESC>F

The Dialogic® D/42D-SX channel must be off-hook when dialing this string.

■ Northern Telecom SL-1 (Dialogic® D/42D-SL Board)

<ESC>F,<extension>,<ESC>F

The Dialogic® D/42D-SL channel must be on-hook when dialing this string.

Dialogic® D/42 Series Software API Library Reference

■ **Northern Telecom Norstar (Dialogic® D/42-NS Board)**

<ESC>F,<extension>,<ESC>F

The Dialogic® D/42-NS channel must be on-hook when dialing this string. A message waiting indicator can only be disabled by the extension that enables it.

■ **NEC KTS (Dialogic® D/42-NE2 Board)**

<ESC>F,<extension>,<ESC>F

The Dialogic® D/42-NE2 channel must be on-hook when dialing this string.

■ **NEC PBX (Dialogic® D/42-NE2 Board)**

<ESC>F,<extension>,<ESC>F

The Dialogic® D/42-NE2 channel must be off-hook when dialing this string.

4. Programming Considerations

■ Example

```
#define ESC 0x1b
unsigned int devh; /* channel descriptor */
char digstr[40];

int turn_off_mwl()
{
    /* set up dial string */
    switch (ATD4_CHTYPE(devh))
    {
        case TYP_SX:
            sprintf(digstr, "%cF,555,%cF", ESC, ESC);
            break;
        case TYP_SL:
        case TYP_NS:
        case TYP_NE2KTS:
            sprintf(digstr, "%cF,555,%cF", ESC, ESC);
            break;

        case TYP_NE2PBX:
            sprintf(digstr, "%cF,555,%cF", ESC, ESC);
            /* for the NEC PBX, channel must be off-hook before dialing */
            if(dx_sethook(devh, DX_OFFHOOK, EV_SYNC)==-1)
            {
                /* Process error */
                exit(1);
            }
            Sleep(1000); /* Allow 1 second for offhook to register */
            break;
    }
    /* turn off message waiting indicator on ext. 555 */
    if (dx_dial(devh, digstr, NULL, EV_SYNC) == -1)
    {
        printf("\nDial failed\n");
        exit(1);
    }

    if(ATD4_CHTYPE(devh) == TYP_NE2PBX)
    {
        if (dx_sethook(devh, DX_ONHOOK, EV_SYNC)==-1)
        {
            /* Process error */
            exit(1);
        }
    }

    return (0);
}
```

4.2.3. Dial Programmable Keys

■ Description

The dial string `<ESC>K<key>`, enables the Dialogic® D/42-xx Boards to access features programmed into the programmable keys available to extensions on the PBX. The dial string contains the following components:

- `<ESC>` the ASCII escape character (0x1B)
- `K` identifies the Dial Programmable Key feature
- `<key>` indicates which programmable feature key to access
- `,` pause (optional)

NOTE: The pause in the dial string may be needed to give the PBX time to activate the feature. The “K” and `<key>` characters are case sensitive. Dial strings using a lower case “k” will be ignored. Use the correct case for the `<key>` characters to ensure the proper function is accessed.

■ MITEL PBX (Dialogic® D/42D-SX Board)

To access the dial string features on a MITEL SUPERSET 4 Phone, refer to *Figure 7* and use the direct key dialing sequences listed in *Table 6*. Also, refer to the *Dialogic® D/42 Series Boards User's Guide* for more information about programmable keys.

4. Programming Considerations

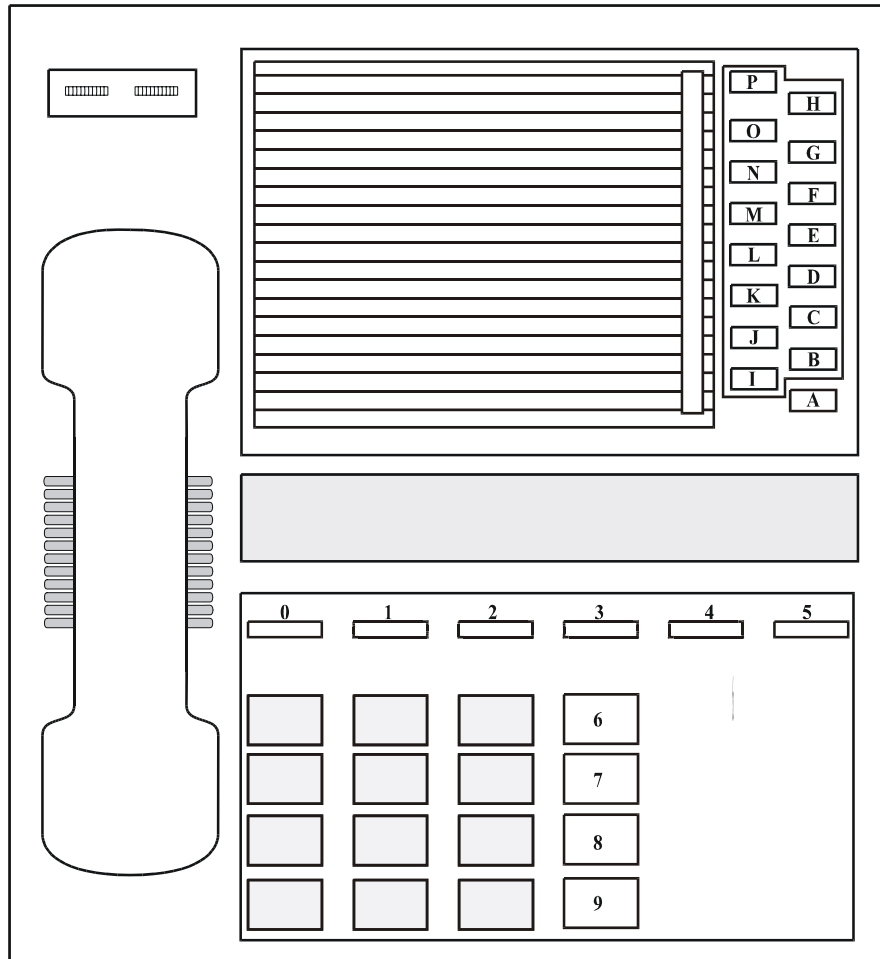


Figure 7. MITEL SUPERSET 4 Telephone

Table 6. MITEL Direct Key Dialing Sequences

Dial Code	Key Description
<ESC>K0	Soft Key 1
<ESC>K1	Soft Key 2
<ESC>K2	Soft Key 3
<ESC>K3	Soft Key 4
<ESC>K4	Soft Key 5
<ESC>K5	Soft Key 6
<ESC>K6	Feature Key 0 (display)
<ESC>K7	Feature Key 1 (select features)
<ESC>K8	Feature Key 2 (speaker on/off)
<ESC>K9	Feature Key 3 (mic. on/off)
<ESC>KA	Line Key 0 (hold)
<ESC>KB	Line Key 1 (prime line)
<ESC>KC	Line Key 2 (line or speed dial)
<ESC>KD	Line Key 3 (line or speed dial)
<ESC>KE	Line Key 4 (line or speed dial)
<ESC>KF	Line Key 5 (line or speed dial)
<ESC>KG	Line Key 6 (line or speed dial)
<ESC>KH	Line Key 7 (line or speed dial)
<ESC>KI	Line Key 8 (line or speed dial)
<ESC>KJ	Line Key 9 (line or speed dial)
<ESC>KK	Line Key 10 (line or speed dial)
<ESC>KL	Line Key 11 (line or speed dial)
<ESC>KM	Line Key 12 (line or speed dial)
<ESC>KN	Line Key 13 (line or speed dial)
<ESC>KO	Line Key 14 (line or speed dial)
<ESC>KP	Line Key 15 (line or speed dial)

4. Programming Considerations

■ Northern Telecom SL-1 (Dialogic® D/42D-SL Board)

To access the dial string features on a Northern Telecom Digit Display Phone, refer to *Figure 8* and use the direct key dialing sequences listed in *Table 7*. Also, refer to the *Dialogic® D/42 Series Boards User's Guide* for more information about programmable keys.

