Voice Features Guide
for Windows NT

05-0168-007

COPYRIGHT NOTICE

Publication Date: December, 1997
Dialogic Corporation
1515 Route Ten
Parsippany, NJ 07054

Table of Contents

List of Tables

List of Figures

1. Introduction to Voice Features

This chapter provides an overview of the voice features. The following major features of the voice software for Windows NT are described in this guide:

- Call Analysis
- Global Tone Detection and Global Tone Generation
- R2MF Signaling
- Speed and Volume Control
- Demonstration Programs

1.1. Call Analysis Overview

Call Analysis is used to monitor the progress of a call after dialing into the Public Switched Telephone Network (PSTN). There are two forms of Call Analysis, Basic Call Analysis and PerfectCall Call Analysis. PerfectCall Call Analysis uses an improved method of signal identification and can detect fax machines and answering machines. Basic Call Analysis provides backward compatibility for older applications written before PerfectCall Analysis became available.

Call Analysis is initiated using the **dx_dial( )** function which uses input from the Call Analysis Parameter (DX_CAP) data structure.

Call Analysis is available on all voice boards.

See *Chapter* 2. Call Analysis for detailed information about Call Analysis.

1.2. Global Tone Detection/Generation

Global Tone Detection (GTD) and Global Tone Generation (GTG) are available on all voice boards.

1.2.1. Global Tone Detection

Global Tone Detection allows a voice board to detect single or dual frequency tones other than DTMF tones 0-9, a-d, *, and #. Through GTD, a user can define the characteristics of a tone in order to detect a tone with the same characteristics.

The characteristics of a tone can be defined and tone detection can be enabled using GTD functions provided in the Voice Library.

See *Chapter 3. Global Tone Detection/Generation* for detailed information about Global Tone Detection.

1.2.2. Global Tone Generation

Global Tone Generation permits the creation of user-defined tones using the TN_GEN template data structure and allows the user to play the tone using the **dx_playtone( )** function. The **dx_bldtngen( )** can be used to build the TN_GEN template.

See *Chapter* 3. Global Tone Detection/Generation for detailed information about Global Tone Generation.

1.3. R2MF Signaling

R2MF signaling is an international signaling system that is used in Europe and Asia to permit the transmission of numerical and other information relating to the called and calling subscribers' lines.

See *Chapter 4. R2MF Signaling* for a description of the aspects of R2MF signaling and how to use R2MF signaling with the voice boards.

1.4. Speed and Volume Control

Speed and Volume control are available on D/21D, D/21E, D/41D, D/41E, D/41ESC, D/81A, D/121B, D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1, and D/320SC boards only. Do not use the Speed and Volume control functions to control speed on the D/120, D/121, or D/121A boards.

Several functions and data structures can be used to control the speed and volume of play on a channel. See *Chapter* 5. Speed and Volume Control for instructions about using the functions and data structures to control play speed and play volume.

1.5. SCbus Overview

The SCbus is a real-time, high speed, time division multiplexed (TDM) communications bus connecting Signal Computing System Architecture (SCSA), voice, telephone network interface and other technology resource boards together. SCbus boards are treated as board devices with on-board voice and/or telephone network interface devices that are identified by a board and channel (time slot for digital network channels) designation, such as a voice channel, analog channel, or digital channel.

For more information on the SCbus and SCbus routing, refer to the *SCbus Routing Guide* and the *SCbus Routing Function Reference for Windows NT*.

1.6. Voice Product Terminology

The following product naming conventions are used throughout this guide:

D/2x refers to any model of the Dialogic DIALOG/ series of 2-channel voice-store-and-forward expansion boards. This series includes **D21D** and **D21E** boards.

D/4x refers to any model of the Dialogic DIALOG/ series of 4-channel voice store-and-forward expansion boards. This series includes the **D41/D**, **D/41E**, and **D/41ESC** boards.

D/12x refers to any model of the Dialogic series of 12-channel voice-store-and-forward expansion boards. **D/120**, **D/121**, **D/121A**, and **D/121B** are specific models of this board.

D/81A refers to the Dialogic 8-channel voice-store-and-forward expansion board.

D/160SC-LS refers to the Dialogic 16-channel voice board with onboard analog loop start interface.

D/240SC refers to the Dialogic 24-channel voice board for use with a network interface board.

D/240SC-T1 refers to the Dialogic 24-channel voice board with onboard T-1 digital interface.

D/300SC-E1 refers to the Dialogic 30-channel voice board with onboard E-1 digital interface.

D/320SC refers to the Dialogic 30-channel voice board for use with a network interface board.

D/xxx refers to D/2x, D/4x, D/81A and D/12x expansion boards.

