# PBX Integration board User's Guide

## for Linux and Windows

**Copyright © 2001 Dialogic Corporation**

05-1277-004

# COPYRIGHT NOTICE

Copyright 2001 Dialogic Corporation. All Rights Reserved.

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

All names, products, and services mentioned herein are the trademarks or registered trademarks of their respective organizations and are the sole property of their respective owners. DIALOGIC (including the Dialogic logo), DTI/124, and SpringBoard, are registered trademarks of Dialogic. A detailed trademark listing can be found at http://www.dialogic.com/legal.htm.

Publication Date: September, 2001

Part Number: 05-1277-004

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

**Technical Support**
http://support.dialogic.com

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

# Table of Contents

# List of Tables

# List of Figures

# 1. How To Use This Manual

## 1.1. Audience

This manual is addressed to programmers and engineers who are computer-literate and are interested in using Dialogic PBX Integration boards and the Dialogic Unified API to develop a computer telephony application for use on a PBX. 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.

## 1.2. Product Terminology

This manual includes information about using your *Private Branch eXchange* (PBX) or *Key Telephone System* (KTS) with a Dialogic PBX Integration board. A PBX is a privately owned, mini version of a telephone company's *central office* (CO) switch. For businesses, the key advantage to owning a PBX is the efficiency and cost savings of sharing a specific number of telephone lines among a large group of users. Grouped with PBXs are KTSs, which are generally smaller versions of a PBX that provides direct access to CO telephone lines. For simplicity, the term PBX will be used to denote both a PBX and KTS.
In the PBX environment a *line* from the CO is called a *trunk* and a phone is called a *line*, *extension*, or *station*.

## 1.3. PBX Models Covered in this Manual

This manual includes support for the following PBXs and KTSs and associated telephones:

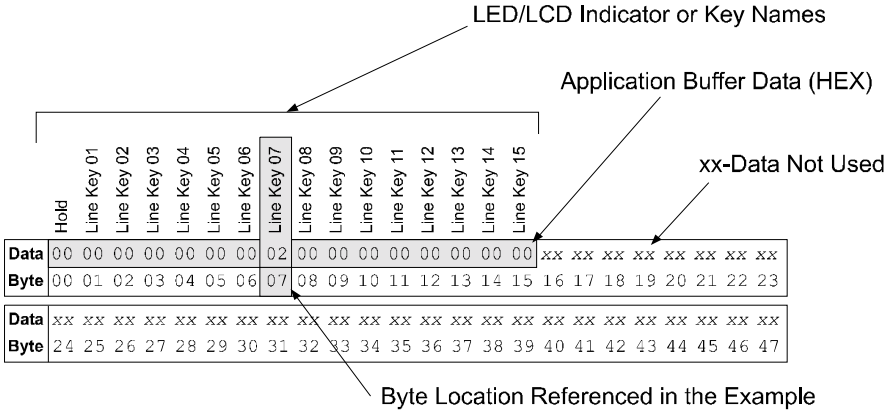
| Manufacturer | PBX Hardware | Telephone Emulations |
|---|---|---|
| Lucent | Definity System 75/85 | 7434 (4-wire) |
| | Definity System G3 Ver. 4 | 8434 (2-wire) |
| Siemens | ROLM 9005 and 9006 | ROLMphone 400 (RP400) |
| | Hicom 150E and 300E | Optiset E |
| Mitel | SX-50 | Superset 420 (DNIC) |

| | | |
|---|---|---|
| | SX-200ML<br>SX-2000 | Superset 430 (DNIC) |
| Nortel | Norstar | M7324 |
| | Meridian 1 | M2616 |

## 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 **d42_display( )**.
- Names of defines or equates are shown in uppercase, such as T_DTMF.
- File names are italicized and in uppercase, such as *D42DRV.EXE*.
- Examples included in this manual show data that is stored in an application buffer. The contents of a buffer is illustrated as follows:




Application buffers are typically 48 bytes long (plus a null). The actual data (in HEX) is shown in the gray area. The byte(s) referenced in the example is shown in boldface.

## 1.5. Voice Hardware Covered by This Manual

The PBX Integration board voice hardware is designed to provide a set of cost-effective tools for implementing computerized voice and call processing applications for PBXs. It provides the basic voice and call processing capabilities of D/4x voice hardware and adds hardware and firmware that eases integration with supported PBXs. The PBX Integration board hardware also provides access to PBX functions not normally available. Refer to the *Voice Software Reference* for your operating system for more information on voice and call processing. The PBX Integration hardware models covered by this manual include the following:

**D/42JCT-U –** an 4-channel voice board with station interfaces for connecting directly to a number of different PBXs.

**D/82JCT-U –** an 8-channel voice board with station interfaces for connecting directly to a number of different PBXs.

### 1.5.1. Voice Hardware Model Names

Model names for Dialogic voice boards are based upon the following pattern:
D/NNNoRBB-TT-VVV
where:

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

and 120)

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

## 1.6. When To Use This Manual

This *PBX Integration User's Guide* contains information for configuring and using specific PBX hardware for use with PBX Integration boards. For information about installing hardware, refer to the *PBX Integration Quick Install Card* provided with your board. For information about installing PBX Integration software, refer to the *System Release Software Installation Reference* for your particular operating system.

## 1.7. How This Manual Is Organized

**Chapter 1 – How To Use This Manual** describes the *PBX Integration User's Guide*.

**Chapter 2 – Introduction to PBXs and KTSs** provides a brief description of Private Branch Exchanges (PBXs), Key Telephone Systems (KTSs) and hybrid systems.

**Chapter 3 – PBX Integration Overview** provides information about the voice and PBX-specific features supported by the PBX Integration products and a description of the Unified API™.

**Chapter 4 – PBX Configuration and Integration** contains general descriptions, capabilities, switch requirements, and direct key dial sequences of all supported PBXs.

**Appendix A – PBX Integration Specifications** contains a data sheet for the PBX Integration circuit cards.

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

**Index** contains an alphabetical index of features and topics.

# 2. Introduction to PBXs and KTSs

A PBX, or *private branch exchange*, is a telephone system that is usually installed in a business. It provides service among many extensions within the business as well as outside lines. Typically, PBXs are used when a large number of extensions are needed. A PBX can be thought of as a mini version of a telephone company's *central office* (CO) switch. Key advantages to owning a PBX are:

- increased efficiency and cost savings because a specific number of CO telephone lines are shared among a large group of users
- special PBXs features.

Grouped with PBXs are *key telephone systems* (KTSs). A KTS is generally a smaller version of a PBX that also provides direct access to outside telephone lines (trunks). When you press a "line" key on a KTS you immediately hear a dial tone from the central office. In contrast, on a PBX system, you have to dial a digit, usually "9", to get the dial tone from the central office. Typically, KTSs are used when less than 50 extensions are needed. Advantages of having a KTS are that anyone in your office can answer an incoming call simply by pressing the correct line button and KTSs usually cost less than PBXs.

Systems have been developed that combine PBX and KTS features. These hybrid systems typically serve up to 100 users and contain some features found only in PBXs (the ability to use single line phones) and features typically found in KTSs (hands free announcing and answerback). An example of a hybrid system is the NEC Electra Professional which can connect to a maximum of 64 outside lines and 96 extensions. Some features include least cost routing, call forwarding, call hold, automated attendant, and caller ID.

For simplicity, throughout this manual the term PBX will be used to denote a PBX, KTS, or hybrid system.

Most PBX systems are digital. In a digital system, both the *voice signals* and *control information* transmitted between *station sets* within the PBX are sent as binary data. Analog voice signals received from outside the PBX (usually a CO) are converted to digital voice data and sent through the PBX. Digital voice data may be sent outside the PBX if outside networks also use digital circuits; however, they are usually converted back to analog voice signals.

PBXs use control information to instruct their station sets to perform specific functions such as setting the message waiting indicator and call transfer. This control information is sent using proprietary digital *protocols*. A protocol is a set of rules relating to the format and timing of data transmissions. These protocols

not only contain control information, but also "message" data that can be used to significantly enhance *computer telephony* (CT) applications that use PBX call control elements such as called/calling number ID.
The term "computer telephony" refers to the ability to interact with computer databases or applications from a telephone. Computer telephony technology 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

PBXs can communicate with their station sets using in-band or out-of-band signaling. In-band signaling is a method used by analog (2500) telephones (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 send and receive data from station sets or a CT computer. This data that can include information such as called/calling number ID. Out-of-band signals do not use the band of frequencies use by the voice signals. They can be transmitted using the same wires as the telephone set or separate wires (e.g., RS-232). Because of its versatility, out-of-band signaling is the preferred method.
CT equipment comprises a PC containing a Dialogic PBX Integration board and a software application. A PBX Integration board, together with the Dialogic Unified

API™, make it easier to create applications that are tightly integrated with a PBX and take advantage of call control elements.
Below is a list of popular PBX features and functions supported by a Dialogic PBX Integration board. KTSs and hybrid systems may support only some of these features.

- supervised call transfer
- blind call transfer
- caller ID
- called party ID
- positive disconnect supervision
- in-band signaling
- out-of-band signaling
- read display messages
- "press" programmable keys
- message waiting indication

## 2.1. Supervised Call Transfer

A supervised transfer is a method of transferring an incoming call to another extension, making use of call progress results (i.e., answered, busy, and ring no answer). This type of transfer is equivalent to the following manual operations:

1) answer a call
2) place the caller on hold
3) press the transfer key (hook flash)
4) dial the destination number
5) if the destination party answers, hang up (the transfer is complete)
6) if the destination party does not answer, switch back to the caller and provide choices to leave voice mail, select another extension, or hang up.

While a supervised transfer can be implemented without a PBX Integration board (using hook flash), the availability and ease of implementation is inconsistent. By using a PBX Integration board and the appropriate dial string, you can initiate a

transfer the same way for all supported switches. Also, by incorporating call progress analysis, you can offer consistent, high-performance call transfer features in your applications. For example, if during the transfer the application detects a busy signal, the call is automatically sent to a mailbox.
In a supervised transfer, an incoming call answered by a channel on a PBX Integration will only be transferred after a PBX Integration board establishes a connection with another station (the call is not released to the PBX). If the extension is busy or does not answer, the PBX Integration board reconnects to original call.

## 2.2. Blind Call Transfer

A blind transfer is initiated the same way as a supervised transfer. However, after dialing the destination number, the extension performing the transfer hangs up and does not wait to determine the outcome of the call. The call is released to the PBX. Blind transfers are used in most voice mail applications. A blind call transfer is equivalent to the following manual operations:

1) answer a call
2) put the call on hold
3) press the transfer key
4) dial the destination number
5) hang up.

The call is immediately sent to the new extension. It is up to the PBX to determine what to do if the transferred call is not answered (because of busy or no answer). Usually, if a transferred call is not answered it is routed back to the voice mail system, and eventually to the operator (or an automated attendant).
The advantage of a blind transfer is that the immediate release to the PBX frees the voice processing resources to handle new calls rather than being used to perform call progress. The only potential drawback of a blind transfer is when phone traffic is heavy, in which case the application may need to handle a call overflow condition.
An application can perform blind transfers without special integration tools. However, by using a PBX Integration board and the Unified API to access the called number ID from the PBX, the application can differentiate between:

- a new call coming in that needs to be processed: "Hello and thank you for calling Dialogic Corporation."

- a call that was transferred at least once already and is being routed by the PBX into voice mail: “You’ve reached the desk of Marcia Jones in Engineering, please leave a message.”

If the call was transferred, the application can use the called number ID to send the call directly into the appropriate voice mail box, allowing the caller to leave a message without having to navigate through a series of menus for a second or third time.

## 2.3. Caller ID

Caller ID is the phone number that identifies the person who is placing the call. These digits are typically transmitted at the beginning of a call, usually between the first and second ring.

While telephone companies are beginning to sell a caller ID service to residential customers, the scope of this commercially available caller ID is different from the caller ID feature available with many PBXs. The caller ID from the telephone company is often referred to as automatic number identification (ANI) and identifies callers whose numbers are assigned by the telephone company. Caller ID from within the PBX identifies callers whose telephone extensions are assigned through the PBX (referred to in this document as calling number ID).

Calling number ID from within the PBX system has powerful business applications. For example, a voice mail application may use calling number ID to let users reach individual mailboxes without having to dial extra digits. Other applications may use calling number ID for screening phone calls, allowing employees to respond to urgent calls first, as well as for automatic voice message reply, without making users redial the caller’s extension. Calling number ID is useful whenever you need to know who is calling and from where they are calling.

## 2.4. Called Number ID

Called number ID is also a feature provided within a PBX system and is usually combined with the calling number ID. Called number ID is the phone number of the extension being called. When a call is from outside the PBX, it is the number of the trunk receiving the call. The called/calling number ID remains the same when a call is routed through the PBX system.

For example, when a call has been routed through the PBX because the first intended extension was not answered or busy, the final destination answering the call can determine the extension that called plus the extension that was originally called.

Called number ID can also be used by an application to automatically direct a call to an appropriate extension or group of extensions based on the number called (generally the last four digits).
For example, an application may provide specific information about four different programs through an interactive voice response (IVR) system. Depending on the phone number being called, the application can route the caller directly to the desired program:
Program A: 555-1202 (trunk 01)
Program B: 555-1203 (trunk 02)
Program C: 555-1205 (trunk 03)
Program D: 555-1200 (trunk 04)
Using a PBX Integration board and the Unified API, an application can read the called number ID (the trunk line) and route the call depending on which extension receive the call. If the call is received on trunk line 01 it will be routed to the extension for Program A. Without access to the called number ID information, callers would need to listen to a long list of prompts to obtain the four digit extension code to access Program A.

## 2.5. Positive Disconnect Supervision

In any PBX phone system, it is important to accurately detect when an outside caller has "hung up" the phone. This capability allows the PBX to also hang up, completing the disconnection. Once the call is fully terminated, not only is the phone line available for other calls, but more importantly the phone company's billing charge for that call ends. One common way in which a phone or PBX manages call termination is positive disconnect supervision.
In a typical external call scenario (where a call is placed through a CO, not between extensions of the PBX), the CO detects when the caller hangs up and then sends a disconnect signal (loop current drop) to the PBX. The PBX is responsible for detecting and handling the disconnect signal from the CO.
After receiving a disconnect signal from the CO, the PBX may:

- terminate the outside call immediately and send a disconnect message to the called extension
- send a disconnect message to the called extension and wait for the called extension to hang up before formally terminating the call

In both cases, a disconnect message, not a loop current drop, is sent to the called extension. Standard analog voice boards cannot interpret disconnect messages because these messages are usually digital. PBX Integration boards can, however,

detect disconnect messages and send a disconnect event to an application where it is used by the Dialogic standard voice programming mechanisms for handling call termination.
When a call is placed between extensions of the PBX, a disconnect message, not a loop current drop, is also used to indicate when a caller hangs up. In this scenario, the application has no way of knowing when the caller has hung up so it can receive another call. PBX Integration boards can detect the disconnect message and send a disconnect event to an application.
Not all PBXs have positive disconnect supervision. Refer to the documentation for your PBX to determine if your PBX provides positive disconnect supervision.

## 2.6. In-Band Signaling

PBXs may use a method called in-band signaling to control their station sets. In-band signals use the same band of frequencies as the audio signal; this is usually accomplished with touch-tone signals. This method provides a limited amount of integration because there are no standards and different PBXs provide varying levels of control. Call progress tones that even similar models send can vary. This means that applications, even on identical PBXs, have to be tuned with each installation.
An example of in-band signaling is transferring a call using the flashhook method There is no data (e.g., caller ID information) passed along when the call is transferred.

## 2.7. Out-Of-Band Signaling

Many PBXs use a method called out-of-band signaling to control their station sets. Out-of-band signals do not use the band of frequencies used by the voice signals. These PBXs transmit control signals and data that can include information such as called/calling number ID. Because of its versatility, out-of-band signaling is the preferred method.

## 2.8. Read Display Messages

Most PBX station sets have an LCD or LED screen that can display messages. The type of information that is displayed varies with the PBX manufacturer and the programming capabilities of the switch. Typical information includes: calling/ called number ID from within the switch, ANI digits from the CO, hook state, time and length of call, name assigned to the extension, and message waiting notification. With a PBX Integration board, this information can be easily passed

"unprocessed" to the application. This means that the same data that is sent to the display is captured by a PBX Integration board.
By capturing the same display messages that a phone set receives, an application can "see" and "record" the display information. This display information (in ASCII format) is especially useful in CT applications because it enables an application to know exactly what state the extension connected to the PBX Integration board is in. Applications used with a PBX that provides ANI digits may process the display data and use those digits to access related database information.
For applications using a PBX Integration board to program the Nortel Norstar, display data is indispensable. Because the programming menus and key functions change at different levels within the PBX software, the only way to know the current menu options is by having display text available.