NOTE: For in-band and out-of-band signaling, the default is set to out-of-band. If you need to invoke in-band signaling, you must use the <ESC>DI dial string. The signaling will remain in-band until either the **dx_sethook()** function is used to go on hook, or the <ESC>DO dial string is used.

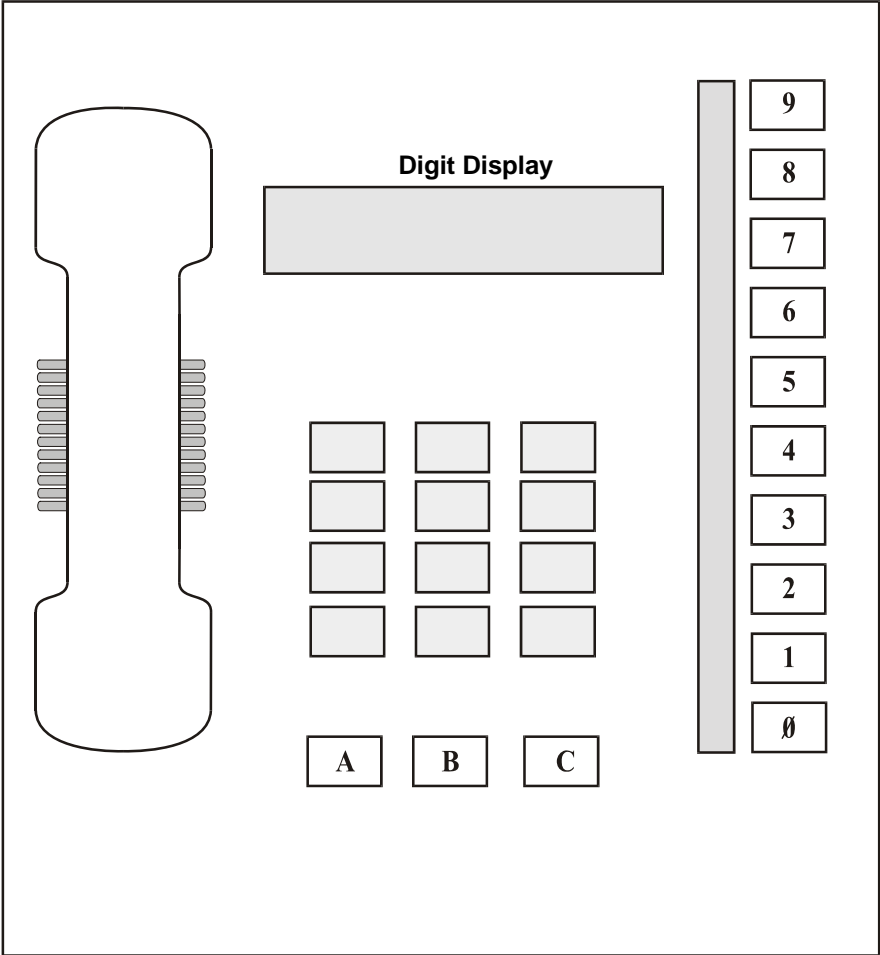


Figure 8. Northern Telecom Digit Display Telephone

4. Programming Considerations

Table 7. Northern Telecom SL-1 Direct Key Dialing Sequences

Dial Code	Key Description
<ESC>K0	ACD or SCR - Automatic Call Distribution - This line (port) will be used for call reception (ACD) or origination (SCR). Program this key as an ACD agent or as an SCR according to ACD requirements.
<ESC>K1	TRN - Call Transfer - Enables an extension to transfer calls to other extensions.
<ESC>K2	MIK - Message Indication - Enables an extension to turn on the message waiting indicator of another extension.
<ESC>K3	MCK - Message Cancellation - Enables an extension to turn off the message waiting indicator of another extension.
<ESC>K4	SCR - Single Call Ringing - This is the extension that the Dialogic® D/42D-SL channel originates calls on if ACD is enabled.
<ESC>K5	User programmable.
<ESC>K6	User programmable.
<ESC>K7	User programmable.
<ESC>K8	Handsfree - This enable/disables the speaker.
<ESC>K9	RLS - Release - This key allows an extension to release a call.
<ESC>KA	Volume up.
<ESC>KB	Volume down.
<ESC>KC	Hold.
<ESC>DI	Enable in-band DTMF signaling.
<ESC>DO	Enable out-of-band DTMF signaling.

■ Northern Telecom Norstar (Dialogic® D/42-NS Board)

To access the dial string features on a Northern Telecom Model 7310 telephone, refer to *Figure 9* and use the direct key dialing sequences listed in *Table 8*. Also, refer to the *Dialogic® D/42 Series Boards User's Guide* for more information about programmable keys.

NOTE: For in-band and out-of-band signaling, the default is set to out-of-band. If you need to invoke in-band signaling, you must use the <ESC>DI dial string. The signaling will remain in-band until either the **dx_sethook()** function is used to go on hook, or the <ESC>DO dial string is used.

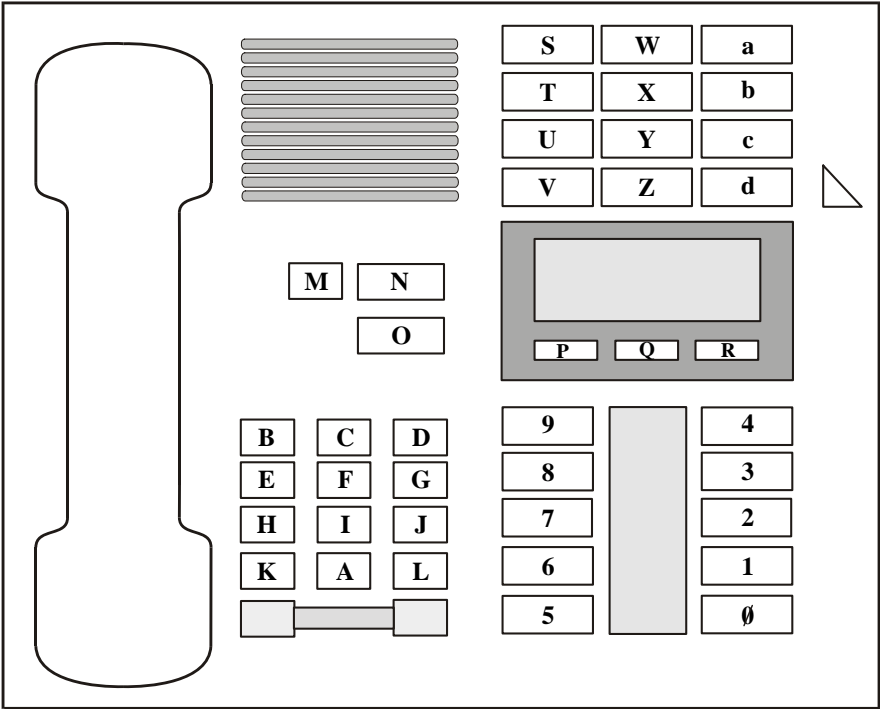


Figure 9. Northern Telecom Model 7310 Telephone

4. Programming Considerations

Table 8. Northern Telecom Norstar Direct Key Dialing Sequences

Dial Code	Key Description
<ESC>K0	Memory Button 00 - Handsfree/Mute
<ESC>K1	Memory Button 01 - Intercom
<ESC>K2	Memory Button 02 - Intercom
<ESC>K3	Memory Button 03 - Transfer
<ESC>K4	Memory Button 04
<ESC>K5	Memory Button 05
<ESC>K6	Memory Button 06
<ESC>K7	Memory Button 07
<ESC>K8	Memory Button 08
<ESC>K9	Memory Button 09
<ESC>KA	Dialpad 0
<ESC>KB	Dialpad 1
<ESC>KC	Dialpad 2
<ESC>KD	Dialpad 3
<ESC>KE	Dialpad 4
<ESC>KF	Dialpad 5
<ESC>KG	Dialpad 6
<ESC>KH	Dialpad 7
<ESC>KI	Dialpad 8
<ESC>KJ	Dialpad 9
<ESC>KK	Dialpad *
<ESC>KL	Dialpad #
<ESC>KM	Release
<ESC>KN	Feature
<ESC>KO	Hold
<ESC>KP	Display button 00