D/xxxSC refers to voice and telephone network interface resource boards that communicate via the SCbus. These boards include **D/41ESC**, **D/160SC-LS**, **D/240SC**, **D/240SC-T1**, **D/300SC-E1**, and **D/320SC**.

DIALOG/HD refers to voice and telephone network interface resource boards that communicate via the SCbus. These boards include **D/160SC-LS**, **D/240SC**, **D/240SC-T1**, **D/300SC-E1**, and **D/320SC**.

DTI/xxx refers to any of Dialogic's digital telephony interface expansion boards for the AT-bus architecture. These boards include: **DTI/211** and **DTI/212** boards.

DTI/xxxSC refers to telephone network interface resource boards that communicate via the SCbus. These boards include the **DTI/240SC, DTI/241SC, DTI/300SC,** and **DTI/301SC** boards.

FAX/xxx refers to Dialogic's FAX resource expansion boards. **FAX/120** is a 12-channel model that connects to a D/121A or D/121B board.

Firmware Load File refers to the firmware file that is downloaded to a Voice board. This file has a *.fwl* extension.

LSI refers to Dialogic's PEB-based loop start interface expansion boards. The **LSI/120** is a specific model of this board. **LSI/80-int** refers to the international versions of Dialogic's loop start interface expansion boards.

PEB is the PCM expansion bus connecting the D/81A or D/12x voice boards to the network interface boards.

SCbus is the TDM (Time Division Multiplexed) bus connecting SCSA (Signal Computing System Architecture) voice, telephone network interface and other technology resource boards together.

SpringBoard refers to the hardware platform used with the D/21D, D/41D, D/21E, D/41E, D/81A, D/121, D/121A, and D/121B board.

SpringWare refers to the software algorithms built into the downloadable firmware that provides the voice processing features available on all Dialogic voice boards.

VFX/40ESC is a Dialogic SCbus voice and FAX resource board with on-board loop-start interfaces. The **VFX/40ESC** board provides 4-channels of enhanced voice and FAX services in a single slot.

Voice hardware and software refers to D/2x, D/4x, D/81A, D/12x, and D/xxxSC expansion boards and associated software.

1.7. Organization of This Voice Features Guide

The *Voice Software Reference for Windows NT* describes the voice features provided with the voice software for Windows NT.

Chapter *1. Introduction to Voice Features* provides an overview of the voice features.

Chapter 2. Call Analysis describes Call Analysis, including how to use the DX_CAP Call Analysis structure.

Chapter *3. Global Tone Detection/Generation* describes Global Tone Detection and Global Tone Generation.

Chapter 4. R2MF Signaling describes the R2MF technology and Dialogic's support for R2MF.

Chapter *5. Speed and Volume Control* describes how to control play-speed and play-volume.

Chapter 6. Voice Features Demonstration Programs describes how to use the demonstration programs.

Appendix A provides a list of related Dialogic publications.

A **Glossary** and an **Index** are also provided.

2. Call Analysis

This chapter provides a description of Call Analysis. Call Analysis is available as a standard feature on all Voice boards.

2.1. What Is Call Analysis?

Call Analysis is used to determine the progress of a call after dialing into the Public Switched Telephone Network (PSTN), where a wide variety of signal possibilities can occur.

Call Analysis monitors the progress of an outbound call after dialing, which allows you to process the call based on the outcome.

By using Call Analysis you can determine the following:

- if the line is answered and, in many cases, how the line is answered
- if line rings but is not answered
- if line is busy
- if there is a problem in completing the call

There are two forms of Call Analysis, Basic Call Analysis and PerfectCall Call Analysis. PerfectCall Call Analysis uses an improved method of signal identification, and can also detect fax machines and answering machines. Basic Call Analysis provides backward compatibility for older applications; any application which was written before PerfectCall Analysis became available will continue to work unchanged. However, it is recommended that all new applications be designed for PerfectCall Call Analysis.

Call Analysis is initiated when a call is dialed using the **dx_dial( )** function. This function uses input from the Call Analysis Parameter structure (DX_CAP). You can adjust the DX_CAP parameters to fit the needs of your application. When the Voice Driver determines the outcome of the call, information is returned using Extended Attribute functions. To find out how to use Call Analysis see *Section* 2.5. How to Use Call Analysis.

2.2. What Does Call AnalysisDetect?

Call Analysis determines the outcome of the call from among the following possibilities:

Intercept	A Special Information Tone (SIT) was detected; an invalid number was dialed or there was a problem completing the call. This is also known as an *operator intercept*.
No Ringback	No discernable signal pattern was detected.
Connect	The phone was answered.
No Answer	The line was ringing but was not answered.
Busy	A busy signal was detected.