## 2.9. "Pressing" Keys

Station sets typically have Feature Keys that can be programmed to perform specific functions (e.g., transfer, hold, speaker phone, speed dial, or connect to trunk lines). Since a PBX Integration board emulates a station set, applications can "press" these keys. If the station set can be used to program Feature Keys, an application can also control the assignment of programmable keys. For instance, if a specific key must be assigned to the transfer function, you can include a sequence of "pressing" keys at the start of the application to ensure that the environment has been set correctly.

## 2.10. Message Waiting Indication

Most PBX systems turn on message waiting lights on station set phones when messages arrive, and clear the light after messages are retrieved. These tasks can be handled manually, by an attendant, or be automated through a voice mail application. Using a PBX Integration board, an application can also control the state of message waiting indications on other station sets (if this feature is available on your PBX).

## 2.11. Automated Attendant

An auto attendant is a device connected to a PBX that answers incoming calls. After answering, it may perform functions such as playing a greeting, asking the caller to press a button, or routing the call to the proper destination.

# 3. PBX Integration Overview

The PBX Integration board combines the voice and fax features available in the D/4x product line with the ability to access enhanced PBX features on several different PBXs. The voice features include:

- play and record voice messages
- dial and recognize DTMF digits
- detect and answer incoming call
- call progress analysis
- send and receive faxes

The PBX specific features include:

- retrieve Called/Calling number ID
- retrieve LCD/LED prompts and indicators
- read displays
- accessing PBX features using dial strings
- disconnect supervision

## 3.1. Voice Features Supported

The PBX Integration board uses a dual-processor architecture comprising a DSP (Digital Signal Processor) and a general purpose microprocessor to handle all voice processing functions. This dual processor approach off loads many low-level decision making tasks from the host computer.

When a PBX Integration system is initialized, firmware is downloaded from the host PC to the firmware RAM and DSP memory on the PBX Integration board. This downloadable firmware gives the board all of its intelligence and enables easy feature enhancement and upgrades. Based on this, the PBX Integration board can perform the following operations on incoming calls:

- automatically control the volume of the incoming audio signal
- record and compress the incoming audio voice signal. Sampling rates and coding methods are selectable on a channel by channel basis
- detect the presence of tones - DTMF, MF, or an application defined signal or dual tone
- perform call progress analysis (CPA) to determine the state of an incoming call.

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

For outbound calls, the PBX Integration board can perform the following:

- play stored compressed audio files
- adjust the volume and speed of playback upon application or user request
- generate tones - DTMF, MF, or an application defined signal or dual tone.

The PBX Integration board is basically a D/41D board with specialized PBX circuitry replacing the analog front end. The PBX Integration board performs features available on a D/41D and D/42-xx, as well as emulating phones connected to a PBX. With the current D/42-xx PBX Integration boards, it is necessary to choose a particular board depending on which PBX you plan to use. With the PBX Integration board, however, a single board can work with several different PBXs, with the software configuration selected to reflect the PBX in use. When recording speech, the PBX Integration board digitizes it as Pulse Code Modulation (PCM), Adaptive Differential Pulse Code Modulation (ADPCM), GSM 610, or G.726. The digitizing rate is selected on a channel-by-channel basis and can be changed each time a record or play function is initiated. The processed speech is stored on the host PC's hard disk. When playing back a stored file, the voice information from the host PC is passed to the PBX Integration board where it is converted into analog voice signals for transmission to the PBX.

The on-board control processor controls all operations of the PBX Integration board via a local bus and interprets and executes commands from the host PC. This processor handles real-time events, manages data flow to the host PC to provide faster system response time, reduces PC host processing demands, processes DTMF and PBX signaling before passing them to the application, and frees the DSP to perform signal processing. Communication between this processor and the host PC is via the shared buffer memory that acts as an input/output buffer and thus increases the efficiency of disk file transfers. This shared buffer memory interfaces to the host PC via the PCI bus.

## 3.2. PBX Integration Features Supported

PBX Integration boards incorporate both circuitry and firmware to integrate applications with specific PBXs. The Unified API, used with the PBX Integration board, enables programmers to more easily develop a single application capable of supporting multiple manufacturer's PBXs. The Unified API also enables applications to access the important digital information sent between a PBX and its station sets. This information is useful in a variety of applications including Voice Mail and Call Center.

### 3.2.1. Unified API

The Unified API (Application Programming Interface) allows a single application to function on a variety of manufacturers switches. Functioning as an extension to the Dialogic standard voice API, the Unified API offers a single design model that allows developers to take advantage of advanced PBX features (such as called/calling number ID and ASCII display information).

- **Called/Calling number ID** - usually two sets of digits representing either a trunk line or an extension. This is not to be confused with caller ID received from a CO which provides the telephone number of an outside caller. It is important for an application to know where a call originated and to what extension it is intended. When a call is transferred (or "bounced") through a PBX, this information may be needed by an application at the final destination. If it is not present, the originator (if they are still connected) will have to re-enter the information.
- **Retrieve LCD/LED prompts and indicators** - Different PBXs have different types of prompts and indicators that relay status information of the station set. By capturing and processing this data, an application can "see" what prompts or indicators have been set.
- **Read displays** - There are many types of information displayed on a phone; for instance, hook state, messages, features, and other ASCII text. By capturing and processing this data, an application can "see" what is on the display. This is useful for determining the state of the PBX Integration board. Also, when ANI and DNIS digits are available through the PBX, the CO caller ID can be obtained. Display data is also useful when programming a PBX. Because the PBX Integration boards allow applications to "press" buttons, applications can be written to program the PBX in the same way as using a station set to program the PBX.

- **Accessing PBX features using dial strings** - The PBX Integration board allows applications to access features that are available through a station set. These functions include call transfer, hold, setting the message waiting indicator, and dialing programmable keys.
- **Disconnect supervision** - When a PBX detects a hang-up from one of its extensions, information is passes to the CO, which in turn hangs up. Typically this is accomplished using a loop current drop. However, if the CO hangs up first, a loop current drop is sent to the PBX but is not passed to the station set. Instead, the station set receives a disconnect message. The PBX Integration board interprets this disconnect message as a loop current drop event. *Not all PBXs support disconnect supervision.*

Utility functions included in the Unified API allow programmers to control the PBX Integration board. Your application can retrieve the PBX Integration channel and board type, obtain and set PBX Integration channel and board parameters, start and stop the D/42 driver, retrieve D/42 firmware/driver/library version numbers, and retrieve error information.

By using the Unified API to determine the type of switch that the PBX Integration board is connected to, programmers can create an application that can provide specific control for each PBX. Specific control is accomplished using dial strings. Some examples are call transfer, call forward, message waiting light manipulation, and pressing console buttons. The PBX Integration board is capable of performing most functions that are available to a telephone connected to the PBX.

Developers who wish to continue designing switch-specific applications can continue to do so, as the 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, the Unified API provides for backward compatibility, preserving their development investment.

## 3.3. PBX Integration Board Description

The PBX Integration board is a PCI form factor voice/FAX processing board that can interface directly to several different types of PBXs. The PBX Integration Board emulates telephones that connect to the supported PBXs. Application programs using the PBX Integration board can answer incoming calls, place outbound calls, record and playback voice files, detect and generate tones, access the called/calling number ID for calls forwarded or transferred from within the PBX, access trunk ID for calls originating outside the PBX, send and receive

faxes, and control message notification. The PBX Integration board also provides positive disconnect supervision to immediately detect when a caller has hung up. When used with one of the supported PBXs, the PBX Integration board provides a flexible platform for developing integrated computer telephony applications. Dialogic developers can integrate current D/4x applications on the PBX Integration board with minimal software modifications and create more efficient applications for the PBX by offering value-added features.
A PBX Integration board has either four or eight channels that can be connected directly to a supported PBX.

### 3.3.1. Features

- voice board with four or eight independent four-wire interfaces to a PBX, thereby reducing the cost and complexity of application integration
- interfaces directly to various PBXs
- emulates telephones
- automatically answers calls
- detects Touch Tones
- plays voice messages to a caller
- digitizes, compresses and records voice signals
- places outbound calls and automatically reports the result
- retrieves called/calling number ID to enable calls to be intelligently handled
- activates/deactivates message waiting indicators to provide message notification
- supports two FAX channels at any given time
- allows supervised (recommended) and blind transfers for automated attendant applications
- provides positive disconnect supervision to immediately detect when a caller has hung up
- enables development of applications across a variety of PBX systems using the Unified API.

### 3.3.2. Functional Description

The PBX Integration board connects to several different PBXs, each of which has one or more compatible telephones with which it communicates. The PBX Integration board emulates these telephones, which have Feature Keys and LCD displays for accessing and employing advanced features of the compatible PBXs. Each of the four or eight line interfaces on PBX Integration boards receive voice and control data from the connected PBX. The voice data is compressed by a DSP using an one of the available encoding methods and then sent to the host PC to be stored.

Control data from the PBX switch passes through the digital duplexer on the PBX Integration board to a command processor where it is converted from its native format to D/41D format. The resulting serial bit stream is then converted into a parallel bit stream that is sent via the local bus to the on-board control processor which either acts on the information or passes the event to the application (see Figure 1).

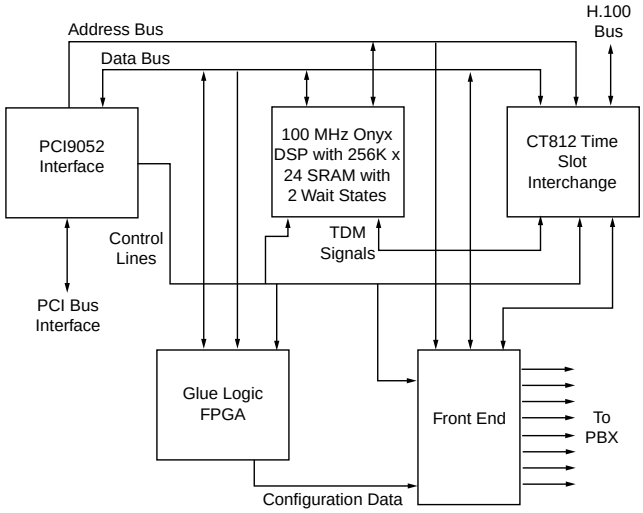



**Figure 1. PBX Integration board Functional Block Diagram**

Voice files stored on the host PC are read by the host driver and transferred to the PBX Integration board via the PCI Bus. These voice signals are buffered by the control processor and decoded into 64 kbps PCM signals by the DSP. These PCM voice signals are then sent to the PBX interface link for transport to the caller. A system-wide, TDM signal sharing bus, called CT Bus, is also provided for the exchange of signal streams with other resource boards, signal transport boards, or other interfaces.

In addition to having all the standard features of a Dialogic D/41D board, the PBX Integration board can access enhanced PBX features, when available, such as:

- call transfer/conference
- turn phone message waiting indicators on or off
- callback request
- calling number identification (Calling Number ID).

The PBX Integration board has an on-board microprocessor and a high-speed Digital Signal Processor (DSP) to provide voice and call processing. SpringWare™ voice processing firmware is downloaded from the host computer to SRAM and DSP memory when the PBX Integration board is started. SpringWare offers several features, including PerfectPitch™ Speed Control, PerfectLevel™ Volume Control, Global Tone Detection™, and Positive Voice Detection™. Global Tone Detection allows applications to detect special intercept tones, FAX tones, modem tones, and non-standard PBX or user-defined tones, such as those used in international networks.
Other DSP-based SpringWare features include G.711 A-law and µ-law PCM, ADPCM, GSM 610, and G.726 voice encoding. An application may dynamically switch between sampling rates and coding methods to meet specific requirements for voice quality and data storage. Enhanced algorithms provide reliable DTMF detection, DTMF cut-through, and talk off/play off suppression.

### 3.3.3. Configurations

The PBX Integration board connects to a line circuit board in a supported PBX to build sophisticated, computer telephony systems. The PBX Integration board installs in a minimum 90 MHz Pentium™5- or the equivalent Celeron®-based platform with an available PCI bus slot for an 8-port system. The host system must provide a CPU of Pentium or Celeron class at 266 MHz speed or higher for a 64-port system, including eight available PCI slots. The PBX Integration board occupies a single expansion slot, and up to eight boards can be configured in a system, with each board sharing the same interrupt level. The maximum number of ports supported is 64, dependent on the application, the amount of disk I/O required, and the host computer's CPU.
The PBX Integration board shares a large degree of common hardware and firmware architecture with other Dialogic products for maximum flexibility and scalability. Features can be added or systems can grow while protecting investment in hardware and application code. With only minimum modifications, applications can be easily ported to lower or higher line-density platforms.

### 3.3.4. Software Support

The development package includes all required libraries, drivers, and headers for simplified and seamless PBX integration. Diagnostics and demo programs provide additional tools and examples that allow developers to create complex multi-channel voice applications.

# 4. PBX Systems

## 4.1. Lucent Definity PBXs

The Lucent Definity product family includes the Definity 75/85 (4-wire) and the
Definity G3 (2-wire) PBXs. The PBX Integration board can be used with either of these switches. The PBXs use digital signaling to control their station sets and digitized voice.
A PBX Integration board has either four or eight channels that are connected directly to a station module in a Lucent PBX. The PBX switch has many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)
- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable Feature Keys
- called/calling number identification
- call forwarding.

### 4.1.1. Lucent Switch Programming Requirements

There are specific switch programming requirements for using a PBX Integration board with a Lucent Definity PBX. You must ensure that the PBX is configured properly so that the PBX Integration board functions correctly.

#### Port Number Settings

Each board in a Lucent PBX is assigned a port number. The number of ports vary according to the board type (2-wire or 4-wire). A 2-wire board has 16 ports, while the 4-wire boards has eight.
*Table 1* lists the structure used when configuring Lucent Definity PBX. For details about programming a Lucent PBX, refer to the appropriate Lucent manual.
The following are examples of the switch settings:

**Table 1. Lucent Definity Configuration Example**

| Slot # | Board Type | Telephone Type | Extension Numbers | Port Settings |
|---|---|---|---|---|
| 3 | TN2181 2-wire | 8434D | 1000-1015 | 01A0301-01A0316 |
| 4 | TN2181 2-wire | 8434D | 1016-1031 | 01A0401-01A0416 |
| 5 | TN754B 4-wire | 7434D | 1032-1039 | 01A0501-01A0508 |
| 6 | TN754B 4-wire | 7434D | 1040-1047 | 01A0601- 01A0608 |
| 7 | TN754B 4-wire | 7434D | 1048-1055 | 01A0701-01A0708 |
| 8 | TN754B 4-wire | 7434D | 1056-1063 | 01A0801-01A0808 |

The settings above should be tailored according to the your specific needs.

### Message Waiting Light Settings

You must make certain settings from a Lucent management terminal to ensure that Message Waiting Indicator (MWI) features work correctly.

1. Login to switch from a management terminal.
2. Type command '**CH STAT** <ext>' where ext is the extension of a PBX Integration board port.

   On the Lucent phone sets, go to the Button Assignments page and set button 33 to '**lwc-store'** and button 34 to '**lwc-cancel**'.

   **NOTE:** If these features are programmed into any other button, they must be removed, as there may be only one occurrence of these features per extension.

4. Repeat as necessary for other extensions.

### 4.1.2. Using the PBX Integration Board

The PBX Integration board performs functions available to Lucent 7434 (4-wire) and 8434 (2-wire) telephone sets (see *Figure 2* and *Figure 3*). These telephone sets use two LED displays per Feature Button to show status (next to the Feature Buttons) and an LCD display to show user prompts and messages (above the display buttons). The PBX Integration board can:

- transfer calls
- set the message waiting indicator
- read the LED display
- read LED indicators
- read the called/calling number ID
- press keys.