Dialogic® D/42 Series Software API Library Reference

Dial Code	Key Description
<ESC>KQ	Display button 01
<ESC>KR	Display button 02
<ESC>KS	Dual Memory button #0
<ESC>KT	Dual Memory button #1
<ESC>KU	Dual Memory button #2
<ESC>KV	Dual Memory button #3
<ESC>KW	Dual Memory button #4
<ESC>KX	Dual Memory button #5
<ESC>KY	Dual Memory button #6
<ESC>KZ	Dual Memory button #7
<ESC>Ka	Dual Memory button #8
<ESC>Kb	Dual Memory button #9
<ESC>Kc	Dual Memory button #10
<ESC>Kd	Dual Memory button #11
<ESC>Ke	Dual Memory button #12 (Shifted button #0)
<ESC>Kf	Dual Memory button #13 (Shifted button #1)
<ESC>Kg	Dual Memory button #14 (Shifted button #2)
<ESC>Kh	Dual Memory button #15 (Shifted button #3)
<ESC>Ki	Dual Memory button #16 (Shifted button #4)
<ESC>Kj	Dual Memory button #17 (Shifted button #5)
<ESC>Kk	Dual Memory button #18 (Shifted button #6)
<ESC>Kl	Dual Memory button #19 (Shifted button #7)
<ESC>Km	Dual Memory button #20 (Shifted button #8)
<ESC>Kn	Dual Memory button #21 (Shifted button #9)
<ESC>Ko	Dual Memory button #22 (Shifted button #10)
<ESC>Kp	Dual Memory button #23 (Shifted button #11)
<ESC>DI	Enable in-band DTMF signaling
<ESC>DO	Enable out-of-band DTMF signaling

4. Programming Considerations

■ NEC KTS/PBX (Dialogic® D/42-NE2 Board)

To access the dial string features on a Dterm Series III telephone, refer to *Figure 10* and use the direct key dialing sequences listed in *Table 9*. Also, refer to the *Dialogic® D/42 Series Boards User's Guide* for more information about programmable keys.

NOTE: For in-band and out-of-band signaling, the default is set to out-of-band. If you need to invoke in-band signaling, you must use the <ESC>DI dial string. The signaling will remain in-band until either the `dx_sethook()` function is used to go on hook, or the <ESC>DO dial string is used.

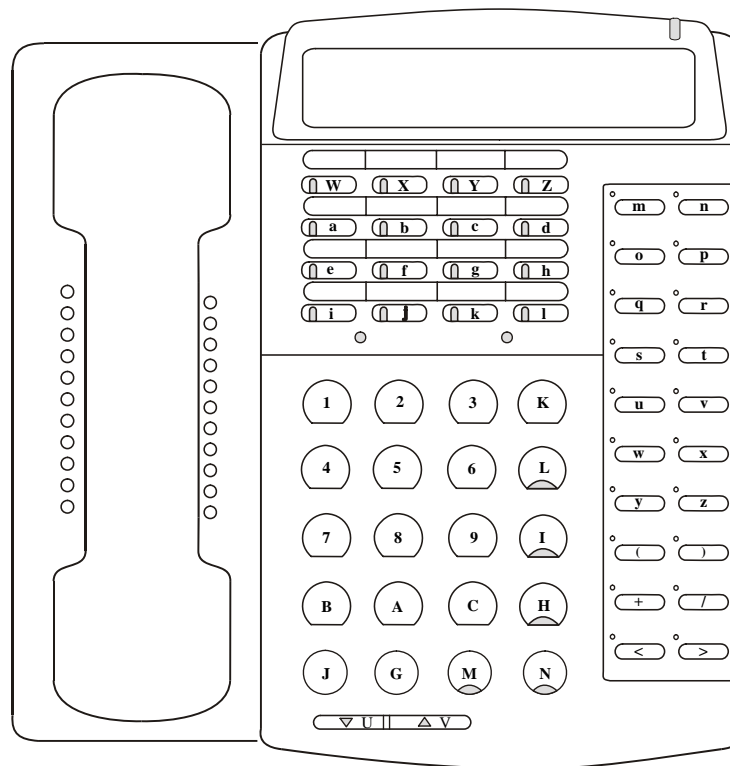


Figure 10. NEC Dterm III Telephone

Table 9. NEC KTS/PBX Direct Key Dialing Sequences

Dial Code	Key Description
<ESC>K1	Dialpad 1
<ESC>K2	Dialpad 2
<ESC>K3	Dialpad 3
<ESC>K4	Dialpad 4
<ESC>K5	Dialpad 5
<ESC>K6	Dialpad 6
<ESC>K7	Dialpad 7
<ESC>K8	Dialpad 8
<ESC>K9	Dialpad 9
<ESC>KA	Dialpad 0
<ESC>KB	Dialpad *
<ESC>KC	Dialpad #
<ESC>KF	Key release
<ESC>KG	Function key 0 - TRF
<ESC>KH	Function key 1 - LNR
<ESC>KI	Function key 2 - CNF
<ESC>KJ	Function key 3 - HOLD
<ESC>KK	Function key 4 - RECALL
<ESC>KL	Function key 5 - FNC
<ESC>KM	Function key 6 - ANS
<ESC>KN	Function key 7 - SPKR
<ESC>KU	Volume up
<ESC>KV	Volume down
<ESC>KW	Line key 1
<ESC>KX	Line key 2

4. Programming Considerations

Dial Code	Key Description
<ESC>KY	Line key 3
<ESC>KZ	Line key 4
<ESC>Ka	Line key 5
<ESC>Kb	Line key 6
<ESC>Kc	Line key 7
<ESC>Kd	Line key 8
<ESC>Ke	Line key 9
<ESC>Kf	Line key 10
<ESC>Kg	Line key 11
<ESC>Kh	Line key 12
<ESC>Ki	Line key 13
<ESC>Kj	Line key 14
<ESC>Kk	Line key 15
<ESC>Kl	Line key 16
<ESC>Km	Direct Station Select (DDS) key 1
<ESC>Kn	Direct Station Select (DDS) key 2
<ESC>Ko	Direct Station Select (DDS) key 3
<ESC>Kp	Direct Station Select (DDS) key 4
<ESC>Kq	Direct Station Select (DDS) key 5
<ESC>Kr	Direct Station Select (DDS) key 6
<ESC>Ks	Direct Station Select (DDS) key 7
<ESC>Kt	Direct Station Select (DDS) key 8
<ESC>Ku	Direct Station Select (DDS) key 9
<ESC>Kv	Direct Station Select (DDS) key 10
<ESC>Kw	Direct Station Select (DDS) key 11
<ESC>Kx	Direct Station Select (DDS) key 12
<ESC>Ky	Direct Station Select (DDS) key 13
<ESC>Kz	Direct Station Select (DDS) key 14

Dialogic® D/42 Series Software API Library Reference

Dial Code	Key Description
<ESC>K(Direct Station Select (DDS) key 15
<ESC>K)	Direct Station Select (DDS) key 16
<ESC>K+	Direct Station Select (DDS) key 17
<ESC>K/	Direct Station Select (DDS) key 18
<ESC>K<	Direct Station Select (DDS) key 19
<ESC>K>	Direct Station Select (DDS) key 20
<ESC>DI	Enable in-band DTMF signaling
<ESC>DO	Enable out-of-band DTMF signaling

4. Programming Considerations

■ Example

```
#define ESC    0x1b
int devh;    /* channel descriptor */
char digstr[40];

int set_spk()
{
    /* set up dial string to press Speaker key */

    switch (ATD4_CHTYPE(devh))
    {
        case TYP_SX:
            sprintf(digstr,"%cK8",ESC);
            break;
        case TYP_SL:
        case TYP_NS:
            sprintf(digstr,"%cK0",ESC);
            break;
        case TYP_NE2KTS:
        case TYP_NE2PBX:
            sprintf(digstr,"%cKN",ESC);
            break;
    }

    /* Program dial programmable key */
    if (dx_dial(devh,digstr,NULL, EV_SYNC))
    {
        printf("\nDial failed\n");
        exit(1);
    }

    return(0);
}
```

4.2.4. Transferring a Call

■ Description

The hook flash character (“&” by default) is used to initiate a transfer instead of an escape sequence as in the other feature dial strings. The hook flash is used because many PBX switches commonly use a hook flash as a transfer key. The following procedure is used by an application to transfer a call:

1. The channel must be off-hook and connected to an extension or trunk.
2. Use the following dial string to transfer the call to another extension:

```
&, <extension>
```

where “&” is the hook flash character, the comma (“,”) is a pause, and <extension> is the extension to which the call is being transferred.