The outcome of the call is returned to the application when Call Analysis has completed.

2.3. How Does Call Analysis Work?

Call Analysis uses the following techniques to determine the progress of the call:

- Cadence Detection
- Frequency Detection
- Loop Current Detection
- Positive Voice Detection and Positive Answering Machine Detection

The following figures illustrate these processes.

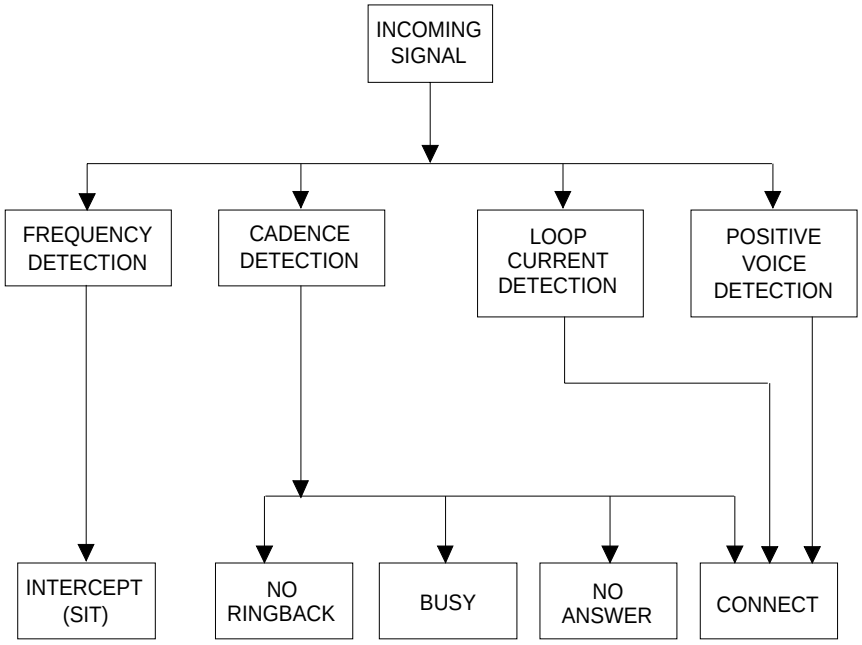

Figure 1. Basic Call Analysis Components

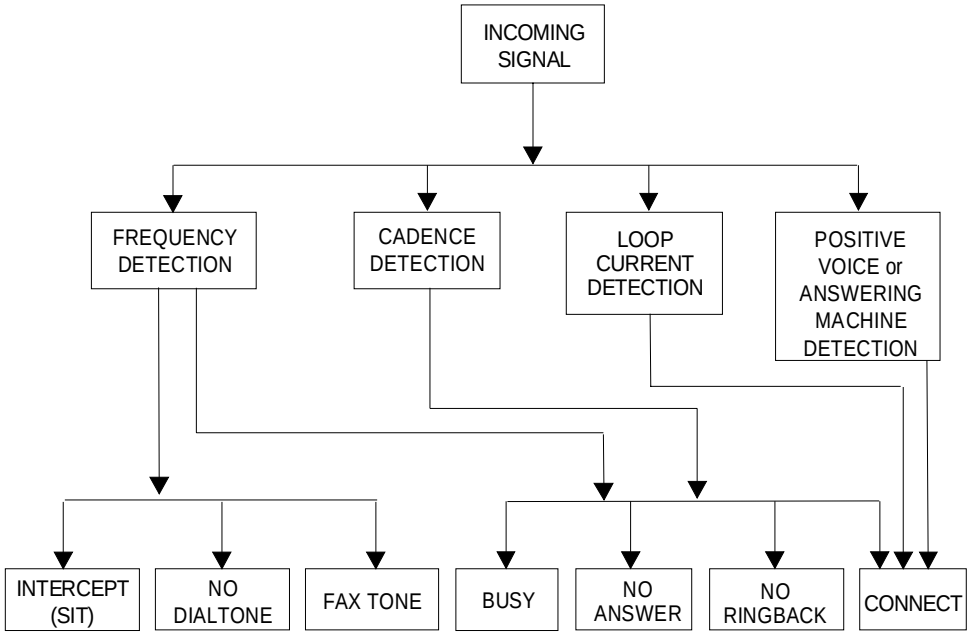

Figure 2. PerfectCall Call Analysis Components

Frequency Detection, Cadence Detection, Loop Current Detection, Positive Voice Detection, and Positive Answering Machine Detection can all operate simultaneously during Call Analysis. See *Section* 2.6. How the DX_CAP Controls Call Analysis for more information.