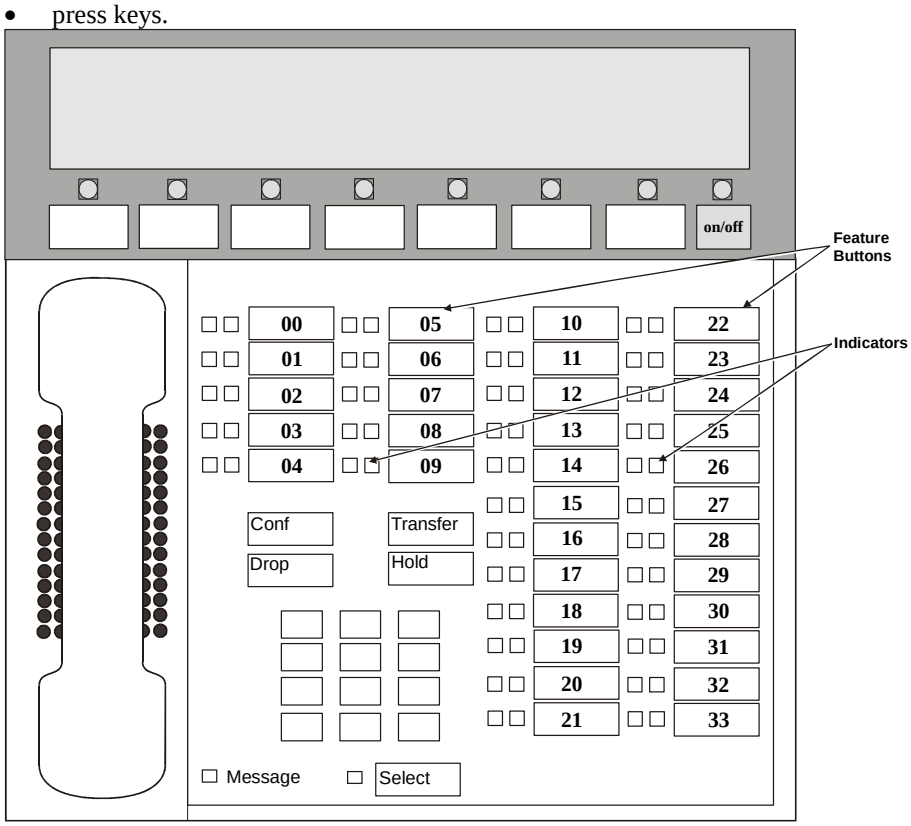



**Figure 2. Lucent 7434 Telephone**

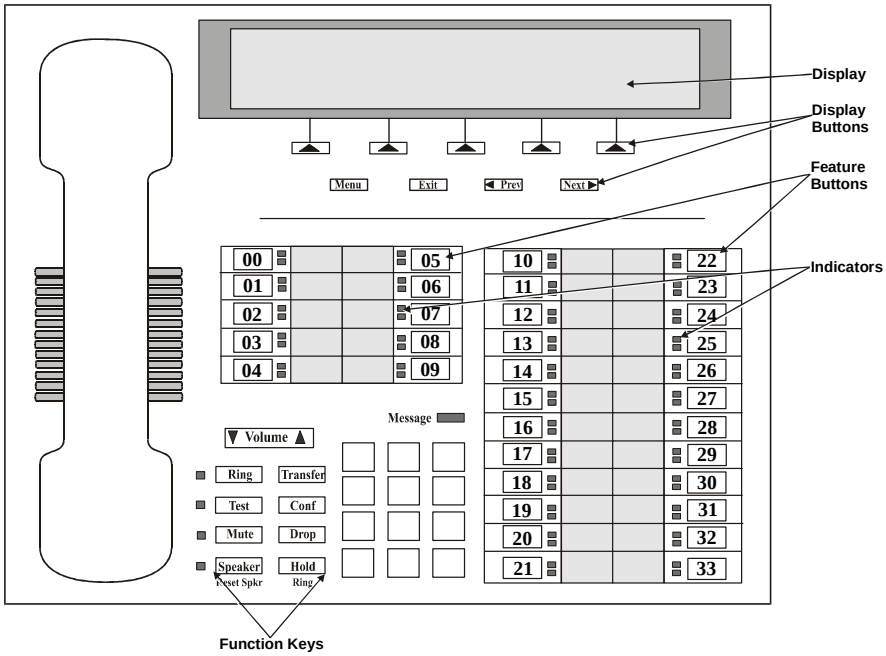



**Figure 3. Lucent 8434 Telephone**

### 4.1.3. Programmable Feature Keys

As illustrated in *Figure 2* and *Figure 3*, there are 34 Programmable Feature Keys found on the Lucent 7434 and 8434 telephones. These keys are configured either during installation or by the user (using the telephone set or the PBX Integration board). There are two LED Indicators associated with each Feature Button. The PBX Integration board can also emulate four Lucent Functions Keys: Transfer Conference, Drop, and Hold.

As mentioned above, each line or Feature Key actually has two indicator lights. The red indicator tells the user that the line is being used or that the line will be the one used when the handset is lifted. The green indicator (bottom on the 8434 and right on the 7434) tells the user that the line or feature is in use. In other words, when you pick up the handset or press a Feature Key, the green indicator goes on. When a call is on hold, the green indicator for that line flashes and the red indicator goes off. The red light is either off or on (a value of eight [0x08] indicates ON), while the green light has six possible values. The status of the indicators is obtained by bitwise-ANDing the returned value

from the green light with the value from the red light (green light value + red light value). In other words, the value for a line indicator in use with a call would be nine--0x08 (for red light on) + 0x01 (for green light on). The status conditions for each byte of the green light are defined as follows:

**Table 2. Lucent 7434 and 8434 LED Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |
| unknown | 0x05 |

### Reading LED Indicators

The PBX Integration board can determine the state of its LED Indicators by using the **d42_indicators( )** function to retrieve the LED Indicators data. This function places the Line Indicator data (34 bytes) in an application buffer. Bytes 1-34 contain the indicator status for Memory Keys 00-33, respectively (see *Table 3*).

**Table 3. Lucent 7434 and 8434**
**Direct Key Dialing Strings for Feature Keys**

| Byte | Key Description | Dial String |
|---|---|---|
| 1 | Feature Button 00 | <ESC>KA |
| 2 | Feature Button 01 | <ESC>KB |
| 3 | Feature Button 02 | <ESC>KC |
| 4 | Feature Button 03 | <ESC>KD |
| 5 | Feature Button 04 | <ESC>KE |
| 6 | Feature Button 05 | <ESC>KF |
| 7 | Feature Button 06 | <ESC>KG |
| 8 | Feature Button 07 | <ESC>KH |
| 9 | Feature Button 08 | <ESC>KI |

| Byte | Key Description | Dial String |
|---|---|---|
| 10 | Feature Button 09 | <ESC>KJ |
| 11 | Feature Button 10 | <ESC>KK |
| 12 | Feature Button 11 | <ESC>KL |
| 13 | Feature Button 12 | <ESC>KM |
| 14 | Feature Button 13 | <ESC>KN |
| 15 | Feature Button 14 | <ESC>KO |
| 16 | Feature Button 15 | <ESC>KP |
| 17 | Feature Button 16 | <ESC>KQ |
| 18 | Feature Button 17 | <ESC>KR |
| 19 | Feature Button 18 | <ESC>KS |
| 20 | Feature Button 19 | <ESC>KT |
| 21 | Feature Button 20 | <ESC>KU |
| 22 | Feature Button 21 | <ESC>KV |
| 23 | Feature Button 22 | <ESC>KW |
| 24 | Feature Button 23 | <ESC>KX |
| 25 | Feature Button 24 | <ESC>KY |
| 26 | Feature Button 25 | <ESC>KZ |
| 27 | Feature Button 26 | <ESC>Ka |
| 28 | Feature Button 27 | <ESC>Kb |
| 29 | Feature Button 28 | <ESC>Kc |
| 30 | Feature Button 29 | <ESC>Kd |
| 31 | Feature Button 30 | <ESC>Ke |
| 32 | Feature Button 31 | <ESC>Kf |
| 33 | Feature Button 32 | <ESC>Kg |
| 34 | Feature Button 33 | <ESC>Kh |

### ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LED Indicators on a given channel on a PBX Integration board. The data placed in the application buffer is shown below. If the data for byte 19 is 0x09 and byte 28 is 0x03, the red and green indicators are on for Feature Button 19 indicating that the line is in use for a call, and the green indicator for Memory Button 28 is flashing, indicating that the call is on hold.
Refer to the *PBX Integration Software Reference* for more information about using the **d42_indicators( )** function.

| | Feature Button 00 | Feature Button 01 | Feature Button 02 | Feature Button 03 | Feature Button 04 | Feature Button 05 | Feature Button 06 | Feature Button 07 | Feature Button 08 | Feature Button 09 | Feature Button 10 | Feature Button 11 | Feature Button 12 | Feature Button 13 | Feature Button 14 | Feature Button 15 | Feature Button 16 | Feature Button 17 | Feature Button 18 | Feature Button 19 | Feature Button 20 | Feature Button 21 | Feature Button 22 | Feature Button 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 09 | 00 | 00 | 00 | 00 |
| **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 |

| | Feature Button 24 | Feature Button 25 | Feature Button 26 | Feature Button 27 | Feature Button 28 | Feature Button 29 | Feature Button 30 | Feature Button 31 | Feature Button 32 | Feature Button 33 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### Pressing Feature Keys

The PBX Integration board can “press” any of the Lucent 7434 or 8434’s Feature Keys using the **dx_dial( )** function. Refer to the *PBX Integration Software Reference* for more information about dialing programmable keys. Each Feature Button on the 7434 and 8434 telephones is assigned a dial string sequence (refer to *Table 3*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Feature Button.

## 4.1.4. Lucent Function Keys

Lucent telephones also include Function Keys that the PBX Integration board can emulate to perform various functions. PBX Integration board can emulate four Lucent Functions Keys: Transfer, Conference, Drop, and Hold.

### Pressing Function Keys

The PBX Integration board can "press" Lucent telephone Function Keys using the **dx_dial( )** function. The Function Keys on the Lucent 7434 and 8434 telephones assigned a dial string sequence are listed in *Table 4*. By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can dial these four Lucent Function Keys. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**Table 4. Lucent 7434 and 8434 Direct Key Dialing Strings for Function Keys**

| Dial String | Key Description |
|---|---|
| <ESC>Ki | Hold |
| <ESC>Kj | Drop |
| <ESC>Kk | Transfer |
| <ESC>Kl | Conference |

### 4.1.5. Display Keys

As shown in *Figure 3*, there are five Display Keys located below the LCD display. These keys are associated with specific prompts shown on the LCD display depending on the current state of the phone (shown on the bottom row of the LCD display). The PBX Integration board cannot use the two bottom, right-most Keys, **Prev** and **Next**.

### Pressing Display Keys

The PBX Integration board can respond to a prompt and "press" the appropriate Display Key using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Display Key on the Lucent 8434 telephone is assigned a dial string sequence (refer to *Table 5*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any of its first seven Display Keys.

**Table 5. 8434 Direct Key Dialing Strings for Display Keys**

| Dial String | Key Description |
|---|---|
| <ESC>Km | Display Key 00 |

| | |
|---|---|
| <ESC>Kn | Display Key 01 |
| <ESC>Ko | Display Key 02 |
| <ESC>Kp | Display Key 03 |
| <ESC>Kq | Display Key 04 |
| <ESC>Kr | Display Key 05 |
| <ESC>Ks | Display Key 06 |

### 4.1.6. Alphanumeric Display

The alphanumeric display is a two row, 50-digit LED that is used to show the activity of the phone. Some examples are:

- date and time
- feature names
- error messages
- called/calling identification
- phone status
- line selection
- Display Key prompts

The data used to display information in the LED alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature). The PBX Integration board can retrieve the information on its alphanumeric display using the **d42_displayex( )** function. The function places the display data (50 bytes) in an application buffer. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_displayex( )** function.