3. Go on-hook to complete the transfer or dial a second hook flash to cancel the transfer.

The pause in the dial string is required. The pause gives the PBX time to activate the feature. Instead of a pause, you can use Enhanced Call Progress Analysis (ECPA) to detect a dial tone before dialing an extension. By using the control character “L” in the dial string, the **dx_dial()** function will wait for a dial tone before dialing. For example, to transfer to extension 555:

```
dx_initcallp(devh)
dx_dial(devh, "&L555",NULL, EV_SYNC)
```

You can also use Global Tone Detection (GTD) to detect a dial tone before dialing an extension. For example, to transfer to extension 555:

```
dx_dial(devh, "&",NULL, EV_SYNC)
*/add code here to wait for tone event /*
dx_dial (devh, "555" ,NULL, EV_SYNC)
```

Refer to the *Dialogic® Voice API Programming Guide* for more information about using ECPA and GTD.

4. Programming Considerations

■ Example

```
int devh;          /* channel descriptor */
char digstr[40];

int transfer_call()
{
    /* transfer the call */
    if (dx_dial(devh, "&,555", NULL, EV_SYNC) == -1)
    {
        printf("\nDial failed\n");
        exit (1);
    }

    /* set the channel onhook after the transfer */
    if (dx_sethook(devh, DX_ONHOOK, EV_SYNC) == -1)
    {
        /* process error */
        exit(1);
    }

    return (0);
}
```

4.2.5. In-Band/Out-of-Band Signaling

In-band signaling is a method used by analog (2500) telephones to communicate with PBXs (e.g., calling into a PBX and using DTMF to respond to voice prompts). In-band signals use the same band of frequencies as the voice signal. This method provides limited integration because there are no standards and different PBXs provide varying levels of control.

Out-of-band signaling is used by PBXs to communicate with their station sets or a CT computer. Out-of-band signals do not use the band of frequencies use by the voice signals. The PBX transmits data that can include information such as called/calling number ID. Because of its versatility, out-of-band signaling is the preferred method.

In-band signaling must be used when DTMF tones are required to communicate (e.g., connecting two voice mail systems through a CO using AMIS - Automated Messaging Interchange Specification). If out-of-band signaling is used, timing problems may occur because digit data (dial pad) sent from the station set (or Dialogic® D/42-xx Board) to the PBX are converted to DTMF and then sent to the CO.

Dialogic® D/42-xx Boards can be set to communicate using either in-band or out-of-band signaling; refer to *Table 10*.

NOTE: When using <ESC>DI and <ESC>DO to set the DTMF signaling method, the Dialogic® D/42-xx channel will return to its default state after a `dx_sethook()` function is called.

Table 10. Setting In-Band and Out-of-Band Signaling

Dialogic® D/42-xx Board	DTMF Signaling		Default Signaling
	In-Band	Out-of-Band	
D/42D-SX	N/A	N/A	In-band only
D/42D-SL	<ESC>DI	<ESC>DO	Out-of-band
D/42-NS	<ESC>DI	<ESC>DO	Out-of-band
D/42-NE2 (PBX)	<ESC>DI	<ESC>DO	Out-of-band
D/42-NE2 (KTS)	<ESC>DI	<ESC>DO	Out-of-band

4.3. Disconnect Supervision

■ Description

Disconnect supervision for Dialogic® D/42-xx Boards functions the same as other Dialogic® D/4x Boards. Refer to the *Dialogic® Voice API Programming Guide* for a description of using I/O terminations to perform disconnect supervision in your application.

As part of disconnect supervision, the Dialogic® D/42-xx Boards monitor communications with the PBX. If communication is lost with the PBX for 60 seconds, the Dialogic® D/42-xx firmware will force a loop current drop condition until communication is re-established.

■ Northern Telecom Norstar (Dialogic® D/42-NS Board)

Not all trunks allow trunk disconnect. Consult your Norstar Hardware Manual to determine if the trunk used allows trunk disconnect. If it does not provide

4. Programming Considerations

disconnect supervision, you will need to use another method to determine disconnect (e.g., GTD-Global Tone Detection).

■ NEC NEAX 2400 IMS and NEAX 2000 IVS PBX (Dialogic® D/42-NE2 Board)

The NEAX 2400 IMS and NEAX 2000 IVS do not support disconnect supervision (loop current drop). When the calling party hangs up, the PBXs issue a reorder tone. You will need to use another method (e.g., GTD-Global Tone Detection) to detect a disconnect.

4.4. Converting Existing Dialogic® D/4x Applications

■ Description

The Dialogic® D/42-xx Board and the Dialogic® D/4x Voice Board use the same Dialogic® D/4x Voice Library and supporting library. Therefore, only minor modifications are required to convert an existing Dialogic® D/4x application into a Dialogic® D/42 application. This conversion only includes new functions provided by the Dialogic® D/42 Runtime Library. Use the following guidelines to convert an existing Dialogic® D/4x application to an application that uses the Dialogic® D/42 Runtime Library:

NOTE: All Dialogic® D/42-xx applications must take into account the delay waiting for loop current to be detected that exists when opening the Dialogic® D/42-xx Board with the **dx_open()** library function.

- To convert an existing application without using the Dialogic® Unified API or called/calling number ID, use the **dx_clrdigbuf()** function immediately after the application receives a rings-received event to clear the called/calling number ID digits from the digit buffer. This will prevent the called/calling number ID from interfering with what the application expects to find in the digit buffer. Alternately, use the **dx_getdig()** function to retrieve the called/calling number ID and then discard the retrieved string. To access the other PBX features, the application must use the dial strings in the **dx_dial()** function using the format described in *Section 4.2. Accessing PBX Features on a PBX Using Dial Strings*.

Dialogic® D/42 Series Software API Library Reference

- To convert an existing application without using the Dialogic® Unified API but access called/calling number ID, use the **dx_getdig()** function to retrieve the called/calling number ID digits and place them in the digit buffer. To access the other PBX features, the application must use the dial strings in the **dx_dial()** function using the format described in *Section 4.2. Accessing PBX Features on a PBX Using Dial Strings*.
- To convert an existing application using the Dialogic® Unified API to retrieve the called/calling number ID, use the **dx_gtcalledid()** function to retrieve the called/calling number ID digits and place them in the application buffer. Refer to *Section 3. Dialogic® Unified API Function Reference*. To access the other PBX features, the application must use the dial strings in the **dx_dial()** function using the format described in *Section 4.2. Accessing PBX Features on a PBX Using Dial Strings*.