The figures show that Cadence Detection is the sole means of detecting a no ringback, busy, or no answer when using Basic Call Analysis. PerfectCall Call Analysis uses Cadence Detection plus Frequency Detection to identify all of these signals plus fax machine frequencies. A connect can be detected through the complementary methods of Cadence Detection, Frequency Detection, Loop Current Detection, Positive Voice Detection, and Positive Answering Machine Detection (Positive Answering Machine Detection is available with PerfectCall Call Analysis only).

2.4. How to Enable PerfectCall Call Analysis

To enable PerfectCall Call Analysis on a specified channel, perform the following steps:

NOTE: This procedure needs to be followed only once per channel; thereafter, any outgoing calls made using **dx_dial( )** will benefit from PerfectCall Call Analysis.

1. Make any desired modifications to the default dial tone, busy tone, fax tone, and ringback signal definitions. The **dx_chgfreq( )**, **dx_chgdur( )**, and **dx_chgrepcnt( )** functions make these modifications.
2. Execute the **dx_initcallp( )** function to activate PerfectCall Call Analysis. PerfectCall Call Analysis stays active until **dx_deltones( )** is called.

   NOTE: To disable PerfectCall Call Analysis, call the **dx_deltones( )** function.

Call dx_deltones() prior to calling dx_initcallp() to clear all tone templates remaining on the channel. However, dx_deltones() deletes GTD tones added to the channel.

2.4.1. Modifying the Default Tone Definitions

PerfectCall Call Analysis makes use of Global Tone Detection (GTD) tone definitions for three different types of dial tones, two busy tones, one ringback tone, and two fax tones. The tone definitions specify the frequencies, durations and repetition counts necessary to identify each of these signals. Each signal may consist of a single tone or a dual tone.

The Voice Driver contains default definitions for each of these tones. The default definitions will allow applications to identify the tones correctly in most countries and for most switching equipment. If, however, a situation arises in which the default tone definitions are not adequate, three functions are provided to modify the standard tone definitions:

- **dx_chgfreq( )** specifies frequencies and tolerances for one or both frequencies of a single frequency or dual-frequency tone.

- **dx_chgdur( )** specifies the cadence (on time, off time, and acceptable deviations) for a tone.
- **dx_chgrepcnt( )** specifies the repetition count required to identify a tone.

These functions only change the tone definitions, they do notalter the behavior of PerfectCall Call Analysis. When the **dx_initcallp( )** function is invoked to activate PerfectCall Call Analysis on a particular channel (see the following section), it uses the current tone definitions to initialize that channel. Multiple calls to **dx_initcallp( )** may therefore use varying tone definitions, and several channels can operate simultaneously with different tone definitions.

Details on these functions may be found in the Function Reference chapter in the *Voice Programmer's Guide for Windows NT.*

2.4.2. Activating PerfectCall Call Progress

PerfectCall Call Analysis is activated on a per-channel basis. For each channel on which PerfectCall Call Analysis is desired, the function **dx_initcallp( )** must be called.

The **dx_initcallp( )** function initializes PerfectCall Call Analysis on the specified channel, using the current tone definitions for local dial tone, international dial tone, extra dial tone, two busy signals, ringback, and two fax tones. Once the channel is initialized with these tone definitions, this initialization cannot be altered. The only way to change the tone definitions in effect for a given channel is to issue a **dx_deltones( )** call for that channel, then invoke another **dx_initcallp( )** with different tone definitions.

Note that **dx_deltones( )** deletes *all* Global Tone Detection definitions for the given channel, and not just those involved with PerfectCall Call Analysis.

Refer to the Function Reference chapter in the *Voice Programmer's Guide for Windows NT* for more information on **dx_initcallp( )** and **dx_deltones( )**.

2.5. How to Use Call Analysis

The following procedure describes how to initiate an outbound call with Call Analysis:

1. Set up the Call Analysis Parameter structure (DX_CAP), which contains parameters that control the operation of Call Analysis.
2. Execute the **dx_dial( )** function to initiate Call Analysis.
   - **If running dx_dial( ) asynchronously**, use the Event Management functions to determine when dialing with Call Analysis is complete (TDX_CALLP termination event).
   - **If running dx_dial( ) synchronously**, wait for **dx_dial( )** to return a value greater than 0 to indicate successful completion.

   See the **dx_dial( )** function description in the *Voice Programmer's Guide for Windows NT* for information about asynchronous and synchronous operation.
3. Use **ATDX_CPTERM( )** to determine the outcome of the call:

an intercept	CR_CEPT
no ringback	CR_NORB
busy signal	CR_BUSY
no answer	CR_NOANS
fax machine	CR_FAXTONE
no dial tone	CR_NODIALTONE
connect	CR_CNCT
Call Analysis stopped	CR_STOPD
Call Analysis error	CR_ERROR

   NOTE: When running **dx_dial( )** synchronously, these results are also returned by **dx_dial( )**.
4. Obtain additional termination, frequency, or cadence information (such as the length of the salutation) as desired using Extended Attribute functions.