#### ■ Example

An application uses the **dx_dial( )** function and the appropriate dial string to press keys to dial extension number 1045. The **d42_display( )** function is used to retrieve the display data and place it in an application buffer (shown below). The information for the top row (last 25 characters) of the display is checked. Data in bytes 00 through 05 indicate that extension 1045 is being dialed.

```
        a = 1 0 4 5
data  61 3D 01 00 04 05 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

| 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 24 |
|---|---|
| data byte | 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20<br>25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 |

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set *processes* the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.

Both the calling and called number IDs can be retrieved using the **d42_gtcallid( )** function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference* for more information about using **d42_gtcallid( )** function.

The contents of the called/calling number ID are shown in *Table 6* as seen by the receiver of the call).

**Table 6. Called/Calling Number ID Data for the Lucent Definity**

| Call Route | Called/Calling Number ID Data |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_displayex( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

### ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 4 | _ | 2 | 2 | 1 | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 34 | 5F | 32 | 32 | 31 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.1.7. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting Indicator (on or off) on another extension using the **dx_dial( )** function and the appropriate dial string. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** Message Waiting can also be set using the dx_dial() function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the dx_dial() function as described so that your application will maintain functionality across different manufacturers' switches.

#### MWI On

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCO><extention><ESCO> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCO> means Escape character followed by O.

#### MWI Off

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCF><extention><ESCF> (Optional pause character may be used)

3) Go On hook using the **dx_sethook()** again

<ESCF> means Escape character followed by F.

The PBX Integration board can determine the state of its Message Waiting Indicator using the **d42_indicators( )** function to retrieve the LED Indicators data. Byte 34 contains the Message Waiting indicator status (0x00 is off; 0x01 is on). Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

## ■ Example

An application uses the **d42_indicators( )** function to retrieve the LED Indicators data for a specified channel on the PBX Integration board to determine if a message is waiting. The LED indicators data is shown below. The data 0x00 shows that the MWI indicator is off (there are no messages waiting).

| | Feature Button 00 | Feature Button 01 | Feature Button 02 | Feature Button 03 | Feature Button 04 | Feature Button 05 | Feature Button 06 | Feature Button 07 | Feature Button 08 | Feature Button 09 | Feature Button 10 | Feature Button 11 | Feature Button 12 | Feature Button 13 | Feature Button 14 | Feature Button 15 | Feature Button 16 | Feature Button 17 | Feature Button 18 | Feature Button 19 | Feature Button 20 | Feature Button 21 | Feature Button 22 | Feature Button 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| **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 |

| | Feature Button 24 | Feature Button 25 | Feature Button 26 | Feature Button 27 | Feature Button 28 | Feature Button 29 | Feature Button 30 | Feature Button 31 | Feature Button 32 | Feature Button 33 | MWI | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### 4.1.8. Transferring a Call

The PBX Integration board can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string; however, you should use the

&,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (Refer to the *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer).* When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

### ■ Example

An application answers a call and plays a greeting message prompting the caller to enter the extension they wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If the extension is busy or not answered, the application reconnects to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

## 4.2. Siemens ROLM PBX

The ROLM product family actually includes three generations of ROLM and related PBXs:

1. The original ROLM
2. IBM ROLM 9751 series
3. Siemens Hicom 300 with the appropriate interface cards

The PBX Integration board emulating the ROLM 400 telephone can be used with any of these switches. The ROLM PBXs use digital signaling to control their station sets and digitized voice.

The PBX Integration board has either four or eight channels that are connected directly to a station module in a Siemens ROLM PBX. The PBX switch has many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)

- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable feature keys
- called/calling number identification
- call forwarding.

### 4.2.1. Siemens ROLM Programming Requirements

There are specific switch programming requirements for using a PBX Integration Board with a Siemens ROLM PBX. You must ensure that these features are set exactly (and assigned to the right keys) so that the PBX Integration Board and the Unified API function correctly.

- All PBX Integration Board ports on a ROLM system must be programmed as ROLMphone 400 telephones.
- LINE must be programmed on Feature Key 09, and the ROLMphone must be programmed to select this line when going offhook.
- XFER (transfer) must be programmed on Feature Key 38.
- MWI (Message Waiting Indication) mechanisms are different with ROLM 9006 (or ROLM integration on the Hicom 300E) and ROLM 9005 PBX.

#### For ROLM 9006 PBX or ROLM integration on the Hicom 300E

- DDS (speed dial) must be programmed on Feature Key 03 for the correct message waiting **"O**N" feature access code, which is *59 (*default, but is dependent on the PBX setup. Consult the PBX Administrator for the correct feature access code*) when you are using the ROLM integration on the Hicom 300E or when you are using the ROLM 9006 PBX. To program this key on the ROLMphone:
  1. Press PROG (Feature Key 20).
  2. Then press Feature Key 03.
  3. Dial *59 (or the correct PBX dependent Feature Access Code), and press PROG again.
  4. The phone display indicates "STORED" and message-waiting light (or Mailbox indicator light) ON is now set for Feature Key 03.
- DDS (speed dial) must be programmed on Feature Key 04 for the correct message-waiting OFF feature access code, which is #60 (*default, but is dependent on the PBX setup. Consult the PBX Administrator for the correct feature access code*) when using the ROLM integration on the

Hicom 300E or when you are using the ROLM 9006 PBX. To program this key on the ROLMphone:
1. Press PROG (Feature Key 20).
2. Then press Feature Key 04.
3. Dial #60 (or the correct PBX dependent Feature Access Code), and press PROG again.
4. The phone display indicates STORED and message-waiting light (or Mailbox indicator light) OFF is now set for Feature Key 04.

**For ROLM 9005 PBX**

- In this case the MWI ON/OFF key is a toggle key and it must be programmed to be the feature key 37.

**NOTE:** For transferred calls, the called-party ID appears as a direct call because the PBX does not write the called-party ID to the display.

**NOTE:** For message waiting, only the port that sets a message-waiting indicator can clear it.

### 4.2.2. Using the PBX Integration Board

The PBX Integration board performs functions available to a ROLMphone 400 telephone set (see *Figure 5*). An ROLMphone 400 telephone set uses an LED displays to show key status (next to the keys) and user prompts and messages on the display to provide various options. The PBX Integration board can:

- transfer calls
- set the message waiting indicator
- read the LCD display
- read LED indicators
- read the called/calling number ID
- press keys.

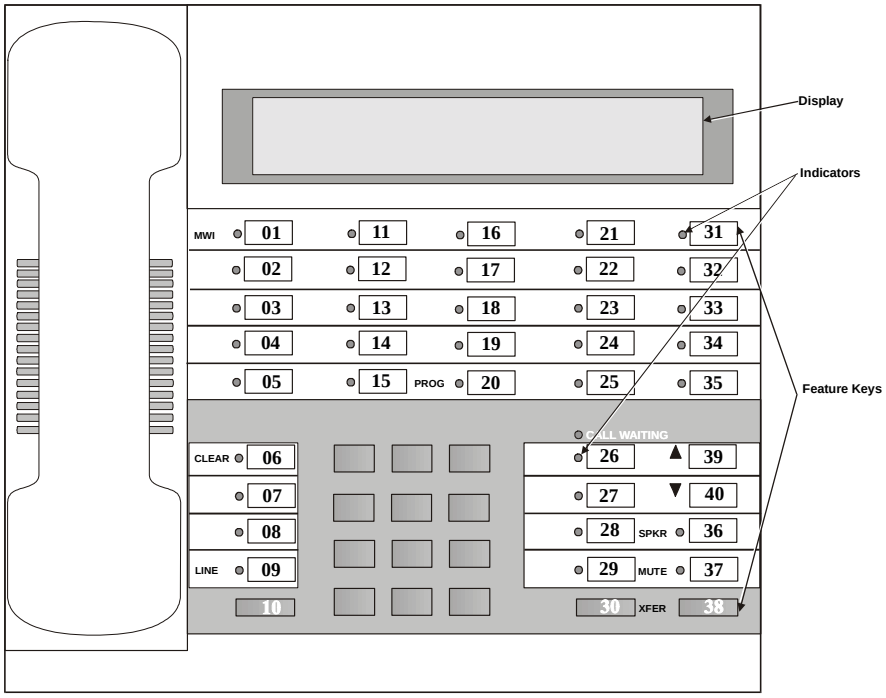



**Figure 4. Siemens ROLMphone 400**

### 4.2.3. Programmable Feature Keys

As illustrated in *Figure 4*, there are 40 Feature Keys located below the display on the ROLMphone 400 telephone. These keys are configured either during PBX installation or by the user (using the telephone set or the PBX Integration board). The CLEAR, SPEAKER, MUTE, XFR, and LINE keys are assigned during PBX configuration and cannot be user programmed. The MAILBOX indicator programmed on each phone (see *4.2.1. Siemens ROLM Programming Requirements* above) for Feature Key 01. Feature Keys 39 and 40 are used for volume control and cannot be programmed either. There is an LED Indicator associated with each key, except those discussed in the following paragraph. The LED Indicators are circular and can take on one of the six states listed in *Table 7*.

**Table 7. ROLMphone 400 LED Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |
| unknown | 0x05 |

### Reading LED Indicators

The PBX Integration board can determine the state of its LED Indicators by using the **d42_indicators( )** function to retrieve the LED Indicators data. This function places the LED Indicator data (37 bytes) in an application buffer. Bytes 00-36 contain the indicator status for Feature Keys 01-37, respectively (see *Table 8*). As indicated in the example below, Feature Keys 10, 30, and 38-40 do not have LED indicators.

**Table 8. ROLMphone 400 Direct Key Dialing Strings for Feature Keys**

| Byte | Key Description | Dial String |
|---|---|---|
| 00 | Feature Key 01 - MAILBOX | <ESC>KI |
| 01 | Feature Key 02 | <ESC>KH |
| 02 | Feature Key 03 | <ESC>KG |
| 03 | Feature Key 04 | <ESC>KF |
| 04 | Feature Key 05 | <ESC>KE |
| 05 | Feature Key 06 - CLEAR (flash) | <ESC>KD |
| 06 | Feature Key 07 | <ESC>KC |
| 07 | Feature Key 08 | <ESC>KB |
| 08 | Feature Key 09 - LINE | <ESC>KA |
| 09 | Feature Key 10 | <ESC>Kl |
| 10 | Feature Key 11 | <ESC>KN |
| 11 | Feature Key 12 | <ESC>KM |
| 12 | Feature Key 13 | <ESC>KL |
| 13 | Feature Key 14 | <ESC>KK |
| 14 | Feature Key 15 | <ESC>KJ |
| 15 | Feature Key 16 | <ESC>KS |
| 16 | Feature Key 17 | <ESC>KR |
| 17 | Feature Key 18 | <ESC>KQ |
| 18 | Feature Key 19 | <ESC>KP |
| 19 | Feature Key 20 - PROG (program) | <ESC>KO |
| 20 | Feature Key 21 | <ESC>KX |
| 21 | Feature Key 22 | <ESC>KW |
| 22 | Feature Key 23 | <ESC>KV |
| 23 | Feature Key 24 | <ESC>KU |
| 24 | Feature Key 25 | <ESC>KT |

| Byte | Key Description | Dial String |
|---|---|---|
| 25 | Feature Key 26 | <ESC>Kg |
| 26 | Feature Key 27 | <ESC>Kf |
| 27 | Feature Key 28 | <ESC>Ke |
| 28 | Feature Key 29 | <ESC>Kd |
| 29 | Feature Key 30 | <ESC>Km |
| 30 | Feature Key 31 | <ESC>Kc |
| 31 | Feature Key 32 | <ESC>Kb |
| 32 | Feature Key 33 | <ESC>Ka |
| 33 | Feature Key 34 | <ESC>KZ |
| 34 | Feature Key 35 | <ESC>KY |
| 35 | Feature Key 36 - SPEAKER | <ESC>Ki |
| 36 | Feature Key 37 - MWCTR* | <ESC>Kh |
| 37 | Feature Key 38 - XFER | <ESC>Kn |
| 38 | Feature Key 39 - Volume Up | <ESC>Kk |
| 39 | Feature Key 40 - Volume Down | <ESC>Kj |

*MWCTR = Message Waiting Control

## ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LED Indicators on a given channel on a PBX Integration board. The data placed in the application buffer is shown below. If the data for byte 08is 0x01, the circular indicator for Feature Key 09 is on. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

| | Feautre Key 01 | Feautre Key 02 | Feautre Key 03 | Feautre Key 04 | Feautre Key 05 | Feautre Key 06 | Feautre Key 07 | Feautre Key 08 | Feautre Key 09 | No LED Inidcator | Feautre Key 11 | Feautre Key 12 | Feautre Key 13 | Feautre Key 14 | Feautre Key 15 | Feautre Key 16 | Feautre Key 17 | Feautre Key 18 | Feautre Key 19 | Feautre Key 20 | Feautre Key 21 | Feautre Key 22 | Feautre Key 23 | Feature Key 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | *xx* | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | *xx* | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

| | Feature Key25 | Feature Key26 | Feature Key27 | Feature Key28 | Feature Key29 | NO LED Indicator | Feature Key31 | Feature Key32 | Feature Key33 | Feature Key34 | Feature Key35 | Feature Key36 | Feature Key37 | NO LED Indicator | NO LED Indicator | NO LED Indicator | MWI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

### Pressing Feature Keys

The PBX Integration board can "press" any of the ROLMphone 400 Feature Keys using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Feature Key on the ROLMphone 400 telephone is assigned a dial string sequence (refer to *Table 8*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Feature Key.

### 4.2.4. Alphanumeric Display

The alphanumeric display is a two row, 60-character LCD that is used to show the activity of the phone. Some examples are:

- date and time
- feature names
- text messages
- error messages
- called/calling identification
- phone status
- line selection

The data used to display information in the LCD alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature).

The PBX Integration board can retrieve the information on its alphanumeric display using the **d42_display( )** function. The function places the display data (48 bytes) in an application buffer. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

## ■ Example

An application uses the **dx_dial( )** function and the appropriate dial string to press keys dial extension number 1045. The **d42_display( )** function is used to retrieve the display data and place it in an application buffer (shown below). The information for the top row (first 30 characters) of the display is checked. Data in bytes 00 through 03 indicate that extension 1045 is being dialed.

| | 1 0 4 5 | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 01 | 00 | 04 | 05 | 4C | 4C | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set processes the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.

Both the calling and called number IDs can be retrieved using the **d42_gtcallid( )** function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference for* more information about using **d42_gtcallid( )** function.

The contents of the called/calling number ID are shown in *Table 13* (as seen by the receiver of the call).

**Table 9. Called/Calling Number ID Data for the ROLM**

| Call Route | Called/Calling Number ID Data |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_display( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

### ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 1 | _ | 2 | 2 | 4 | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 31 | 5F | 32 | 32 | 34 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.2.5. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting Indicator (on or off) on another extension using the **dx_dial( )** function and the appropriate dial string, as described in the *PBX Integration board Software Reference* for your particular operating system.

**NOTE:** Message Waiting can also be set using the dx_dial() function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the dx_dial() function as described so that your application will maintain functionality across different manufacturers' switches.

#### MWI On

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCO><extention><ESCO> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCO> means Escape character followed by O.

#### MWI Off

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**

2) Call the **dx_dial()** function. The dial string is <ESCF><extention><ESCF> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

**NOTE:** <ESCF> means Escape character followed by F.
The PBX Integration board can determine the state of its Message Waiting Indicator using the **d42_indicators( )** function to retrieve the LED Indicators data. Byte 40 contains the Message Waiting indicator status (0x00 is off; 0x01 is on). Refer to the *PBX Integration board Software Reference for* more information about using the **d42_indicators( )** function.

### ■ Example

An application uses the **d42_indicators( )** function to retrieve the LED Indicators data for a specified channel on the PBX Integration board to determine if a message is waiting. The LED indicators data is shown below. The data 0x01 shows that the MWI indicator is on (there are messages waiting).

| | MWI | Feautre Key 02 | Feautre Key 03 | Feautre Key 04 | Feautre Key 05 | Feautre Key 06 | Feautre Key 07 | Feautre Key 08 | Feautre Key 09 | No LED Inidcator | Feautre Key 11 | Feautre Key 12 | Feautre Key 13 | Feautre Key 14 | Feautre Key 15 | Feautre Key 16 | Feautre Key 17 | Feautre Key 18 | Feautre Key 19 | Feautre Key 20 | Feautre Key 21 | Feautre Key 22 | Feautre Key 23 | Feature Key 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| **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 |

| | Feature Key 25 | Feature Key 26 | Feature Key 27 | Feature Key 28 | Feature Key 29 | NO LED Indicator | Feature Key 31 | Feature Key 32 | Feature Key 33 | Feature Key 34 | Feature Key 35 | Feature Key 36 | Feature Key 37 | NO LED Indicator | NO LED Indicator | NO LED Indicator | Call Waiting | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | *xx* | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### 4.2.6. Transferring a Call