Appendix A

Dialogic® D/42 Series Software Quick Reference

ATD4_BDTYPE() *returns the D/42-xx board type*

Name: int ATD4_BDTYPE(devh)
Inputs: int devh • board descriptor
Returns: board type • returns board type information if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

ATD4_CHTYPE() *returns the D/42-xx channel type*

Name: int ATD4_CHTYPE(devh)
Inputs: int devh • channel descriptor
Returns: channel type • channel type information if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_brdstatus() *retrieves the current D/42-xx board status*

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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_chnstatus() *retrieves the current D/42-xx 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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_closefeaturesession() *closes an open feature session*

Name: int d42_closefeaturesession(devh)
Inputs: int devh • channel descriptor
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-NS

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

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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_flags() *retrieves current D/42D-SX LCD Features Display data*

Name: int d42_flags(devh, bufferp)
Inputs: int devh • channel descriptor
 char *bufferp • pointer to an application buffer. The
 buffer will contain the Features
 Display data.
Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SX

d42_getparm() *retrieves a D/42-xx channel or board parameter*

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: ED42_NOERROR • if success
 -1 • if error (see Errors list)
Includes: D42LIB.H
Mode: synchronous
Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_getver() *retrieves the D/42-xx 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: ED42_NOERROR • if success
-1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: all Dialogic® D/42 boards

d42_gtcallid() *retrieves the called/calling number ID*

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: ED42_NOERROR • if success
-1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_indicators() *retrieves the status of LCD/LED indicators*

Name: int d42_indicators(devh, bufferp)

Inputs: int devh • channel descriptor
char *bufferp • pointer to an application buffer
 containing the indicators data

Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_lcdprompt() *retrieves the current LCD prompt data of the D/42D-SX*

Name: int d42_lcdprompt(devh, bufferp)

Inputs: int devh • channel descriptor
char *bufferp • pointer to an application buffer
 containing Dialogic® D/42-SX LCD
 prompt data

Returns: ED42_NOERROR • if success
 -1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-SX

d42_openfeaturesession() *opens a phone extension feature session*

Name: int d42_openfeaturesession(devh, bufferp, termtype, evtmask)

Inputs: int devh • channel descriptor
char *bufferp • pointer to a buffer specifying a valid phone extension number in ASCII character string format
int *termtype • pointer to memory location that receives the type of phone display
int evtmask • specifies the events to enable for the feature session

Returns: ED42_NOERROR • if success
-1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-NS

d42_setparm() *sets a D/42-xx board or channel parameter*

Name: int d42_setparm(devh, parmnum, parmvalp)

Inputs: int devh • board or channel descriptor
int parmnum • parameter name
void *parmvalp • pointer to an application buffer containing the parameter value

Returns: ED42_NOERROR • if success
-1 • if error (see Errors list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-SL, D/42-SX, D/42-NS, D/42-NE2

d42_writetodisplay() *writes to the phone set display*

Name: int d42_writetodisplay(devh, bufferp)

Inputs: int devh • channel descriptor
char *bufferp • pointer to a buffer containing
 ASCII character string data to be
 displayed

Returns: ED42_NOERROR • if success
 -1 • if error (see Error list)

Includes: D42LIB.H

Mode: synchronous

Supports: Dialogic® D/42-NS

Appendix B

Dialogic® D/42 Series Software Demonstration Program

This appendix provides instructions for running the Dialogic® D/42 Series Software demonstration program for Windows®. This program demonstrates the Dialogic® Unified API functions using the Dialogic® D/42-NS and D/42-NE2 Boards. Basic operations performed by the Dialogic® D/42-xx Boards include:

- answer
- dial
- supervised/blind transfer
- play/record messages

Requirements

- 2 phones connected to a PBX
- 1 Dialogic® D/42-NS or D/42-NE2 Board connected to a PBX
- The PBX must be configured according to the *Dialogic® D/42 Series Boards User's Guide*.

Setup

Before running the Dialogic® D/42 Series Software demonstration program, perform the following procedures:

1. Connect one channel of your Dialogic® D/42-xx Board to an extension of the PBX.
2. Connect two telephones to two extensions of the PBX.
3. In necessary, start the Dialogic® D/42-xx Board using the Dialogic® Configuration Manager (DCM).

Documentation Conventions

The following conventions and terminology are used throughout the instructions contained in this section:

- Window titles are in italics.
- Menu items are in bold.
- The extension used to call the Dialogic® D/42-xx channel is called Phone A.
- The extension used receive the transfer is called Phone B.

Running the Demo

To run the Dialogic® D42 Series Software demonstration program:

1. Go to the Samples\d42 subdirectory under the Dialogic home directory.
2. Start the Dialogic® D/42 Series Software demonstration program by double-clicking on *D42DEMO.EXE*.
3. The *Dialogic® D/42 Demo* window will open - see *Figure 11*.

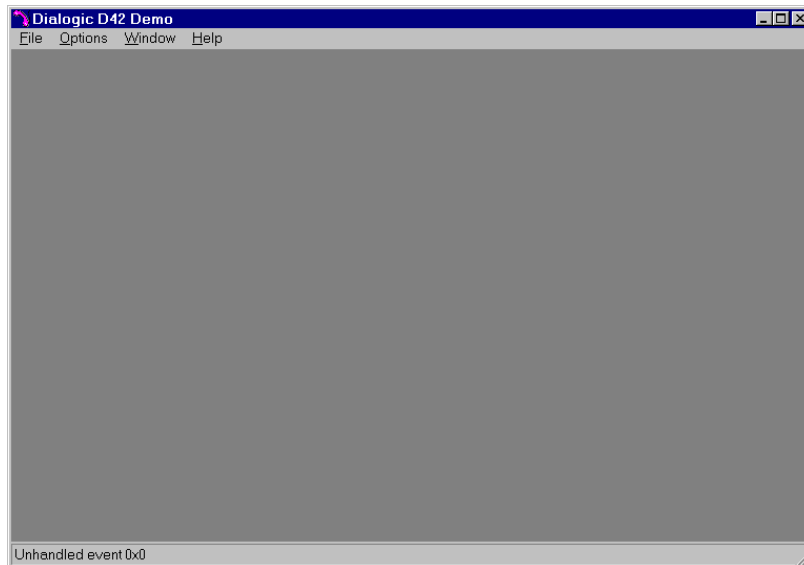


Figure 11. Dialogic® D42 Demo Window

4. From the **Options** menu, choose **Properties**. The *D42 Options* window is displayed - see *Figure 12*.

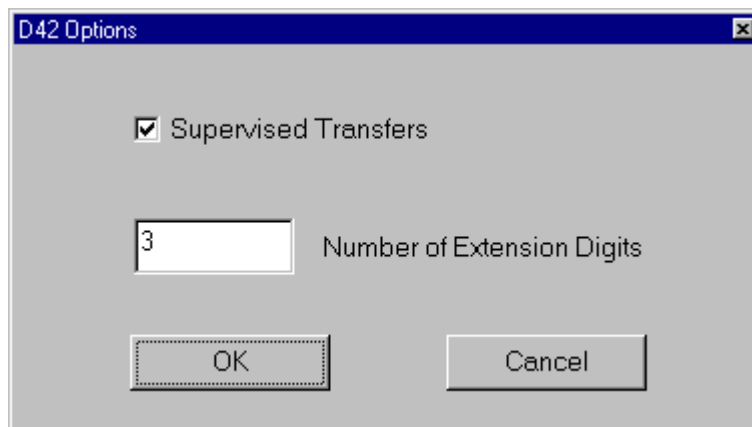


Figure 12. D42 Options Window

Dialogic® D/42 Series Software API Library Reference

5. Enter the number of extension digits that corresponds to your PBX configuration.
NOTE: If this option is not set correctly, the demo program will not perform transfers correctly.
6. If you want to use supervised transfers, check the Supervised Transfers box.
NOTE: In a supervised transfer, the demo program puts the incoming call (Phone A) on hold and attempts to establish a connection with Phone B before the transfer is completed.

In a non-supervised, or blind transfer, the incoming call (Phone A) is transferred immediately to Phone B. You will hear the ring signal, and the Dialogic® D/42-xx channel is ready to accept a new call (indicators are gray).
7. Press **OK** to close the *D42 Options* window.
8. From the **File** menu, choose **Open**. The *Select Your D42 Channel* window will be displayed - see *Figure 13*.