Each of these steps is described in detail in the following pages.

NOTE: For information about the following, see the indicated chapter in the *Voice Programmer's Guide for Windows NT*:

- **dx_dial( )**, see *Chapter 3.*
- **ATDX_CPTERM( )**, see *Chapter 3.*

- DX_CAP data structure, see *Chapter 4.*

2.5.1. Set Up the Call Analysis Parameter Structure (DX_CAP)

The **dx_dial( )** function enables Call Analysis after dialing. The parameters in the DX_CAP structure are used. To use the default values for DX_CAP specify NULL in the function.

If you want to customize the parameters for your environment, you must set up the Call Analysis Parameter structure before calling **dx_dial( )**. By adjusting the DX_CAP parameters, you can:

- Eliminate Call Analysis functions that do not pertain to your environment.
- Optimize performance of the required functions.
- Support non-standard system configurations.
- Perform additional functions, such as determining whether the called party is a business, residence, or answering machine.

To set up the DX_CAP structure for Call Analysis:

1. Execute the **dx_clrcap( )** function to clear the DX_CAP and initialize the parameters to 0. The value 0 indicates that the default value will be used for that particular parameter. **dx_dial( )** can also be set to run with default Call Analysis parameter values, by specifying a NULL pointer to the DX_CAP structure.
2. Set the parameter to another value if you do not want to use the default value for a given parameter.

For more detailed information on the DX_CAP block parameters, refer to the *Section* 2.6. How the DX_CAP Controls Call Analysis.

2.5.2. Use the dx_dial( ) Function to Initiate Call Analysis

Enable Call Analysis by calling **dx_dial( )** with the **mode** function argument set to DX_CALLP. Termination of dialing with Call Analysis is indicated differently depending on whether the function is running asynchronously or synchronously. More information on **dx_dial( )** can be found in *Chapter 3* in the *Voice Programmer's Guide for Windows NT.*

2.5.3. Determine the Outcome of the Call

Once **dx_dial( )** with Call Analysis has terminated, use the Extended Attribute function **ATDX_CPTERM( )** to determine the outcome of the call. **ATDX_CPTERM( )** will return one of the following Call Analysis Termination results:

CR_BUSY	Called line was busy.
CR_CNCT	Called line was connected.
CR_FAXTONE	Called line was answered by a fax machine or a modem (PerfectCall Call Analysis only).
CR_NOANS	Called line did not answer.
CR_NODIALTONE	Called line failed to produce a dial tone (PerfectCall Call Analysis only).
CR_NORB	Called line did not ring.
CR_CEPT	Called line received operator intercept (SIT). The Extended Attribute functions provide information on the detected frequencies and durations.
CR_STOPD	Call Analysis stopped due to **dx_stopch( )**.
CR_ERROR	Call Analysis error occurred. ATDX_CPERROR( ) returns the type of the Call Analysis error.

Figure 3 illustrates the possible outcomes of Call Analysis.
Figure 4 illustrates the possible outcomes of PerfectCall Call Analysis.