The PBX Integration board can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference for* more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string; however, you should use the &,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (Refer to the *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer*). When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed

and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

### ■ Example

An application answers a call and plays a greeting message prompting the caller to enter the extension she wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If the extension is busy or not answered, the application reconnects to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

## 4.3. Siemens Hicom PBX

The Siemens Hicom are a full-featured PBXs that can provide thousands of ports and many PBX voice and data features. The Hicom uses digital signaling to control its station sets and digitized voice.
The PBX Integration board has either four or eight channels that are connected directly to a station module in a Siemens Hicom. The PBX has many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)
- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable Feature Keys
- called/calling number identification
- call forwarding.

### 4.3.1. Siemens Hicom Programming Requirements

There are specific switch programming requirements for using a PBX Integration board with a either a Siemens Hicom 150E or a Hicom 300E PBX. This allows the D./82JCT-U to correctly emulate a Optiset E telephone. Note that the programming is quite different for the two Hicom PBXs supported, so you must ensure that these features are set exactly (and assigned to the right

keys) so that the PBX Integration board and the Unified API function correctly.

### Siemens Hicom 150E

When the Hicom 150E is used with Optiset E phones (see *Figure 5*), the top three programmable keys on the right must be programmed as General Call, Send Message, and Consultation keys, respectively. If these keys are not programmed in this manner, loop current detection, CPID, & (transfer) key, and message waiting will not work. This programming can be accomplished using the following instructions:

1. On the Siemens Optiset E phone, press the Service Menu key, which is the top left programmable key.
2. Press the Scroll Forward key (>) repeatedly to scroll through the choices until the display screen shows "*91 prog feature key," and then press the Select OptiGuide key (the key with the check mark).
3. Press the top right programmable key.
4. After the screen displays the current assignment for the key, press the **Select OptiGuide** key to change the assignment.
5. Press the Scroll Forward key repeatedly to scroll through the choices until the display screen shows "General call key," and then press the **Select OptiGuide** key.
6. Press the Select OptiGuide key again to save and exit.
7. Repeat the above procedure for the second and third programmable key on the top right, programming the second key as the “Send message” key and the third key as the “Consultation” key.

### Siemens Hicom 300E

When the Hicom 300E is used with Optiset E phones (see *Figure 6*), the top two programmable keys (Key 00 and 01) on the left must be programmed as Mailbox and Callback, respectively. Key 02 must be configured to dial the **message waiting lamp on** (MWL_ON) string. Key 03 must be configured to dial the **message waiting lamp off** (MWL_OFF) string. This programming allows an application to use the specified dial string to turn the MWL on and off. In addition, Key 07 must be programmed as the Consultation (transfer) key. Keys 08-12 must be programmed as Line keys, with Key 12 programmed as the General Call Key, which provides the off-hook indicator. Refer to *Figure 5* and *Table 11* for specific Key locations and set-up requirements.

If these keys are not programmed in this manner, loop current detection, CPID, & (transfer) key, and message waiting will not work.

To configure Keys 02 and 03 for the MWL functionality, use the following instructions:

1. Need a button programmed as PROG in the PBX
2. Program DDS keys on button 02 and 03 in the button table of the PBX
3. Press the Scroll Forward key (>) repeatedly to scroll through the choices Siemens Optiset E phone to reach the **Program/Service** option on the display.
4. Press the Select **OptiGuide key** (the key with the check mark) to select.
5. Press the Select OptiGuide key again when **1-Change destinations** appears on the display.
6. Press the Scroll Forward key once to scroll to the **2-Redial** option and then press the **Select OptiGuide** key to select.
7. Press Key 02 (third from the top left, see *Figure 6* below) to set the dial string for MWL_ON.
8. Enter the dial string you wish to use with your Optiset 300E for MWL_ON (for example, #*8)
9. Press the **Select OptiGuide** key again to save.
10. Press the **Select OptiGuide** key again to exit.

11. Repeat the above procedures for the Key 03 to set the MWL_OFF functionality, using a different dial string.

### 4.3.2. Using the PBX Integration Board

The PBX Integration board performs functions available to a Optiset E telephone set (see *Figure 5* and *Figure 6*). An Optiset E telephone set uses an LED displays to show key status (next to the keys) and user prompts and messages on the display to provide various options. The PBX Integration board can:

- transfer calls
- set the message waiting indicator
- read the LCD display
- read LED indicators
- read the called/calling number ID
- press keys.

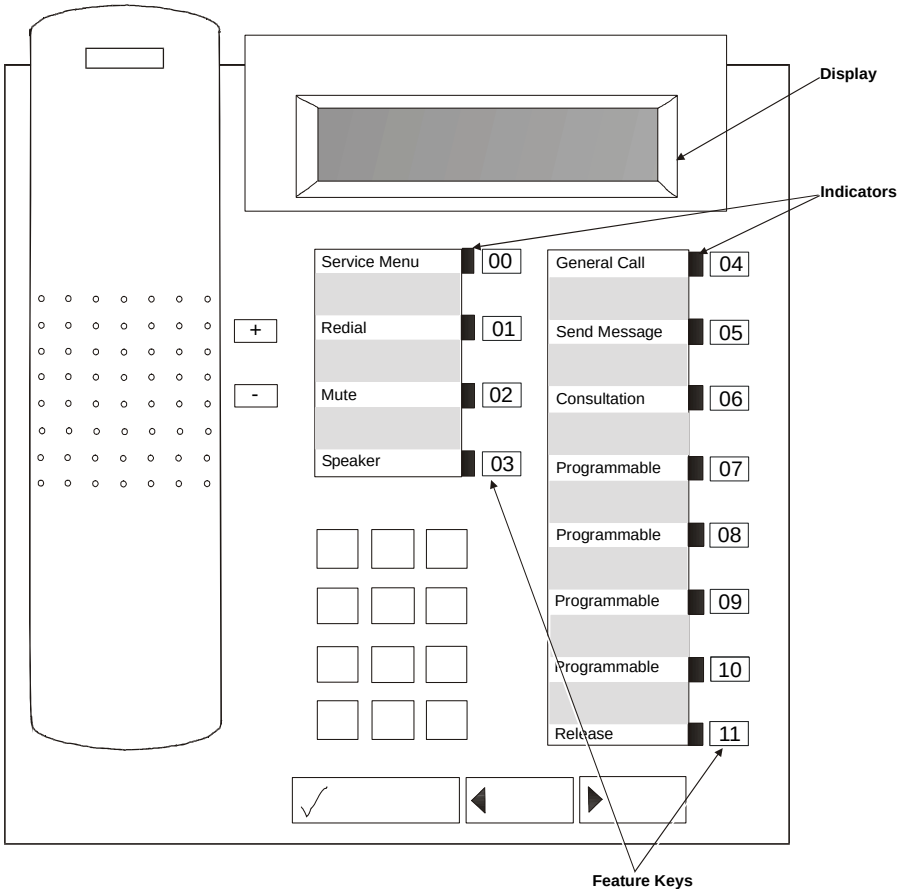



**Figure 5. Siemens Optiset E Telephone with the Hicom 150E**

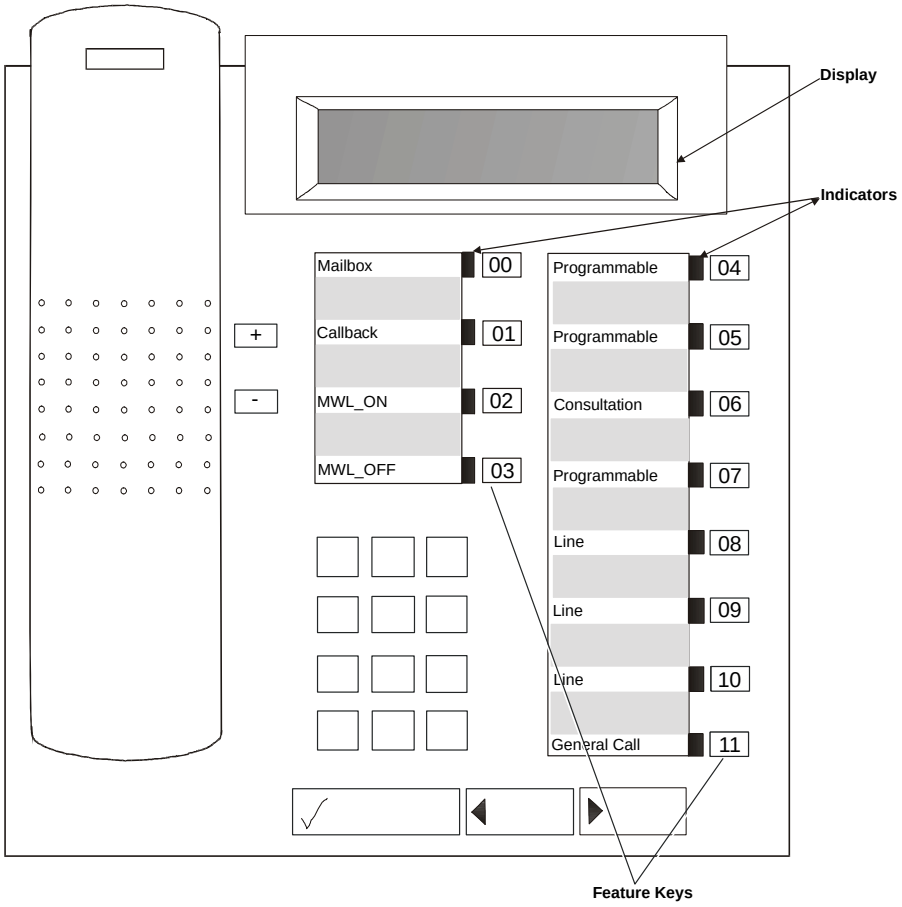



**Figure 6. Siemens Optiset E Telephone with the Hicom 300E**

### 4.3.3. Programmable Feature Keys

As illustrated in *Figure 5* and *Figure 6* , there are 12 Programmable Feature Keys located below the display on the Optiset E telephone. These keys are configured either during PBX installation or by the user (using the telephone set or the PBX Integration board). When using the Optiset E with the Hicom 150E, for example, the Service Menu, Redial, Mute, Speaker, General Call, Send Message, Consultation, and Release keys are assigned during PBX configuration and cannot be user programmed. There is an LED Indicator

associated with each key. The LED Indicators are rectangular and can take on one of the six states listed in *Table 10*.

**Table 10. Optiset E LED Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |
| unknown | 0x05 |

### Reading LED Indicators

The PBX Integration board can determine the state of its LED Indicators by using the **d42_indicators( )** function to retrieve the LED Indicators data. This function places the Line Indicator data (12 bytes) in an application buffer. Bytes 0-11 contain the indicator status for Feature Keys 00-11, respectively (see *Table 11* and *Table 12*).

**Table 11. Optiset E Direct Key Dialing Strings for Feature Keys with Hicom 150E**

| Byte | Key Description | Dial String |
|---|---|---|
| 0 | Feature Key 00 - Service Menu | <ESC>KA |
| 1 | Feature Key 01 - Redial | <ESC>KB |
| 2 | Feature Key 02 - Mute | <ESC>KC |
| 3 | Feature Key 03 - Speaker | <ESC>KD |
| 4 | Feature Key 04 - General Call | <ESC>KE |
| 5 | Feature Key 05 - Send Message | <ESC>KF |
| 6 | Feature Key 06 - Consultation | <ESC>KG |
| 7 | Feature Key 07 - Programmable | <ESC>KH |
| 8 | Feature Key 08 - Programmable | <ESC>KI |
| 9 | Feature Key 09 - Programmable | <ESC>KJ |
| 10 | Feature Key 10 - Programmable | <ESC>KK |

| Byte | Key Description | Dial String |
|---|---|---|
| 11 | Feature Key 11 - Release | <ESC>KL |

**Table 12. Optiset E Direct Key Dialing Strings for Feature Keys with Hicom 300E**

| Byte | Key Description | Dial String |
|---|---|---|
| 0 | Feature Key 00 - Mailbox | <ESC>KA |
| 1 | Feature Key 01 - Callback | <ESC>KB |
| 2 | Feature Key 02 -<br>(Configure to dial MWL_ON) | <ESC>KC |
| 3 | Feature Key 03 - Redial<br>(Configure to dial MWL_OFF) | <ESC>KD |
| 4 | Feature Key 04 - Programmable | <ESC>KE |
| 5 | Feature Key 05 - Programmable | <ESC>KF |
| 6 | Feature Key 06 - Consultation | <ESC>KG |
| 7 | Feature Key 07 - Line | <ESC>KH |
| 8 | Feature Key 08 - Line | <ESC>KI |
| 9 | Feature Key 09 - Line | <ESC>KJ |
| 10 | Feature Key 10 - Line | <ESC>KK |
| 11 | Feature Key 11 - General Call<br>(Indicates when the phone is off-hook) | <ESC>KL |

## ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LED Indicators on a given channel on a PBX Integration board. The data placed in the application buffer is shown below. If the data for byte 1 is 0x01, the rectangular indicator for Feature Key 1 is on. Refer to the *PBX Integration board Software Reference for* more information about using the **d42_indicators( )** function.

| | Feature Key 00 | Feature Key 01 | Feature Key 02 | Feature Key 03 | Feature Key 04 | Feature Key 05 | Feature Key 06 | Feature Key 07 | Feature Key 08 | Feature Key 09 | Feature Key 10 | Feature Key 11 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | 00 | 00 | *00* | *00* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### Pressing Feature Keys

The PBX Integration board can "press" any of the Optiset E Feature Keys using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference for* more information about dialing programmable keys. Each Feature Key on the Optiset E telephone is assigned a dial string sequence (refer to *Table 11* and *Table 12*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Feature Key.

### 4.3.4. Alphanumeric Display

The alphanumeric display is a two row, 48-character LCD that is used to show the activity of the phone. Some examples are:

- date and time
- feature names
- text messages
- error messages
- called/calling identification
- phone status
- line selection

The data used to display information in the LCD alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature). The PBX Integration board can retrieve the information on its alphanumeric display using the **d42_display( )** function. The function places the display data (48 bytes) in an application buffer. Refer to the *PBX Integration board Software Reference for* more information about using the **d42_display( )** function.

### ■ Example

An application uses the **dx_dial( )** function and the appropriate dial string to press keys dial extension number 1045. The **d42_display( )** function is used to retrieve the display data and place it in an application buffer (shown below). The information for the top row (first 24 characters) of the display is checked. Data in bytes 00 through 03 indicate that extension 1045 is being dialed.

| | 1 | 0 | 4 | 5 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data byte | 01 | 00 | 04 | 05 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |

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

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

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set processes the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.

Both the calling and called number IDs can be retrieved using the **d42_gtcallid( )** function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference for* more information about using **d42_gtcallid( )** function.

The contents of the called/calling number ID are shown in *Table 13* (as seen by the receiver of the call).

**Table 13. Called/Calling Number ID Data for the Hicom**

| Call Route | Called/Calling Number ID Data |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_display( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

### ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 1 | _ | 2 | 2 | 4 | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 31 | 5F | 32 | 32 | 34 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.3.5. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting Indicator (on or off) on another extension using the **dx_dial( )** function and the appropriate dial string, as described in the *PBX Integration board Software Reference* for your particular operating system.

**NOTE:** Message Waiting can also be set using the dx_dial() function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the dx_dial() function as described so that your application will maintain functionality across different manufacturers' switches.

#### MWI On

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCO><extention><ESCO> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCO> means Escape character followed by O.

#### MWI Off

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**

2) Call the **dx_dial()** function. The dial string is <ESCF><extention><ESCF> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

**NOTE:** <ESCF> means Escape character followed by F.

With the Hicom 150E PBX, the PBX Integration board can determine the state of its Message Waiting display using the **d42_display( )** function to retrieve the display data. Bytes 00 through 47 are used for the message waiting prompt and displays *Messages received: 1* and *View messages?* Refer to the *PBX Integration board Software Reference for* more information about using the **d42_display( )** function.

With the Hicom 300E PBX, the PBX Integration board can determine the state of its Message Waiting Indicator using the **d42_indicators( )** function to retrieve the LED Indicators data. Byte 00 contains the Message Waiting indicator status (0x00 is off; 0x01 is on). Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

## ■ Example

With the Hicom 150E, an application uses the **d42_display( )** function to retrieve the display data for a specified channel on the *PBX Integration board* to determine if a message is waiting , as shown in Figure 7 below.

**NOTE:** Bytes 00-23 represent the top row of the display. Bytes 24-47 represent the bottom row of the display.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 4D | 65 | 73 | 73 | 61 | 67 | 65 | 73 | 20 | 72 | 65 | 63 | 65 | 69 | 76 | 65 |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 64 | 3A | 00 | 31 | 20 | 20 | 20 | 20 | 56 | 69 | 65 | 77 | 20 | 6D | 65 | 73 |
| byte | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 73 | 61 | 67 | 65 | 73 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 3E |
| byte | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

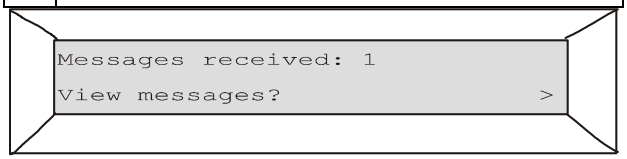



**Figure 7. Optiset E Message Waiting Display with Hicom 150E**

With the Hicom 300E, an application uses the **d42_indicators( )** function to retrieve the LED Indicators data for a specified channel on the PBX Integration board to determine if a message is waiting. The LED indicators data is shown below. The data 0x01 shows that the MWI indicator is on (there are messages waiting).

| | Feautre Key 01 | Feautre Key 02 | Feautre Key 03 | Feautre Key 04 | Feautre Key 05 | Feautre Key 06 | Feautre Key 07 | Feautre Key 08 | Feautre Key 09 | Feautre Key 10 | Feautre Key 11 | Feautre Key 12 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### 4.3.6. Transferring a Call

The PBX Integration board can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string; however, you should use the &,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (Refer to the *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer*). When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

#### ■ Example

An application answers a call and plays a greeting message prompting the caller to enter the extension she wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If the extension is busy or not answered, the application reconnects to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

## 4.4. MITEL SUPERSWITCH PBXs

MITEL PBXs use digital signaling to control its station sets and digitized voice. Digital Network Interface Circuit (DNIC) Line Cards provide an interface between the station sets and the switch.

The PBX Integration board has four or eight channels that are connected to a MITEL DNIC Line Card. The PBX Integration board can be used with the SX-

50, SX-200ML, and SX-2000 PBXs. These MITEL PBXs have many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)
- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable Feature Keys
- called/calling number identification
- call forwarding.

### 4.4.1. MITEL SUPERSWITCH Programming Requirements

There are specific switch programming requirements for using a PBX Integration board with a MITEL SUPERSWITCH. You must ensure that these features are set exactly (and assigned to the right keys) so that the PBX Integration board and the Unified API function correctly.
The PBX uses Class of Service (COS) to determine which features are available to an extension. The features available to an extension are shown in the telephone set's LCD Features display. Any feature not in the COS will not be displayed.

### MITEL SX Requirements for using MWI

If you are using a MITEL SX and wish to use the set Message Waiting Indicators (MWI) feature, the PBX Integration board must enable Auxiliary Attendant capabilities, and a line key must be set to act as the Attendant Console MWI. To configure MWI on a MITEL SUPERSWITCH:

- Enable Auxiliary Attendant capabilities
- Configure Personal Key 02 (see *Figure 8*) to act as to act as an Attendant Console Message Waiting Indicator key.

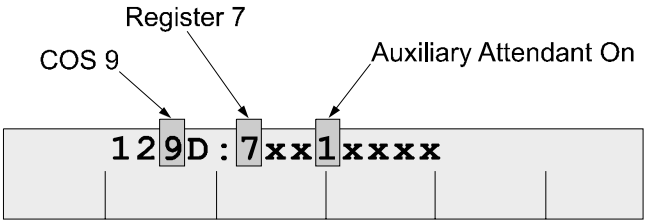
See MITEL manuals for more information on programming a SUPERSWITCH.