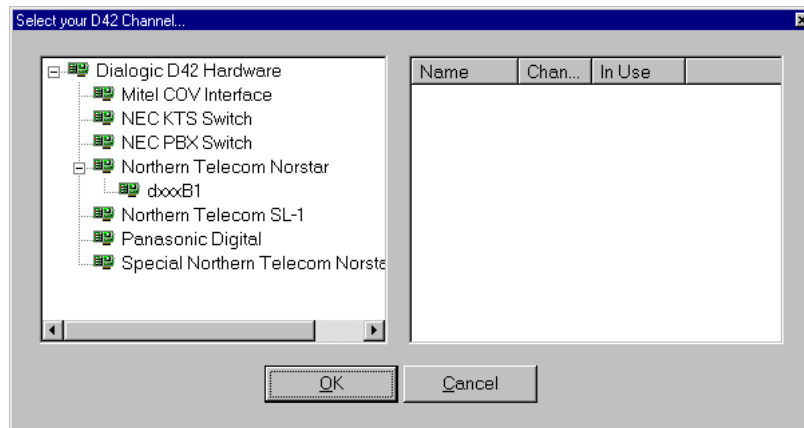


Figure 13. Select Your D/42 Channel

9. Choose a board listed below the applicable Dialogic® D/42 Series board (Example: dxxxB1 under the Northern Telecom Norstar). A

channel list will be displayed in the right-hand window - see *Figure 14*.

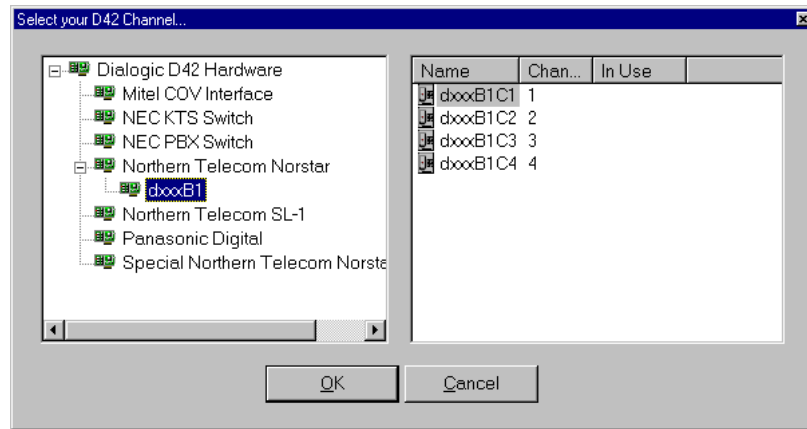


Figure 14. Select a D/42 Channel

10. Choose a channel connected to the PBX and press **OK** (Example: dxxxB1C1). A window will be displayed simulating the appropriate phone set for your PBX.

NOTE: All four channels are displayed, not just the channel connected to the PBX.

Figure 15 shows the Norstar M7310 phone set.

Figure 16 shows the NEC Dterm Series III phone set.

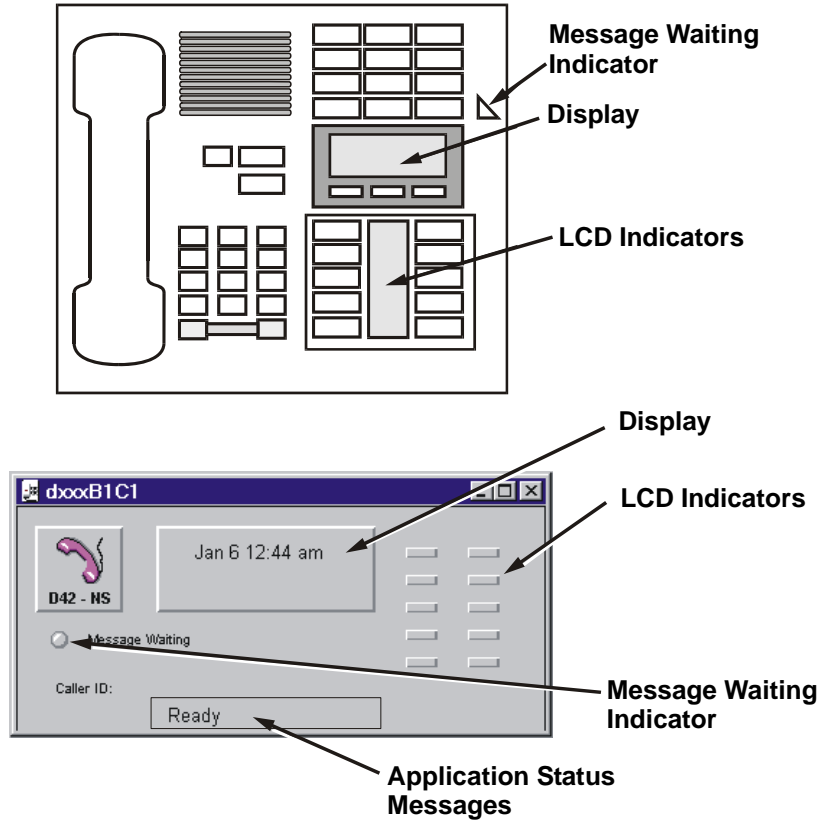


Figure 15. Northern Telecom M7310 Window

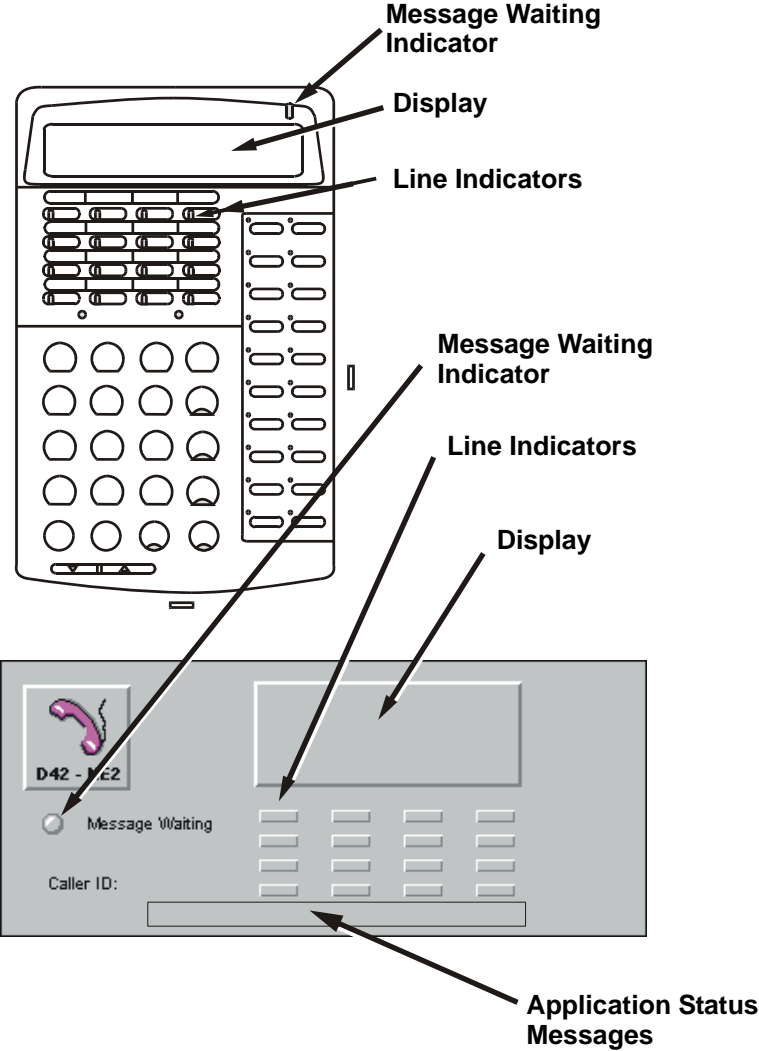


Figure 16. NEC Dterm Series III Window

Dialogic® D/42 Series Software API Library Reference

11. Observe that the date and time appear in the display. This verifies that the Dialogic® D/42-xx Board is communicating with the PBX. If the date and time do not appear, verify that the correct channel is selected and that the phone line is connected to the proper channel on the Dialogic® D/42-xx Board.
12. From Phone A, dial the number of the extension connected to the Dialogic® D/42-xx channel selected in step 10.
13. Listen to the greeting.
14. Enter the extension for Phone B.
NOTE: If you do not enter an extension within five seconds, the demo will play the message “Thank you for calling Dialogic Corporation” and hang up.
15. Phone B will ring. Answer the phone.
NOTE: If the demo is performing a supervised transfer, the transfer will complete immediately after voice is detected on the “transfer to” extension, or after 30 seconds of silence.