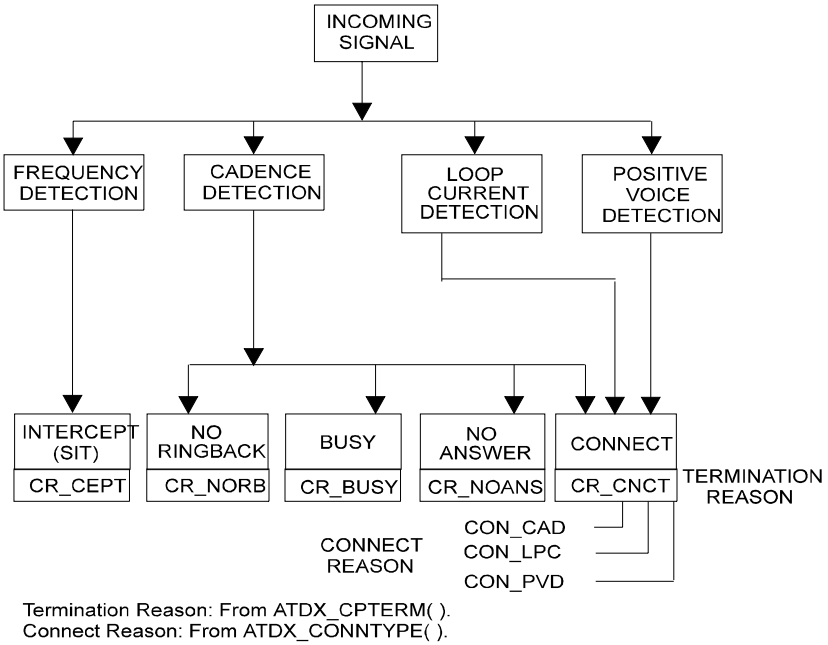

Figure 3. Call Analysis Outcomes for Basic Call Analysis

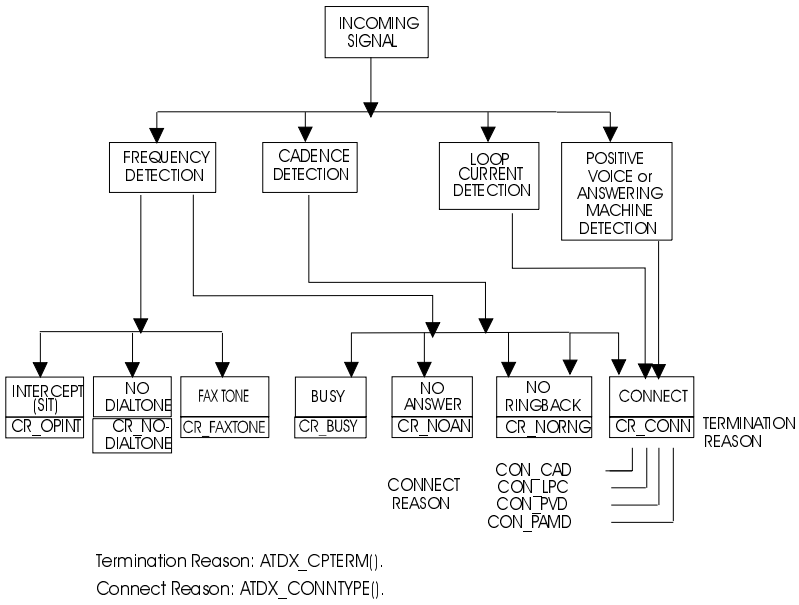

Figure 4. Call Analysis Outcomes for PerfectCall Call Analysis

2.5.4. Obtain Additional Call Outcome Information

Additional Call Analysis information can be retrieved using the following Extended Attribute functions:

ATDX_ANSRSIZ( )	• Returns duration of answer
ATDX_CPERROR( )	• Returns call analysis error
ATDX_CPTERM( )	• Returns last call analysis termination
ATDX_CONNTYPE( )	• Returns connection type
ATDX_CRTNID( )	• Returns the identifier of the tone that caused the

	most recent Call Analysis termination
ATDX_DTNFAIL( )	• Returns the dial tone character that indicates which dial tone Call Analysis failed to detect
ATDX_FRQDUR( )	• Returns duration of first frequency detected
ATDX_FRQDUR2( )	• Returns duration of second frequencydetected
ATDX_FRQDUR3( )	• Returns duration of third frequency detected
ATDX_FRQHZ( )	• Returns frequency detected in Hz of first detected tone
ATDX_FRQHZ2( )	• Returns frequency of second detected tone
ATDX_FRQHZ3( )	• Returns frequency of third detected tone
ATDX_LONGLOW( )	• Returns duration of longer silence
ATDX_FRQOUT( )	• Returns percent of frequency out of bounds
ATDX_SHORTLO( )	• Returns duration of shorter silence
ATDX_SIZEHI( )	• Returns duration of non-silence

For a discussion of how frequency and cadence information returned by these Extended Attribute functions relate to the DX_CAP parameters, refer to *Section* 2.6. How the DX_CAP Controls Call Analysis.

2.6. How the DX_CAP Controls Call Analysis

The following sections describe the DX_CAP parameters that control Frequency Detection, Cadence Detection, Loop Current Detection, Positive Voice Detection, and Positive Answering Machine Detection.

The DX_CAP structure, as defined in the header file, is listed in *Chapter 4* of the *Voice Programmer's Guide for Windows NT.*

2.6.1. Selecting SIT Frequency Detection, Positive Voice Detection, and Positive Answering Machine Detection

The Call Analysis Parameter structure (DX_CAP) parameter **ca_intflg** (intercept mode flag) is used to enable or disable Frequency Detection, Positive Voice Detection, and/or Positive Answering Machine Detection for Call Analysis.

The **ca_intflg** parameter also determines when SIT Frequency Detection should terminate and return an *intercept.* This can occur immediately upon detection of the specified frequencies or after waiting for a connect indicated by Cadence Detection, Loop Current Detection, Positive Voice Detection, or Positive Answering Machine Detection.