#### Configure a COS to have enhanced Auxiliary Attendant capabilities

1. From an attendant console, enter Programming Mode.
2. Enter the Command Number corresponding to the COS to which you want to add Auxiliary Attendant capabilities. Use commands 121 through 129

for COS 1 through COS 9. For example, if you want to change COS 9, use Command Number 129.

3. Set register 7, field "d" (Auxiliary Attendant Position) to 1 (enable Auxiliary Attendant Position).for the desired COS (1 - 9). The illustration below shows the Auxiliary Attendant feature enabled on COS 9.




4. Exit the Programming Mode.

For more information, see the MITEL SUPERSWITCH manuals.

#### Programming a Personal Key to act as an Attendant Console MWI key

1. On a 420/430 Superset phone, press the SuperKey.
2. Press the "No" Display Key until the display screen shows "Personal Keys," then press the "Yes" Display Key.
3. Press Personal Key 02, which is the third key from the bottom right.
4. If the screen shows that Personal Key 02 is programmed differently than as the MWI, press the "Change" Display Key.
5. Press the "No" Display Key until the screen shows: "Att. func keys," then press the "Yes" Display Key.
6. After the screen shows "Dial feature No," use the keypad to enter the number "10" for message waiting.
7. After the display screen shows "10 = msg wait," press the "Save" Display Key to confirm and exit.

To determine the current setting of a Feature key, press the SUPERKEY and then press the Feature Key you want to check. The display shows the name of the feature programmed.

### 4.4.2. Using the PBX Integration Board

The PBX Integration board performs functions available to a SUPERSET 400 Series telephone sets. As shown in *Figure 8*, a SUPERSET 420 telephone set uses two LCD displays to show line status (next to the line keys) and user prompts (above the soft keys):

- transfer calls
- set the message waiting indicator
- read the LCD alphanumeric display
- read the LCD features display
- read the LCD prompts display
- read LCD line indicators
- read the calling number ID
- press keys.

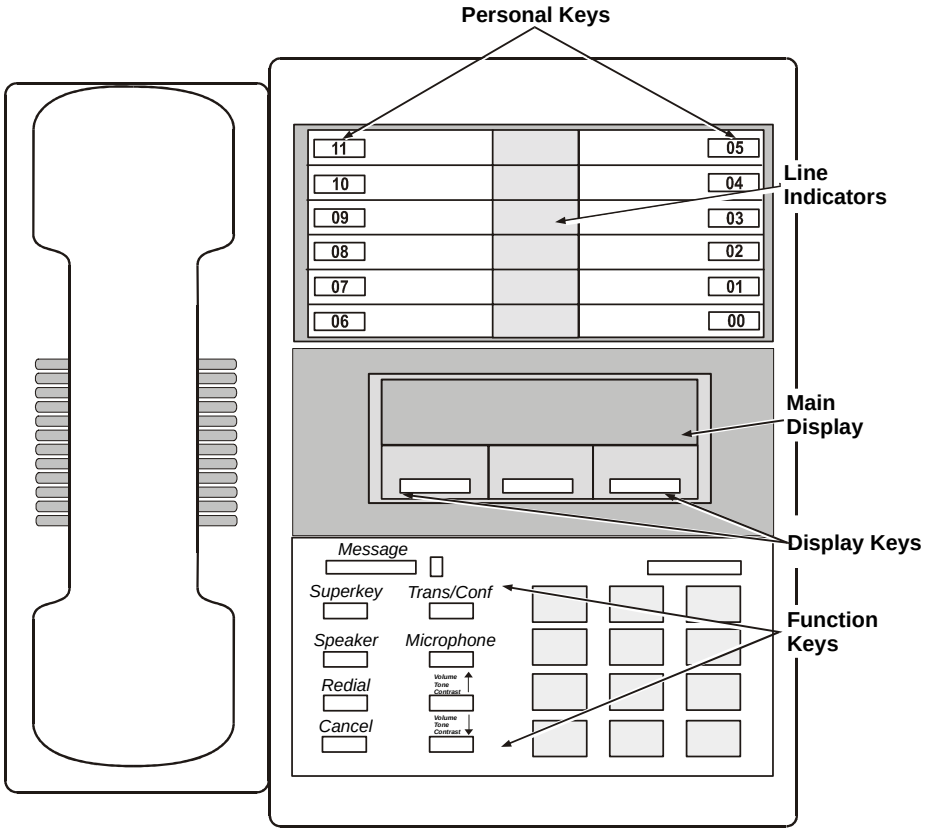



**Figure 8. MITEL SUPERSET 400 Series Telephone**

### 4.4.3. Programmable Personal Keys

As seen in *Figure 8*, there are 12 Personal Keys located on the top-right portion of SUPERSET 400 telephones. Some of these keys are configured when the PBX is programmed to select preassigned lines. Keys that are not configured can be defined by the user (using the telephone set or the PBX Integration board) as speed dial or Feature Keys. There is an LCD Line Indicator associated with each Personal Key. The LCD Indicators are triangular and can take on one of the six states listed in *Table 14*.

**Table 14. MITEL SUPERSET 400 Series LCD Line Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |
| unknown | 0x05 |

### Reading LCD Line Indicators

The PBX Integration board can determine the state of its Line Indicators by using the **d42_indicators( )** function to retrieve the LCD Indicators data. This function places the Line Indicator data (12 bytes) in an application buffer. Bytes 0-11 contain the indicator status for Feature Keys 00-11, respectively (see *Table 15*).

**Table 15. MITEL SUPERSET 400 Series LCD Line Indicators and Dial Strings**

| Byte | Key Description | Dial String |
|---|---|---|
| 00 | Personal Key 00 | <ESC>KA |
| 01 | Personal Key 01 | <ESC>KB |
| 02 | Personal Key 02 - Message Waiting | <ESC>KC |
| 03 | Personal Key 03* | <ESC>KD |
| 04 | Personal Key 04* | <ESC>KE |
| 05 | Personal Key 05* | <ESC>KF |
| 06 | Personal Key 06* | <ESC>KG |
| 07 | Personal Key 07* | <ESC>KH |
| 08 | Personal Key 08* | <ESC>KI |
| 09 | Personal Key 09* | <ESC>KJ |
| 10 | Personal Key 10* | <ESC>KK |

| Byte | Key Description | Dial String |
|---|---|---|
| 11 | Personal Key 11* | <ESC>KL |

*line, speed dial, or feature access

## ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LCD Line Indicators for a given channel on a PBX Integration board. The data placed in the application buffer is shown below. If the data for byte 07 is 0x02, the indicator corresponding to the Feature Key 07 is indicating ringing (see *Figure 9*). Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

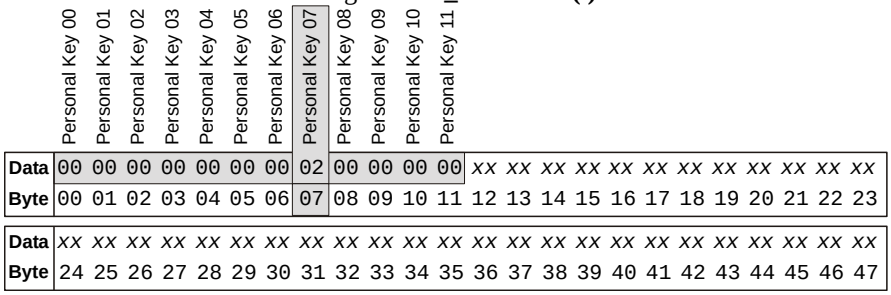



**Figure 9. MITEL SUPERSET LCD Line Indicator**

### Pressing Personal Keys

The PBX Integration board can "press" any of the MITEL SUPERSET Personal Keys using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Personal Key on the MITEL SUPERSET telephone is assigned a dial string sequence (refer to *Table 15*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Personal Key.

### 4.4.4. Function Keys

As shown in *Figure 8*, there are a number of Function Keys found to the left of the dial key pad on the MITEL SUPERSET telephones. The PBX Integration board can emulate these keys to perform various operational functions.

### Pressing Function Keys

The PBX Integration board can "press" any of its function key using the **dx_dial( )** function. Each function key on SUPERSET 400 series telephones is assigned a dial string sequence (refer to *Table 16* and *Table 17* ). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can dial any of its function keys. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**Table 16. MITEL SUPERSET 420 Direct Key Dialing Strings for Function Keys**

| Dial String | Key Description |
|---|---|
| <ESC>KM | Message Key |
| <ESC>KN | SuperKey |
| <ESC>KO | Cancel |
| <ESC>KP | Microphone |
| <ESC>KQ | Hold |
| <ESC>KR | Redial |
| <ESC>KS | Speaker |
| <ESC>KT | Trans/Conf |
| <ESC>KU | V/T/C up |
| <ESC>KV | V/T/C down |

**Table 17. MITEL SUPERSET 430 Direct Key Dialing Strings for Function Keys**

| Dial String | Key Description |
|---|---|
| <ESC>KM | Message Key |
| <ESC>KN | SuperKey |
| <ESC>KO | *Not Used* |
| <ESC>KP | Microphone |
| <ESC>KQ | Hold |
| <ESC>KR | Applications |
| <ESC>KS | Speaker |

| | |
|---|---|
| <ESC>KT | *Not Used* |
| <ESC>KU | V/T/C up |
| <ESC>KV | V/T/C down |

### 4.4.5. Display Keys

As shown in *Figure 8*, there are three Display Keys or Soft Keys located below the LCD display on the MITEL SUPERSET 420 telephone. These keys are associated with specific prompts shown on the LCD display depending on the current state of the phone.

**NOTE:** This information applies to the SUPERSET 420 only. The *PBX Integration board* cannot detect the Display Key prompts on the SUPERSET 430. The *PBX Integration board* can, however, "press" the Display Keys on both telephones. See *Pressing Display Keys* below.

#### Reading Display Key Prompts

The PBX Integration board can determine which of its prompts are currently displayed by using the **d42_display( )** function to retrieve display data and read the information for the bottom row (last 16 characters). The total length of the display data is 32 bytes. The data location for the Display Key prompts is as follows:

| | |
|---|---|
| Display Key 00 | bytes 16 - 20 |
| Display Key 01 | bytes 21 - 26 |
| Display Key 02 | bytes 27 - 31 |

Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

#### ■ Example

An application uses the **d42_display( )** function to retrieve the prompt data displayed for Display Key 00, as shown in *Figure 10*. The data placed in the application buffer is shown below. Data in bytes 16 through 31 indicate that the prompts Yes and No are displayed below Display Keys 00 and 02, respectively.

**NOTE:** Bytes 00-15 represent the top row of the display. Bytes 16-31 represent the bottom row of the display.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 43 | 41 | 4C | 4C | 46 | 4F | 52 | 44 | 57 | 41 | 52 | 49 | 4E | 47 | 3F | 20 |
| byte | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |

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

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

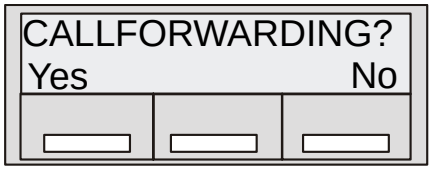



**Figure 10. MITEL SUPERSET 420 Display Keys**

### Pressing Display Keys

The PBX Integration board can respond to a prompt and "press" the appropriate Display Key using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. As shown in *Table 18* and *Table 19*, each Display Key on the SUPERSET 400 series telephones is assigned a dial string sequence. By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any of its Display Keys.

**Table 18. MITEL SUPERSET 420 Direct Key Dialing Strings for Display Keys**

| Dial String | Key Description |
|---|---|
| <ESC>Ka | Display Key 00 (left) |
| <ESC>Kb | Display Key 01 (middle) |
| <ESC>Kc | Display Key 02 (right) |

**Table 19. MITEL SUPERSET 430 Direct Key Dialing Strings for Display Keys**

| Dial String | Key Description |
|---|---|
| <ESC>Ka | Top Left Softkey |
| <ESC>Kb | Top Middle Softkey |
| <ESC>Kc | Top Right Softkey |
| <ESC>Kd | Bottom Left Softkey |
| <ESC>Ke | Bottom Middle Softkey |
| <ESC>Kf | Bottom Right Softkey |

### 4.4.6. Alphanumeric Display

The alphanumeric display is a 32-or 80-digit LCD that is used to show:

- date and time when the extension is idle
- SuperKey instructions and Softkey labels during programming and feature access
- call status
- messaging information
- telephone system error messages
- saved numbers (speed dialing)
- saved number for redial
- timed reminder setting
- call forward type and destination
- calling number ID
- trunk line ID

The data used to display information in the LCD alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature). The PBX Integration board can retrieve the information on its display using the **d42_display( )** function. The function places the display data (32 or 80 bytes) in an application buffer. Refer to the PBX Integration board *Software Reference* for more information about using the **d42_display( )** function.

### ■ Example

An application uses the **dx_dial( )** function to press the "SuperKey" key and "Display Key 1" for "Yes" on a specified channel on the PBX Integration board to display the call forwarding extension. The **d42_display( )** function is then used to retrieve the display data and verify that a call forwarding extension has not been programmed. The display data is shown below.

| | N | O | N | E | | A | C | T | I | V | E | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 4E | 4F | 4E | 45 | 20 | 41 | 43 | 54 | 49 | 56 | 45 | 20 | 20 | 20 | 20 |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| | P | r | o | g | r | a | m | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 50 | 72 | 6F | 67 | 72 | 61 | 6D | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

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

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set *processes* the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.

Both the calling and called number IDs can be retrieved using the **d42_gtcallid( ) )**function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference* for more information about using **d42_gtcallid( )** function.

The contents of the called/calling number ID are shown in *Table 20* (as seen by the receiver of the call).

**Table 20. Called/Calling Number ID Data for the Hicom**

| Call Route | Called/Calling Number ID Data |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_display( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

### ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 1 | _ | 2 | 2 | 4 | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 31 | 5F | 32 | 32 | 34 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.4.7. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting Indicator (ON or OFF) on another extension using the **dial( )** function and the appropriate dial string, as described in the *PBX Integration board Software Reference* for your particular operating system.

**NOTE:** Message Waiting can also be set using the dx_dial() function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the dx_dial() function as

described so that your application will maintain functionality across different manufacturers' switches.

#### MWI On

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCO><extention><ESCO> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCO> means Escape character followed by O.

#### MWI Off

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCF><extention><ESCF> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCF> means Escape character followed by F.

The PBX Integration board can determine the state of its Message Waiting Indicator using the **d42_indicators( )** function to retrieve the LED Indicators data. Byte 13 contains the Message Waiting indicator status (0x00 is off; 0x01 is on). Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

### ■ Example

An application uses the **d42_indicators( )** function to retrieve the LED Indicators data for a specified channel on the *PBX Integration board* to determine if a message is waiting. The indicators data is shown below. The data 0x01 shows that the MWI indicator is on (there are messages waiting).

| | Personal Key 00 | Personal Key 01 | Personal Key 02 | Personal Key 03 | Personal Key 04 | Personal Key 05 | Personal Key 06 | Personal Key 07 | Personal Key 08 | Personal Key 09 | Personal Key 10 | Personal Key 11 | MWI | MIC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### 4.4.8. Transferring a Call

The *PBX Integration board* can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string to press the Trans/Conf soft key; however, you should use the &,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (refer to *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer).* When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

#### ■ Example

An application answers an incoming call and plays a greeting message prompting the caller to enter the extension they wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If the extension is busy or not answered, the application reconnects

to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

## 4.5. Nortel Norstar

The Norstar product family includes the Compact version and the expandable Modular model. The PBX Integration board can be used with either of these switches. The PBXs use digital signaling to control their station sets and digitized voice. PBXs use plug-in station modules to connect to station sets, and trunk modules to connect to trunk lines.
The PBX Integration board has either four or eight channels that are connected directly to a station module in a Nortel Norstar. The switch has many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)
- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable Feature Keys
- called/calling number identification
- call forwarding.

### 4.5.1. Nortel Norstar Programming Requirements

There are specific switch programming requirements for using a PBX Integration board with a Nortel Norstar. You must ensure that these features are set exactly (and assigned to the right keys) so that the PBX Integration board and the Dialogic APIs function correctly.
*Table 21* lists the menu structure used when configuring a Nortel Norstar (with DR5 or later revision installed). The shaded areas indicate the actual menu items to change in order to use the KSU with a PBX Integration board. For details about programming a Norstar KSU, refer to the appropriate Norstar manual.

The table only shows the configuration for one trunk line (001) and one extension (221). If you are using more than one trunk line, configure each trunk line the same. If you are using more than one extension, ensure that all the extensions are configured the same with the exception of the *Forward on busy* and *Forward no answer* options. For these menu items, the first extension should be forwarded to the second extension and the second extension should be forwarded to the third extension, and so on. The last extension should be forwarded back to the first extension.

**Table 21. Norstar Configuration Requirements (DR5)**

| Menu Option/Default Value | New Value |
|---|---|
| A-Configuration | |
| 1. Trk/Line Data | |
| a) Show line: *Enter Trunk #* | 001 |
| b) Trunk data | |
| Line001: Loop* | |
| Trunk mode:Unspr | |
| Dial mode: Pulse | Tone |
| Full AutoHold:N | |
| c) Line data | |
| Line type: Public | PoolA |
| Prime set: 21 | 221* |
| Aux. ringer: Y | |
| Auto privacy:Y | |
| 2. Line Access | |
| a) Show set: *Enter extension* | 221 |
| b) Line assignment *(no changes required)* | |
| c) Answer DNs *(no changes required)* | |
| d) Ringing *(no changes required)* | |
| e) Line pool access *(no changes required)* | |
| f) Intercom keys: | 1 |
| g) Prime line: None | I/C |
| 3. Call Handling | |
| a) Held reminder:N | |
| b) DRT to prime: Y | N |
| c) Trnsfr callbk: 3 | 12 |
| d) Park prefix:1 | |
| e) Park timeout:45 | |

* Extension number assignment is system dependent.

**Table 21. Norstar Configuration Requirements (DR5) - (Cont.)**

| Menu Option/Default Value | New Value |
|---|---|
| f) Camp timeout:45 | |
| g) Directed pickup:Y | |
| h) On hold:Tones | |
| 4. Miscellaneous *(no changes required)* | |
| 5. System Data *(no changes required)* | |
| B-General admin | |
| 1. Sys speed dial *(no changes required)* | |
| 2. Names *(no changes required)* | |
| 3. Time and date *(no changes required)* | |
| 4. Direct-Dial *(no changes required)* | |
| 5. Capabilities | |
| a) Dialing filters *(no changes required)* | |
| b) Rem access pkgs *(no changes required)* | |
| c) Set abilities | |
| Show set: *Enter extension* | 221 |
| (1) Set filter:02 | |
| (2) Line/set filters *(no changes req'd)* | |
| (3) Set lock:None | |
| (4) Full handsfree: N | Y |