If performing a blind transfer, the transfer is completed immediately.
16. Observe the indicators, display, caller ID, and status areas in the demo window.
 - Indicators – refer to *Table 11*.
 - Display - shows information sent from the PBX.
 - Caller ID - shows the caller ID data sent by the PBX.
 - Message Waiting - turns on when a message is recorded.
 - Status Area - shows demo application status messages (e.g., dialing, ready, playing intro prompt, and message access code).

Table 11. Demo Indicator Definitions

Indicator Color	Definition *	
	M3710	Dterm Series III
gray	off	off
green	on	flutter
red	flash, Ihold, Uhold	wink (and all others)

* Refer to **d42_indicators()** in *Section 3* for a description of the indicators.

17. You may repeat steps 12 through 15 using different scenarios for Phone B (e.g., busy, no answer, forward).

NOTE: If you choose busy or no answer, you will be prompted to leave a message. Recording stops when silence is detected. When you leave a message, the Message Waiting indicator will turn on (red). Only one message is saved (any previously recorded message will be overwritten). The saved message will be deleted when the channel is closed.

To listen to the message, call back the Dialogic® D/42-xx extension and press the # key. The access number will be displayed in the status area.

18. To close the open channel, choose **Close** from the **File** menu.
19. To exit the program, choose **Exit** from the **File** menu.

Dialogic® D/42 Series Software API Library Reference

Appendix C

Error and Event Definitions

Table 12. List of Error Codes

Error Code Name	Description
ED42_BADDEVICE	Invalid or wrong device handle
ED42_BADPARAM	Invalid value for parameter
ED42_DLLINIT	Unable to initialize DLL
ED42_FEATSESSION ALREADYOPEN	Attempt to open more than one feature session per channel
ED42_FWREQFAILURE	Firmware error
ED42_INVALIDARG	Illegal argument in function
ED42_MAXCHAN	Maximum channel capacity reached
ED42_NOCOMM	No communication with switch
ED42_NOERROR	Operation completed
ED42_NOFEATURESESSION	Function requires an open feature session
ED42_NOTIDLE	Device is not idle
ED42_NOTIMP	Function is not implemented
ED42_RDFWVER	Error reading firmware version
ED42_SYSTEM	System level error
ED42_UNKNOWNBOARD	Unknown D/42 board type
ED42_UNSUPPORTED	Unsupported feature

Voice Errors

DXLIB_INVNRB	Internal voice library error
EDX_SYSTEM	System level error

Table 13. List of Event Codes

NOTE: See the **d42_setparm()** function in *Section 3* for the use and limitations of these events.

Event Code Name	Description
TD42_ASYNCCALLID	Caller ID information available
TD42_ASYNCCHSTATUS	Change in channel status
TD42_ASYNCCLOSEFEATSESSION	End of feature session
TD42_SOFTKEYINPUT	Softkey input received

Glossary

Adaptive Differential Pulse Code Modulation (ADPCM): A technique for reducing voice data storage requirements that is used by Dialogic in the voice boards. With ADPCM, rather than store the value of the speech sample (i.e., all 8-bits), only the change in the signal level between the present and the previous sample is stored. Fewer bits are needed to describe the change from one sample to the next because voice signals vary relatively slowly.

ADPCM: See Adaptive Differential Pulse Code Modulation.

analog: **1.** A method of telephony transmission in which the information from the source (for example, speech in a human conversation) is converted into an electrical signal that varies continuously over a range of amplitude values. **2.** Used to refer to applications that use loop start signaling instead of digital signaling.

answer supervision: A telephone system feature that returns a momentary drop in loop current when a connection has been established. When call progress analysis detects a transient loop current drop, it returns a connect event.

base address: A starting memory location (address) from which other addresses are referenced.

buffer: A block of memory or temporary storage device that holds data until it can be processed. It is used to compensate for the difference in the rate of flow of information (or time occurrence of events) when transmitting data from one device to another.

bus: An electronic path that allows communication between multiple points or devices in a system.

called/calling number ID: A PBX feature that identifies the number of the calling party to the extension that is called.

call progress analysis: A voice software feature that monitors the progress of an outbound call by detecting the different results that can occur after dialing, which allows you to process the call based on the

outcome. By using call progress analysis, you can determine whether the line is answered, the line rings but is not answered, the line is busy, or there is a problem in completing the call.

central office (CO): The telephone company (informally). A local telephone switching exchange.

channel: A voice I/O port on a voice board. **1.** When used in reference to a Dialogic board that is analog, an audio path, or the activity happening on that audio path (for example, in “the channel goes off-hook”). **2.** When used in reference to a Dialogic board that is digital, a data path, or the activity happening on that data path. **3.** When used in reference to a bus, an electrical circuit carrying control information and data.

class of service (COS): A defined group of features. Once an extension is assigned to a COS, the COS determines which features may be accessed by that extension.

computer telephony: The extension of computer-based intelligence and processing over the telephone network to a telephone. Lets you interact with computer databases or applications from a telephone and also enables computer-based applications to access the telephone network. Computer telephony makes computer-based information readily available over the world-wide telephone network from your telephone. Computer telephony technology incorporated into PCs supports applications such as: automatic call processing; automatic speech recognition; text-to-speech conversion for information-on-demand; call switching and conferencing; unified messaging that lets you access or transmit voice, fax, and E-mail messages from a single point; voice mail and voice messaging; fax systems including fax broadcasting, fax mailboxes, fax-on-demand, and fax gateways; transaction processing such as Audiotex and Pay-Per-Call information systems; call centers handling a large number of agents or telephone operators for processing requests for products, services, or information; etc.

configuration file: A file used to download voice hardware and software specifications to a voice board.

connect: A call progress analysis event indicating that the call has been answered. A connect can be established by cadence detection, loop current detection, or positive voice detection.

- D/4x:** A general term used to refer to Dialogic® 4-channel voice boards (e.g., Dialogic® D/41D, D/41E, and D/41ESC).
- D/xxx:** A general term used to refer to all models of Dialogic® voice boards.
- D40CHK:** The Dialogic® diagnostic program used to test voice boards for hardware problems.
- digit queue:** The location where digits are stored after they are detected. Digits are processed on a first-in, first-out basis, and can be accessed by the `getdtmfs()` function.
- disconnect supervision:** A feature that detects and acts on the change in electrical state from off-hook to on-hook.
- driver:** A software module that provides a defined interface between a program and the hardware. It directly controls the data transfer to and from I/O.
- DSP: 1.** Digital signal processor. A specialized microprocessor designed to perform speedy and complex operations with digital signals. **2.** Digital signal processing.
- DTMF:** Dual Tone Multi Frequency. **1.** A signaling method. **2.** The tone made by pressing a button on a push-button telephone. This tone is actually the combination of two tones, one high frequency and one low frequency.
- event: 1.** A specific activity that has occurred on a channel. The voice driver reports channel activity to the application program in the form of events, which allows the program to identify and respond to a specific occurrence on a channel. Events provide feedback on the progress and completion of functions and indicate the occurrence of other channel activities. Events are sometimes referred to in general as termination events, because most of them indicate the end of an operation. **2.** Any signal or condition that causes a state transition in a state machine, the majority of which are usually the physical events produced by the voice driver.
- Event Block (EVTBLK):** A data structure that is used as output for the `gtevtblk()` function. The `gtevtblk()` function removes an event from the queue and places it into an EVTBLK for use by the application program.

Dialogic® D/42 Series Software API Library Reference

FCC: Federal Communications Commission. The governing body for communications regulations within the U.S.

firmware: Software downloaded to a Dialogic board and stored in semi-permanent memory.

flash: A signal that consists of a momentary off-hook/on-hook/off-hook transition that is most often used by a voice board to alert a telephone switch. This signal usually initiates a call transfer. The **dial()** function can generate a hook flash by including the flash character in the dial string.

hook flash: See flash.

hook switch: The name given to the circuitry that controls the on-hook and off-hook state of the voice board telephone interface.

idle: The channel state when no multitasking function is in operation on the channel. The opposite of busy.