The following definitions are provided for use with the **ca_intflg** parameter:

ca_intflg Intercept Mode Flag: This parameter enables or disables SIT Frequency Detection, Positive Voice Detection (PVD), and/or Positive Answering Machine Detection (PAMD), and selects the mode of operation for Frequency Detection. Default: 1 (DX_OPTEN). The following modes are possible:

DX_OPTEN	Enable Frequency Detection (wait for a connect to terminate Call Analysis and return *an intercept*). If a valid frequency is detected, Call Analysis waits for detection of a connect using Cadence Detection or Loop Current Detection before returning an *intercept*.
DX_OPTDIS	Disable Frequency Detection and PVD.
DX_OPTNOCON	Enable Frequency Detection (terminate Call Analysis and return an *intercept* immediately upon detecting tones). Frequency Detection returns an *intercept* immediately after detecting a valid frequency.
DX_PVDENABLE	Enable PVD.
DX_PVDOPTEN	Enable PVD and DX_OPTEN.
DX_PVDOPTNOCON	Enable PVD and DX_OPTNOCON.
DX_PAMDENABLE	Enable PAMD.
DX_PAMDOPTEN	Enable PAMD and DX_OPTEN.

2.6.2. SIT Frequency Detection

SIT Frequency Detection operates simultaneously with all other Call Analysis detection methods. The purpose of Frequency Detection is to detect the tri-tone

Special Information Tone (SIT) sequences and other single-frequency tones. Detection of an SIT sequence indicates an operator intercept or other problem in completing the call.

SIT Frequency Detection can detect virtually any single-frequency tone below 2100 Hz and above 300 Hz.

Tri-Tone SIT Sequences

Tone information for the 4 SIT sequences is provided in Table 1. The frequencies are represented in Hz and the length of the signal is in 10 ms units.

Table 1. Special Information Tone Sequences

SIT Name	SIT Description	1st Tone Freq.	1st Tone Len.	2nd Tone Freq.	2nd Tone Len.	3rd Tone Freq.	3rd Tone Len.
NC	No Circuit Found	985	38	1429	38	1777	38
IC	Operator Intercept	914	27	1371	27	1777	38
VC	Vacant Circuit	985	38	1370	27	1777	38
RO	Reorder (system busy)	9148	27	1429	38	1777	38

The length of the first tone is not dependable; often it is shortened or cut.

Setting Tri-Tone Frequency Detection Parameters

Frequency Detection on voice boards is designed to detect all three tones in the tri-tone SIT sequence. To detect all three tones in the SIT sequence, you must specify the frequency detection parameters in the DX_CAP for all three tones in the sequence.

To detect all four tri-tone SIT sequences:

- Set an appropriate frequency detection range in the DX_CAP to detect each tone across all four SIT sequences. Set the first frequency detection range to detect the first tone for all four SIT sequences (approximately 900 to 1000 Hz). Set the second frequency detection range to detect the second tone for all four SIT sequences (approximately 1350 to 1450 Hz). Set the third frequency detection range to detect the third tone for all four SIT sequences (approximately 1725 to 1825 Hz).
- Set an appropriate detection time using the ca_timefrq and ca_mxtimefrq parameters to detect each tone across all four SIT sequences. For each tone, set ca_timefrq to 5 and ca_mxtimefrq to 50 to detect all SIT tones. The tones range in length from 27 to 38 (in 10 ms units), with some tones occasionally cut short by the central office.

NOTE: Occasionally, the first tone can also be truncated by a delay in the onset of Call Analysis due to the setting of ca_stdely.

- After an SIT sequence is detected, **ATDX_CPTERM( )** will return CR_CEPT to indicate an operator intercept, and you can determine which SIT sequence was detected by obtaining the actual detected frequency and duration for the tri-tone sequence using Extended Attribute functions.

The following fields in the DX_CAP are used for Frequency Detection on voice boards. Frequencies are specified in hertz, and time is specified in 10 ms units. To enable detection of the second and third tones, you must set the frequency detection range and time for each tone.

General

ca_stdely Start Delay: The delay after dialing has been completed and before starting Frequency Detection. This parameter also determines the start of Cadence Detection and Positive Voice Detection. Default: 25 (10 ms units). Note that this can affect detection of the first element of an operator intercept tone.

First Tone

ca_lowerfrq Lower Frequency: Lower bound for first tone in Hz. Default: 900.

ca_lowerfrq Lower Frequency: Lower bound for first tone in Hz. Default: 900.

ca_upperfrq Upper Frequency: Upper bound for first tone in Hz. Default: 1000. Adjust higher for additional operator intercept tones.

ca_timefrq Time Frequency: Minimum time for first tone to remain in bounds. The minimum amount of time required for the audio signal to remain within the frequency detection range for it to be detected. The audio signal must not be greater than **ca_upperfrq** or lower than **ca_lowerfrq** for at least the time interval specified in **ca_timefrq**. Default: 5 (10 ms units)

ca_mxtimefrq Maximum Time Frequency: Maximum allowable time for first tone to be present. Default: 0 (10 ms units).