| (5) Auto handsfree: N | Y |
| (6) HF answerback: Y | N |
| (7) Pickup group:NO | |
| (8) Paging: Y | N |
| (9) Paging zone: 1 | NO |
| (10) Aux. ringer:N | |
| (11) Direct-dial:Set1 | |
| (12) Forward on busy | |
| (a) Forward to: None | 222 |
| (13) Forward no answr | |
| (a) Forward to: None | 222 |
| (b) Forward delay: 3 | 2 |
| (14) Allow redirect:N | |

**Table 21. Norstar Configuration Requirements (DR5) - (Cont.)**

| Menu Option/Default Value | New Value |
|---|---|
| (15) Redirect ring:Y | |
| (16) Hotline:None | |
| (17) Priority call:N | |
| d) Line abilities *(no changes required)* | |
| e) COS passwords *(no changes required)* | |
| 6. Service Modes | |
| a) Control sets | |
| Show line: *Enter line #* | 001 |
| (1) Line001: | 221 |
| (2) Line002: | 221 |
| through | |
| (3) Line008: | 221 |
| (4) Name1:Night | |
| (a) Setting:Manual | |
| (b) Trunk answer:Y | N |
| (c) Extra-dial: | 221 |
| (5) Name2:Evening | |
| (a) Setting:Manual | |
| (b) Trunk answer:Y | N |
| (c) Extra-dial: | 221 |
| (6) Name3:Lunch | |
| (a) Setting:Manual | |
| (b) Trunk answer:Y | N |
| (c) Extra-dial:221 | |
| 5. Password *(no changes required)* | |
| 6. Log Defaults *(no changes required)* | |
| 7. Call Services *(no changes required)* | |
| C-Set copy *(no changes required)* | |
| D-Maintenance *(no changes required)* | |

Memory Keys 00 through 03 must be programmed as follows:

| | |
|---|---|
| Memory Button 00 | Handsfree/mute |
| Memory Button 01 | Intercom |
| Memory Button 02 | Intercom |
| Memory Button 03 | Transfer (Feature 70) |

To determine the current setting of a Memory Button, press

Feature ✻ **0**

then press the Memory Button you want to check. The display shows the name of the feature programmed.

Memory Button 00 is automatically assigned as the Handsfree/mute key when Full Handsfree is set to Y [refer to *Table 21*, B. 5. (c) (4)]. Memory Button 01 is automatically set as the Intercom key when the number of intercom keys is set to 1 [refer to *Table 21*, A. 2. (f)]. To assign Memory Button 03 to Transfer press:

Feature ✻ **3** Key 03 Feature **7** **0**

### 4.5.2. Using the PBX Integration Board

The PBX Integration board performs functions available to a M7324 telephone set (see *Figure 11*). An M7324 telephone set uses three LCD displays. Two is used to show key status indicators (between the line keys), while the other display is used for user prompts and messages (above the display keys). The PBX Integration board can:

- transfer calls
- set the message waiting indicator
- read the LCD display
- read LCD indicators
- read the called/calling number ID
- press keys.

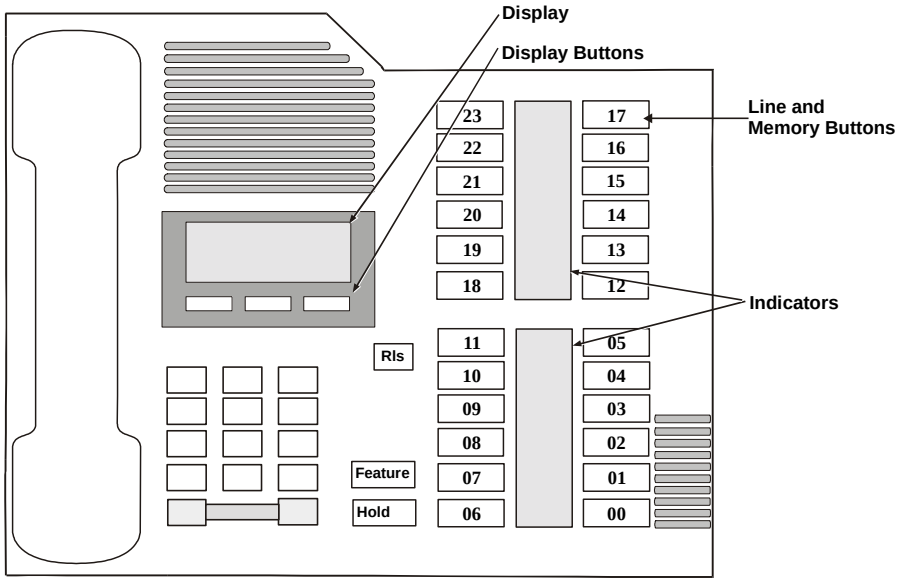



**Figure 11. Nortel M7324 Telephone**

### 4.5.3. Programmable Memory Keys

As illustrated in *Figure 11*, the M7324 has 24 Programmable Memory Keys located to the right of the display . These keys are configured either during PBX installation or by the user (using the telephone set or the PBX Integration board). The Line, Intercom, Answer, and Handsfree keys are assigned during PBX configuration and cannot be user programmed. There is an LCD Indicator associated with each Memory Button. The LCD Indicators are triangular and can take on one of the six states listed in *Table 22*.

**Table 22. M7324 LCD Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |

| | |
|---|---|
| unknown | 0x05 |

### Reading LCD Indicators

The PBX Integration board can determine the state of its LCD Indicators by using the **d42_indicators( )** function to retrieve the LCD Indicators data. This function places the Line Indicator data in an application buffer. For a M7324, bytes 0-23 contain the indicator status for Memory Keys 00-23, respectively (see *Table 23*).

**Table 23. M7324 Direct Key Dialing Strings for Memory Keys**

| Byte | Key Description | Dial String |
|---|---|---|
| 00 | Memory Button 00 | <ESC>K0 |
| 01 | Memory Button 01 | <ESC>K1 |
| 02 | Memory Button 02 | <ESC>K2 |
| 03 | Memory Button 03 | <ESC>K3 |
| 04 | Memory Button 04 | <ESC>K4 |
| 05 | Memory Button 05 | <ESC>K5 |
| 06 | Memory Button 06 | <ESC>K6 |
| 07 | Memory Button 07 | <ESC>K7 |
| 08 | Memory Button 08 | <ESC>K8 |
| 09 | Memory Button 09 | <ESC>K9 |
| 10 | Memory Button 10 | <ESC>KS |
| 11 | Memory Button 11 | <ESC>KT |
| 12 | Memory Button 12 | <ESC>KU |
| 13 | Memory Button 13 | <ESC>KV |
| 14 | Memory Button 14 | <ESC>KW |
| 15 | Memory Button 15 | <ESC>KX |
| 16 | Memory Button 16 | <ESC>KY |
| 17 | Memory Button 17 | <ESC>KZ |
| 18 | Memory Button 18 | <ESC>Ka |

| Byte | Key Description | Dial String |
|---|---|---|
| 19 | Memory Button 19 | <ESC>Kb |
| 20 | Memory Button 20 | <ESC>Kc |
| 21 | Memory Button 21 | <ESC>Kd |
| 22 | Memory Button 22 | <ESC>Ke |
| 23 | Memory Button 23 | <ESC>Kf |

### ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LCD Indicators on a given channel on a PBX Integration board. In the M7324 example shown below, data has been placed in the application buffer. If the data for byte 1 is 0x01, the triangular indicator for Memory Button 1 is on. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

| | Memory Button 00 | Memory Button 01 | Memory Button 02 | Memory Button 03 | Memory Button 04 | Memory Button 05 | Memory Button 06 | Memory Button 07 | Memory Button 08 | Memory Button 09 | Memory Button 10 | Memory Button 11 | Memory Button 12 | Memory Button 13 | Memory Button 14 | Memory Button 15 | Memory Button 16 | Memory Button 17 | Memory Button 18 | Memory Button 19 | Memory Button 20 | Memory Button 21 | Memory Button 22 | Memory Button 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### Pressing Memory Keys

The PBX Integration board can "press" any of the M7324 Memory Keys using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Memory Button on the M7324 telephone is assigned a dial string sequence (refer to *Table 23*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Memory Button.

### 4.5.4. Display Keys

As shown in *Figure 11*, there are three Display Keys located below the LCD display. These keys are associated with specific prompts shown on the LCD

display depending on the current state of the phone (shown on the bottom row of the LCD display).

### Reading Display Key Prompts

The PBX Integration board can determine which of its prompts are currently displayed by using the **d42_display( )** function to retrieve display data and read the information for the bottom row (last 16 characters). The total length of the display data is 32 bytes. The data location for the Display Keys is as follows:

| | |
|---|---|
| Display Key 00 | bytes 16 - 20 |
| Display Key 01 | bytes 22 - 26 |
| Display Key 02 | bytes 28 - 31 |

Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

### ■ Example

An application uses the **d42_display( )** function to retrieve the prompt data displayed for Display Key 00, as shown in *Figure 12*. The data placed in the application buffer is shown below. Data in bytes 16 through 20 indicate that the prompt **EXIT** is displayed below Display Key 00.

**NOTE:** Bytes 00-15 represent the top row of the display. Bytes 16-31 represent the bottom row of the display.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 50 | 72 | 65 | 73 | 73 | 20 | 61 | 20 | 62 | 75 | 74 | 74 | 6F | 6E | 20 | 20 |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | **7** | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 45 | 58 | 49 | 54 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

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

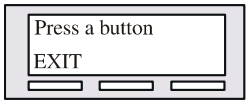



**Figure 12. M7324 Display Keys**

### Pressing Display Keys

The PBX Integration board can respond to a prompt and "press" the appropriate Display Key using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Display Key on the M7324 telephone is assigned a dial string sequence (refer to *Table 24*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any of its Display Keys.

**Table 24. M7324 Direct Key Dialing Strings for Display Keys**

| Dial String | Key Description |
|---|---|
| <ESC>KP | Display Key 00 (left) |
| <ESC>KQ | Display Key 01 (middle) |
| <ESC>KR | Display Key 02 (right) |

### 4.5.5. Alphanumeric Display

The alphanumeric display is a two row, 32-digit LCD that is used to show the activity of the phone. Some examples are:

- date and time
- feature names
- error messages
- called/calling identification
- phone status
- line selection
- Display Key prompts.

The data used to display information in the LCD alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature). The PBX Integration board can retrieve the information on its alphanumeric display using the **d42_display( )** function. The function places the display data

(32 bytes) in an application buffer. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

### ■ Example

An application uses the **dx_dial( )** function and the appropriate dial string (ESC>KN, <ESC>KK, <ESC>KA, <ESC>K3) to press keys to display which feature is assigned to Memory Button 03. Then, the **d42_display( )** function is used to retrieve the display data and place it in an application buffer (shown below). The information for the top row (first 16 characters) of the display is checked. Data in bytes 00 through 15 indicate that *Transfer* is assigned to Memory Button 03.

| | T r a n s f e r | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 54 | 72 | 61 | 6E | 73 | 66 | 65 | 72 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| byte | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |

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

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

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set *processes* the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.

Both the calling and called number IDs can be retrieved using the **d42_gtcallid( )** function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference* for more information about using **d42_gtcallid( )** function.

The contents of the called/calling number ID are shown in *Table 25* (as seen by the receiver of the call).

**Table 25. Called/Calling Number ID Data for the Nortel Norstar**

| **Call Route** | Called/C**alling Number ID Data** |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_display( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

### ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 1 | _ | 2 | 2 | 4 | | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 31 | 5F | 32 | 32 | 34 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.5.6. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting display (on or off) on another extension using the **dx_dial( )** function and the appropriate dial string. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** Message Waiting can also be set using the dx_dial() function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the dx_dial() function as described so that your application will maintain functionality across different manufacturers' switches.

#### MWI On

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCO><extention><ESCO> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCO> means Escape character followed by O.

#### MWI Off

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1) Go Off Hook using **dx_sethook()**
2) Call the **dx_dial()** function. The dial string is <ESCF><extention><ESCF> (Optional pause character may be used)
3) Go On hook using the **dx_sethook()** again

<ESCF> means Escape character followed by F.

The PBX Integration board can determine the state of its Message Waiting display using the **d42_display( )** function to retrieve the display data. Bytes 00 through 15 are used for the message waiting prompt and will display *Message for you*. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

### ■ Example

An application uses the **d42_display( )** function to retrieve the display data for a specified channel on the *PBX Integration board* to determine if a message is waiting (see *Figure 13*). The display data is shown below.

**NOTE:** Bytes 00-15 represent the top row of the display. Bytes 16-31 represent the bottom row of the display.

| data | 4D | 65 | 73 | 73 | 61 | 67 | 65 | 00 | 66 | 6F | 72 | 00 | 79 | 6F | 75 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| data | 4D | 53 | 47 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| byte | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

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

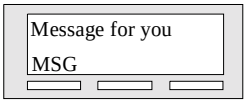



**Figure 13. M7324 Message Waiting Display**

### 4.5.7. Transferring a Call

The PBX Integration board can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string (<ESC>KN, <ESC>KH, <ESC>KA; or <ESC>KN70) to press Feature 70. This method does not depend on Memory Button 03 being programmed correctly; however, you should use the &,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (Refer to the *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer).* When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

### ■ Example

An application answers a call and plays a greeting message prompting the caller to enter the extension they wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If the extension is busy or not answered, the application reconnects to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

## 4.6. Nortel Meridian 1

The Nortel Meridian 1 is a full-featured PBX that can provide thousands of ports and many PBX voice and data features. The Meridian 1 uses digital signaling to control its station sets and digitized voice. The PBX uses plug-in station modules to connect to station sets, and trunk modules to connect to trunk lines.

The PBX Integration board has either four or eight channels that are connected directly to a station module in a Meridian 1. The switch has many standard features that are supported by the PBX Integration board, such as:

- direct inward dialing (DID)
- hands free operation
- speed dialing
- hunt groups
- message waiting indication
- user programmable Feature Keys
- called/calling number identification
- call forwarding.

### 4.6.1. Nortel Meridian 1 Programming Requirements

There are specific switch programming requirements for using a PBX Integration board with a Meridian 1. You must ensure that these features are set exactly (and assigned to the right keys) so that the PBX Integration board and the Unified API function correctly.

*Table 26* lists the menu structure used when configuring a Nortel Meridian 1. For details about programming a Meridian 1, refer to the appropriate Meridian 1 manual.

The M-1 ports should be configured as a M2616 telephone with a display as follows:

**Table 26. Nortel Meridian 1 Configuration Requirements**

| Menu Option | Value |
|---|---|
| CLS | CTD FBD WTA MTD FNA HTA ADD HFD MWA CNDA |
| TYPE | 2616 |
| HUNT | (5502)IS NEXT PHONE IN GROUP |

| | |
|---|---|
| LHK | 1 |
| KEY 0 | SCR 5501 (Ringing Call Appearance) |
| KEY 1 | |
| KEY 2 | |
| KEY 3 | TRN (TRANSFER) |
| KEY 4 | MCK (MESSAGE CANCEL) |
| KEY 5 | MIK (MESSAGE INDICATION) |
| KEY 6 | |
| KEY 7 | |
| KEY 8 | |
| KEY 9 | |
| KEY 10 | |
| KEY 11 | |
| KEY 12 | |
| KEY 13 | |
| KEY 14 | |
| KEY 15 | |

### 4.6.2. Using the PBX Integration Board

The PBX Integration board performs functions available to a M2616 telephone set (see *Figure 14*). An M2616 telephone set uses two LCD displays to show key status (between the line keys) and user prompts and messages (above the display keys). The PBX Integration board can:

- transfer calls
- set the message waiting indicator
- read the LCD display
- read LCD indicators
- read the called/calling number ID
- press keys.

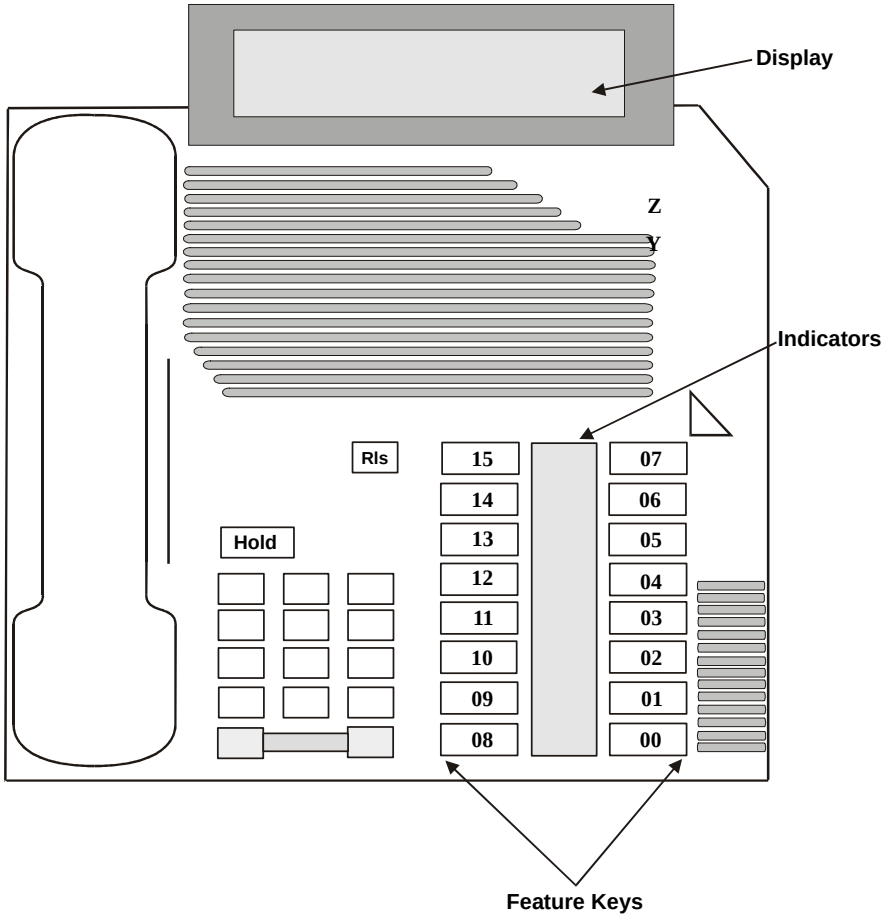



**Figure 14. Nortel M2616 Telephone**

### 4.6.3. Programmable Feature Keys

As illustrated in *Figure 14*, there are 16 Programmable Feature Keys located below the display on the M2616 telephone. These keys are configured either during PBX installation or by the user (using the telephone set or the PBX Integration board). The Line, Program, and Handsfree keys are assigned during PBX configuration and cannot be user programmed. There is an LCD Indicator associated with each Feature Key. The LCD Indicators are triangular and can take on one of the six states listed in *Table 27*.

**Table 27. M2616 LCD Indicator States**

| State | Value (Hex) |
|---|---|
| off | 0x00 |
| on | 0x01 |
| ringing | 0x02 |
| hold | 0x03 |
| error | 0x04 |
| unknown | 0x05 |

### Reading LCD Indicators

The PBX Integration board can determine the state of its LCD Indicators by using the **d42_indicators( )** function to retrieve the LCD Indicators data. This function places the Line Indicator data (16 bytes) in an application buffer. Bytes 00-15 contain the indicator status for Feature Keys 00-15, respectively (see *Table 28*).

**Table 28. M2616 Direct Key Dialing Strings for Feature Keys**

| Byte | Key Description | Dial String |
|---|---|---|
| 0 | Feature Key 00 | <ESC>KA |
| 01 | Feature Key 01 | <ESC>KB |
| 02 | Feature Key 02 | <ESC>KC |
| 03 | Feature Key 03 - Transfer | <ESC>KD |
| 04 | Feature Key 04 | <ESC>KE |
| 05 | Feature Key 05 | <ESC>KF |

| Byte | Key Description | Dial String |
|---|---|---|
| 06 | Feature Key 06 | <ESC>KG |
| 07 | Feature Key 07 - Program | <ESC>KH |
| 08 | Feature Key 08 | <ESC>KI |
| 09 | Feature Key 09 | <ESC>KJ |
| 10 | Feature Key 10 | <ESC>KK |
| 11 | Feature Key 11 | <ESC>KL |
| 12 | Feature Key 12 | <ESC>KM |
| 13 | Feature Key 13 | <ESC>KN |
| 14 | Feature Key 14 | <ESC>KO |