IRQ: Interrupt request. A signal sent to a central processing unit (CPU) to temporarily suspend normal processing and transfer control to an interrupt handling routine. Interrupts may be generated by conditions such as completion of an I/O process and detection of an event.

loop current: The current that flows through the circuit from the telephone switch to the voice board when the channel is off-hook.

loop start: In an analog environment, an electrical circuit consisting of two wires (or leads) called tip and ring, which are the two conductors of a telephone cable pair. The CO provides a voltage (called "talk battery" or just "battery") to power the line. When the circuit is complete, this voltage produces a current called loop current. The circuit provides a method of starting (seizing) a telephone line or trunk by sending a supervisory signal (going off-hook) to the CO.

multitasking functions: Functions that allow the voice software to perform concurrent operations. After being initiated, multitasking functions return control to the program so that during the time it takes the function to complete, the application program can perform other operations, such as initiating a function on another channel.

no answer: A call progress analysis event indicating that the call has not been answered. A no answer event is returned after a ring cadence has

been established by cadence detection and there was no break in the ring cadence for a specified number of times.

no ringback: A call progress analysis event indicating that there is a problem in completing the call. Cadence detection has determined that the signal is continuous silence or nonsilence.

nonsilence: Sound. Used when describing an audio cadence.

off-hook signal: A basic signal used on the telephone network that is produced when the line loop between the telephone set and the central office switch is closed and loop current flows, which also powers the telephone. This term is derived from the position of the old fashioned telephone set receiver in relation to the mounting hook provided for it.

on-hook signal: A basic signal used on the telephone network that is produced when the line loop between the telephone set and the central office (CO) switch is open and no loop current flows. This term is derived from the position of the old fashioned telephone set receiver in relation to the mounting hook provided for it.

ring detect: The act of sensing that an incoming call is present by determining that the telephone switch is providing a ringing signal to the voice board.

signaling: The transmission of electrical signals on the telephone network. The voice software supports the following signaling methods: DTMF, MF, R2 MF, Socotel, Global Tone Detection and Generation, and Dial Pulse Detection and Generation.

standard voice driver: See voice driver.

system events: Events in a state machine that are generated by relevant system signals, such as keyboard input, communications adapters, etc. These generally cause state changes for all channels rather than a specific channel.

talk off: The false tripping of DTMF receivers caused by speech.

telephone switch: A telephone company central office or a PBX (private branch exchange).

termination condition: A requirement that, when met, will cause a multitasking function to terminate. You can enable the termination conditions by setting parameters in the Read/Write Block and then

Dialogic® D/42 Series Software API Library Reference

passing the RWB as one of the function parameters. The termination conditions are monitored while the multitasking function is in progress. The function will continue to execute until one of the selected termination conditions has been met. When the function terminates, an event is produced, indicating which termination condition caused the function to terminate.

tone event: A tone-on or tone-off event that is produced by Global Tone Detection when a GTD tone is detected. A tone event can be accessed on the event queue by using the **gtevblk()** function, which provides the channel, event code, and GTD tone ID.

Unified API: This Dialogic® API provides a single set of basic, high-level calls that can be used for any supported switches and are sent directly to the switch through the Dialogic® D/42 Board, without additional hardware. Functioning as an extension to Dialogic® Voice API, the Dialogic® Unified API offers a single design model that is flexible enough to allow developers to take advantage of the advanced PBX features (such as called/calling number ID and ASCII display information).

voice demonstration programs: The programs that are included with the voice software and which demonstrate voice software features; provided in both source code and executable formats.

voice driver: The device driver for the voice boards; *D40DRV.EXE*. Executes as a terminate-and-stay-resident (TSR) program.

voice hardware diagnostic programs: The *D40CHK.EXE*, *D41ECHK.EXE*, and *UDD.EXE* programs allow you to test the features of the voice hardware.

voice library: A C language function library that can be accessed from assembly language programs or from applications written in a high-level language.

voice processing: Features of the voice software that provide the ability to record and play voice messages.

voice software: The Voice Development Package software, which includes the Voice Installation Programs and Files, Voice Demonstration Programs and Files, Voice Library (C Language Functions), and Voice Driver.

Glossary

voice store-and-forward: A term used to refer to a voice mail system. An early term for voice processing.

wink: A signal that consists of a momentary on-hook/off-hook/on-hook transition, which is used by the voice board as an acknowledgment signal. The **wink()** function generates an out-bound wink on a channel in response to an incoming call.

Dialogic® D/42 Series Software API Library Reference

Index

A

asynchronous, 14
ATD4_BDTYPE(), 18
ATD4_CHTYPE(), 20

C

Call Progress Analysis, 9, 10, 13, 75,
96, 121
Event, 122, 124, 125
call transfer, 14, 73, 75, 96, 124
called/calling number ID, 13, 14, 42, 43,
97, 100, 105, 121, 126

D

D/42 driver, 11, 14
d42_brdstatus(), 22
d42_chnstatus(), 24
d42_closefeaturesession(), 26
d42_display(), 28
d42_getflags(), 32
d42_getparm(), 36
d42_getver(), 39
d42_gtcalled(), 42
d42_indicators(), 45
d42_lcdprompt(), 57
d42_openfeaturesession(), 62
d42_setparm(), 65
d42_writetodisplay(), 71

D4BD_MSGACCESSOFF, 66, 67
D4BD_MSGACCESSION, 66, 67
D4BD_RESETRINGCNT, 67
D4BD_RINGOFF, 67
D4BD_RINGON, 67
D4BD_SPMODE, 67
D4CH_ASYNCCALLID, 68
D4CH_ASYNCCCHSTATUS, 68
D4CH_ASYNCCLOSEFEATURESESION, 69
D4CH_CANCELKEY, 68
D4CH_DNKEY, 68
D4CH_PDNKEY, 68
D4CH_RELEASEKEY, 68
D4CH_SENDKEY, 68
D4CH_SOFTKEYINPUT, 69
D4CH_XFERKEY, 68
dial programmable keys, 75, 82
Dterm Series III, 91
MITEL SUPERSET 4, 82
Northern Telecom Digit Display
Phone, 85
Northern Telecom M3710, 87
disconnect supervision
NEC NEAX 2400 IMS and NEAX
2000 IMS, 99
Northern Telecom Norstar, 98
dx_dial(), 99

Dialogic® D/42 Series Software API Library Reference

G

Global Tone Detection, 96

I

in-band signaling, 97

indicators, 45

- MITEL SUPERSET 4, 46
- NEC Dterm Series III, 51
- Northern Telecom Digit Display Phone, 48
- Northern Telecom Norstar M7310, 50

L

LCD prompts

- MITEL SUPERSET 4, 57

M

message waiting indicator, off

- MITEL SUPERSET 4, 79
- NEC Dterm Series III - KTS, 80
- NEC Dterm Series III - PBX, 80
- Northern Telecom Digit Display Phone, 79
- Northern Telecom M7310, 80

message waiting indicator, on

- MITEL SUPERSET 4, 76
- NEC Dterm Series III - KTS, 77
- NEC Dterm Series III - PBX, 77
- Northern Telecom Digit Display Phone, 77
- Northern Telecom M7310, 77

O

opening a D/42 channel, 73

out-of-band signaling, 97

- NEC, 91, 94
- Northern Telecom Norstar, 88, 90
- Northern Telecom SL-1, 85, 87

P

PBX configuration, 11

programmable key requirements
call transfer, 87

programmable key requirements
Automatic Call Distribution, 87

programmable key requirements
message indication, 87

programmable key requirements
message cancellation, 87

programmable key requirements
release, 87

Programmable key requirements
single call ringing, 87

S

standard voice library, 9, 11, 13, 14

synchronous, 14

T

transfer. *See* call transfer

U

Unified API, 11, 13, 14, 17, 126

V

voice and call processing, 9

voice hardware, 9