| 15 | Feature Key 15 | <ESC>KP |

## ■ Example

An application uses the **d42_indicators( )** function to retrieve the current data for the LCD Indicators on a given channel on a PBX Integration board. The data placed in the application buffer is shown below. If the data for byte 1 is 0x01, the triangular indicator for Feature Key 1 is on. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

| | Feature Key 00 | Feature Key 01 | Feature Key 02 | Feature Key 03 | Feature Key 04 | Feature Key 05 | Feature Key 06 | Feature Key 07 | Feature Key 08 | Feature Key 09 | Feature Key 10 | Feature Key 11 | Feature Key 12 | Feature Key 13 | Feature key 14 | Feature key 15 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *00* | *00* | *00* | *00* | *00* | *00* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **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 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### Pressing Feature Keys

The PBX Integration board can "press" any of the M2616 Feature Keys using the **dx_dial( )** function. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys. Each Feature Key on the M2616 telephone is assigned a dial string sequence (refer

to *Table 28*). By using the **dx_dial( )** function and the appropriate dial string, the PBX Integration board can press any Feature Key.

### 4.6.4. Alphanumeric Display

The alphanumeric display is a two row, 48-digit LCD that is used to show the activity of the phone. Some examples are:

- date and time
- feature names
- error messages
- called/calling identification
- phone status
- line selection

The data used to display information in the LCD alphanumeric display is in ASCII format. When the telephone is not in use, the display normally shows the date and time. The content of the display is changed automatically (e.g., receiving an incoming call, making an outgoing call, or activating a feature). The PBX Integration board can retrieve the information on its alphanumeric display using the **d42_display( )** function. The function places the display data (48 bytes) in an application buffer. Refer to the *PBX Integration board Software Reference* for more information about using the **d42_display( )** function.

#### ■ Example

An application uses the **dx_dial( )** function and the appropriate dial string to press keys dial extension number 1045. Then, the **d42_display( )** function is used to retrieve the display data and place it in an application buffer (shown below). The information for the top row (first 24 characters) of the display is checked. Data in bytes 00 through 05 indicate that extension 1045 is being dialed.

| data byte | 61 | 32 | 01 | 00 | 04 | 05 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |

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

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

### Called/Calling Number ID (within the PBX)

When receiving a call on a PBX Integration board from another extension, the PBX sends calling number ID data (by default, the extension number of the telephone placing the call) to the station set between the first and second rings. The station set *processes* the data and sends an ID message to the display. The calling number ID data sent from the PBX to the station set differs from the calling number ID data presented on the display.

When placing a call to another extension, the called number ID (by default, the extension of the telephone being called) is shown in the display.
Both the calling and called number IDs can be retrieved using the **d42_gtcallid( )** function. The **d42_gtcallid( )** function retrieves the called/calling number ID message sent from the PBX to the station set, not the data sent to the display. Refer to the *PBX Integration board Software Reference* for more information about using **d42_gtcallid( )** function.
The contents of the called/calling number ID are shown in *Table 29* (as seen by the receiver of the call).

**Table 29. Called/Calling Number ID Data for the Meridian 1**

| Call Route | Called/Calling Number ID Data |
|---|---|
| Call received from trunk line 1 | _0-1 |
| Call received from station set 221 | _221 |
| Call originally received on trunk line 1, then transferred to station set 223 | 223_0-1 |
| Call originally received by extension 221, then forwarded to extension 224 | 224_221 |

**NOTE:** The called/calling number ID can also be obtained using the **d42_display( )** function; however, you should use the **d42_gtcallid( )** function so that your application will maintain functionality across different manufacturers' switches.

## ■ Example

An application uses the **d42_gtcallid( )** function to retrieve the calling number ID for a call received on a specified channel on a PBX Integration board. The calling number ID data and corresponding ASCII values are shown below.

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text | *bb* | 2 | 2 | 1 | _ | 2 | 2 | 4 | | | | | | | | | | | | | | | | |
| data | 20 | 32 | 32 | 31 | 5F | 32 | 32 | 34 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| text | | | | | | | | | | | | | | | | | | | | | | | | |
| data | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *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 |

### 4.6.5. Setting the Message Waiting Indicator

The PBX Integration board can set the Message Waiting Indicator (on or off) on another extension using the **dx_dial( )** function and the appropriate dial string. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** Message Waiting can also be set using the **dx_dial()** function and appropriate dial string to press the Feature Key assigned to send messages; however, you should use the **dx_dial()** function as described so that your application will maintain functionality across different manufacturers' switches.

**MWI On**

The recommended technique to turn on the MWI in this switch, using **dx_dial()** with the dial string is:

1. Call the **dx_dial()** function.

   The dial string is <ESCO>**,**<extention>**,**<ESCO>

<ESCO> means Escape character followed by O.

**MWI Off**

The recommended technique to turn off the MWI in this switch, using **dx_dial()** with the dial string is:

1. Call the **dx_dial()** function.

   The dial string is <ESCF>**,**<extention>**,**<ESCF>

<ESCF> means Escape character followed by F.

We strongly recommend using the pause character (comma) in the dial string for MWI manipulation for Nortel Meridian switch in order to avoid unpredictable result under load.

The PBX Integration board can determine the state of its Message Waiting Indicator using the **d42_indicators( )** function to retrieve the LED Indicators data. Byte 16 contains the Message Waiting indicator status (0x00 is off; 0x01 is on). Refer to the *PBX Integration board Software Reference* for more information about using the **d42_indicators( )** function.

#### ■ Example

An application uses the **d42_indicators( )** function to retrieve the LED Indicators data for a specified channel on the PBX Integration board to determine if a message is waiting. The LED indicators data is shown below.

The data 0x00 shows that the MWI indicator is off (there are no messages waiting).

| | Feature Key 00 | Feautre Key 01 | Feature Key 02 | Feature Key 03 | Feature Key 04 | Feature Key 05 | Feature Key 06 | Feature Key 07 | Feature Key 08 | Feature Key 09 | Feature Key 10 | Feature Key 11 | Feature Key 12 | Feature Key 13 | Feature Key 14 | Feature Key 15 | MWI | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **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 |
| **Data** | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* | *xx* |
| **Byte** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

### 4.6.6. Transferring a Call

The PBX Integration board can transfer calls using the **dx_dial( )** function. By using the **dx_dial( )** function and the appropriate dial string (&,<extension>), the PBX Integration board can transfer a call to any extension connected to the switch. Refer to the *PBX Integration board Software Reference* for more information about dialing programmable keys.

**NOTE:** The transfer function can be performed using the **dx_dial( )** function and the appropriate dial string; however, you should use the &,<extension> dial string so your application will maintain functionality across different manufacturers' switches.

The *PBX Integration board* can perform both supervised and blind transfers (Refer to the *Sections 2.1. Supervised Call Transfer* and *2.2. Blind Call Transfer*). When a blind transfer is performed, the PBX controls where the call is routed if the called extension is busy or does not answer. When a supervised transfer is performed, your application can implement call progress analysis and called/calling number ID to intelligently control where the call is routed and what type of message is played if the called extension is busy or does not answer. Because of this capability, supervised transfer is the preferred method.

#### ■ Example

An application answers a call and plays a greeting message prompting the caller to enter the extension they wish to reach (the caller enters 221). Using the **dx_dial( )** function with the dial string (&,221), the application attempts to transfer (supervised) the call to extension 221. Call progress analysis is used to determine if extension 221 is answered, busy, or there is no answer. If extension 221 answers, the application hangs up and the transfer is complete. If

the extension is busy or not answered, the application reconnects to the incoming call and plays a message asking the caller to choose between accessing voice mail or transferring to the operator.

# Appendix A
## Technical Specifications

**PBX Integration board Technical Specifications***

| | |
|---|---|
| Number of ports/card | 8 |
| Total ports/system | 64 |
| Max. boards/system | 8 |
| Microprocessor | Intel 80486GXSF running at 28.5 MHz with 2MB DRAM |
| Digital signal processor | Motorola DSP56303 (Onyx) @ 100 MHz, 24-bit |
| DSP SRAM | 256K SRAM |
| **Host Interface** | |
| Bus compatibility | PCI |
| Bus speed | 33 MHz |
| Shared memory | 64 KB SRAM configured as two 32K x 16 |
| Base addresses | D0000 (default) |
| Interrupt level | One IRQ is shared by all PBX Integration boards. |
| **Telephone Interface** | |
| Support | Lucent 7434 (4-wire), Lucent 8434 (2-wire), Siemens ROLMphone 400, Siemens Optiset E, MITEL Superset 420, MITEL Superset 430, Nortel M7324, Nortel M2616 |
| Connectors | 36-position mini D cable plug |
| **Power Requirements** | |
| +5 VDC | 3.3 A at 5 volts per board |
| Operating temperature | 0ºC to +50ºC |
| Storage temperature | -20ºC to +70ºC |
| Humidity | 8% to 80% noncondensing |
| **Form Factor** | |
| 5V PCI long form factor card. 12.283 in. long. and 4.200 in. high | |

| **Safety & EMI Certifications** | |
|---|---|
| United States | FCC part 68 does not apply |
| Canada | CSO3 does not apply |

### PBX Integration board Firmware Specifications*

| **Audio Signal** | |
|---|---|
| Transmit | -12.5 dBm0 (weighted average)** |
| Receive range | -42 to +2.5 dBm |
| Silence detection | -38 dBm0, software adjustable** |
| Frequency response | 24 Kb/s: 300 Hz to 2600 Hz ±3 dB<br>32 Kb/s: 300 Hz to 3400 Hz ±3 dB<br>48 Kb/s: 300 Hz to 2600 Hz ±3 dB<br>64 Kb/s: 300 Hz to 3400 Hz ±3 dB |
| **Audio Digitizing** | |
| Method | G.711 A-law and µ-law PCM; GSM 610; G.726 |
| Sampling rates | 6 kHz, 8 kHz for PCM |
| Data rates | G.711 A-law and µ-law PCM: 48 Kb/s, 64 Kb/s; |
| **Tone Dialing:** | |
| DTMF digits | 0 to 9, *, #, A, B, C, D |
| MF digits | 0 to 9, KP, ST, ST1, ST2, ST3 |
| Level | Network compatible |
| Rate | 10 digits/s maximum, software adjustable |
| **Pulse Dialing** | |
| 10 digits | 0 to 9 |
| Pulsing rate | 10 pulses/s, nominal |
| Break ratio | 60% |
| **DTMF Tone Detection:** | |
| DTMF digits | 0 to 9, *, #, A, B, C, D per Bellcore LSSGR Sec 6 |
| Dynamic range | -39 dBm0 to +0 dBm0 per tone** |
| Minimum tone duration | 32 ms, software adjustable |
| Acceptable twist: | 10 dB |
| Signal/noise ratio | 10 dB (referenced to lowest amplitude tone) |

| | |
|---|---|
| Talk off | Detects 0 digits while monitoring MITEL speech tape #CM7291. Detects less than 10 digits while monitoring Bellcore TR-TSY-000763 standard speech tapes (LSSGR requirements specify detecting no more than 470 total digits). |
| **MF Tone Detection:** | |
| MF digits | 0 to 9, KP, ST, ST1, ST2, ST3 |
| **Speed Control** | |
| Pitch controlled | Available for 24 and 32 Kb/s data rates |
| Adjustment range | 50% |
| **Volume Control** | |
| Adjustment range | 40 dB, with programmer-definable increments |

* All specifications are subject to change without notice.
**Analog levels: 0 dBm0 corresponds to a level of +3dBm at tip-ring analog point.

## System Requirements

- Minimum 90 MHz Pentium™5- or the equivalent Celeron®-based platform with an available PCI bus slot for an 8-port system. The host system must provide a CPU of Pentium or Celeron class at 266 MHz speed or higher for a 64-port system, including eight available PCI slots. The PBX Integration board occupies a single expansion slot, and up to eight boards can be configured in a system, with each board sharing the same interrupt level. The maximum number of ports supported is 64, dependent on the application, the amount of disk I/O required, and the host computer's CPU. The computer must run the Windows NT or Windows 20000 operating system.

# Glossary

**Analog Signal** A continuously variable signal. Voice signals on telephone lines are usually analog (i.e., transmitted electronically in a form analogous to the spoken form). A representation of an analog signal is a sine wave.

**Attendant** The "operator" of a phone system console. Usually directs incoming calls to the proper person or department. May also assign outgoing lines or trunks. The operator may be a person or an automated system.

**Automatic call distribution** A system used to systematically distribute incoming calls to a number of operators (called agents). Agents are usually sales or service people.

**Call Forwarding** A service which allows a call to be directed to an extension other than the one that was dialed. This is accomplished by the called party programming into the phone system the extension the incoming calls should be forwarded to.

**DID** Direct Inward Dialing - The capability to dial an extension (inside the PBX system) without going through the attendant.

**Digital Signal** A discontinuous signal. One whose state consists of discrete elements representing specific information. Logically, a digital signal can be thought of as a pattern of ones and zeros representing a specific value.

**Handset** the part of the telephone held in the hand. Contains a transmitter and a receiver.

**Hold** Temporarily leave a phone call without disconnecting. You can return to it at any time.

**Hunt** The process of a call reaching a group of lines. If the first line is busy, it will be forwarded to the second line. If the second line is busy, it will be forwarded to the third line, and so on.

**Hybrid System** A term used to describe a telephone system that has attributes of both Key Systems and PBXs. Usually means that incoming

lines (trunks) appear on the phone set, and outbound calls require the use of an access code (typically a "9").

**KTS** Key Telephone System - A telephone system in which the station sets have multiple keys permitting the user to select outgoing or incoming CO phone lines. You do not have to dial an access code (typically "9") to access CO lines.

**KSU** Key Service Unit - The main cabinet which contains all the electronics to run a Key Telephone System.

**LCD** Liquid Crystal Display - An alphanumeric display using liquid crystals sealed between two pieces of glass. Usually a gray background with black characters.

**LED** Light Emitting Diode - A diode which emits light. Can be used as a single indicator or combined with other LEDs to create an alphanumeric display.

**Line Card** A plug-in electronic printed circuit board for a PBX or KSU that operates lamps, ringing, holding, and other features associated with several telephone lines.

**On-hook** When the handset is resting in its cradle. The phone is not connected to any line.

**Off-hook** When the handset is lifted from its cradle. Alerts the CO (or PBX) that it is ready (usually ready to receive a dial tone).

**On-hook Dialing** A feature that allows the caller to dial without lifting the handset. After dialing, the caller can listen to the progress of the call through the built-in speaker.

**PBX** Private Branch Exchange - A private phone system allowing communications within a business and between the business and the outside world. Outside lines are not accessible to the station set. An access code (typically "9") is required to connect to an outside line.

**Speakerphone** A telephone that has a speaker and a microphone for hands-free conversation

**Station Set** A telephone used with a PBX or KTS.

**TDM** Time Division Multiplex - A technique used for transmitting separate data, voice, or video messages simultaneously over one phone line by interleaving elements of each message in fast time sequences.

**Tip and Ring** Another way of saying plus and minus, or positive and ground, in electrical circuits.

**Trunk** A telephone communication path or channel between two points, one being a CO and the other a PBX or KSU.

# Index

## I



## K



## L



## M



## N



## P



## S

## T



## U



## V