Second Tone

NOTE: This tone is disabled initially and must be activated by the application using these variables.

ca_lower2frq Lower Bound for second Frequency: Lower bound for second tone in Hz. Default: 0

ca_upper2frq Upper Bound for second Frequency: Upper bound for second tone in Hz. Default: 0.

ca_time2frq Time for second Frequency: Minimum time for second tone to remain in bounds. Default: 0 (10 ms units).

ca_mxtime2frq Maximum Time for second Frequency: Maximum allowable time for second tone to be present. Default: 0 (10 ms units).

Third Tone

NOTE: This tone is disabled initially and must be activated by the application using these variables.

ca_lower3frq Lower Bound for third Frequency: Lower bound for third tone in Hz. Default: 0.

ca_upper3frq Upper Bound for third Frequency: Upper bound for third tone in Hz. Default: 0.

ca_time3frq Time for third Frequency: Minimum time for third tone to remain in bounds. Default: 0 (10 ms units).

ca_mxtime3frq Maximum Time for third Frequency: Maximum allowable time for third tone to be present. Default: 0 (10 ms units)

Tri-Tone Frequency Information Returned by Extended Attribute Functions

Upon detection of the specified sequence of frequencies, Extended Attribute functions can be used to provide the exact frequency and duration of each tone in the sequence. The frequency and duration information will allow exact determination of all four SIT sequences.

The following Extended Attribute function are used to provide information on the frequencies detected by Call Analysis.

Tri-Tone Frequency Information First Tone (ca_lowerfrq & ca_upperfrq)

ATDX_FRQHZ( ) Frequency Hertz: Frequency in Hz of the tone detected in the tone detection range specified by the DX_CAP **ca_lowerfrq** and **ca_upperfrq** parameters; usually the first tone of an SIT sequence. This function can be called on non-DSP boards.

ATDX_FRQDUR( ) Frequency Duration: Duration of the tone detected in the tone detection range specified by the DX_CAP **ca_lowerfrq** and **ca_upperfrq** parameters; usually the first tone of an SIT sequence (10 ms units).

Tri-Tone Frequency Information Second Tone (ca_lower2frq & ca_upper2frq)

ATDX_FRQHZ2( ) Frequency Hertz 2: Frequency in Hz of the tone detected in the tone detection range specified by the DX_CAP **ca_lower2frq** and **ca_upper2frq** parameters; usually the second tone of an SIT sequence.

ATDX_FRQDUR2( ) Frequency Duration 2: Duration of the tone detected in the tone detection range specified by the DX_CAP **ca_lowerfrq** and **ca_upperfrq** parameters; usually the second tone of an SIT sequence (10 ms units).

Tri-Tone Frequency Information Third Tone (ca_lower3frq & ca_upper3frq)

ATDX_FRQHZ3( ) Frequency Hertz 3: Frequency in Hz of the tone detected in the tone detection range specified by the DX_CAP **ca_lower3frq** and **ca_upper3frq** parameters; usually the third tone of an SIT sequence.

ATDX_FRQDUR3( ) Frequency Duration 3: Duration of the tone detected in the tone detection range specified by the DX_CAP **ca_lower3frq** and **ca_upper3frq** parameters; usually the third tone of an SIT sequence (10 ms units).

Global Tone Detection Tone Memory Usage

If you use Call Analysis to identify the tri-tone SIT sequences, Call Analysis will create tone detection templates internally, and this will reduce the number of tone templates that can be created using Global Tone Detection functions. See *Chapter 3. Global Tone Detection/Generation* for information relating to memory usage for Global Tone Detection.

Call Analysis will create one tone detection template for each single-frequency tone with a 100 Hz detection range. For example, if detecting the set of tri-tone SIT sequences (three frequencies) on each of four channels, the number of allowable user-defined tones will be reduced by three per channel.

If you initiate Call Analysis and there is not enough memory to create the SIT tone detection templates internally, you will get a CR_MEMERR error. This indicates that you are trying to exceed the maximum number of tone detection templates. The tone detection range should be limited to a maximum of 100 Hz per tone to reduce the chance of exceeding the available memory.

Frequency Detection Errors

The frequency detection range specified by the lower and upper bounds for each tone cannot overlap with each other, otherwise an error will be produced when the driver attempts to create the internal tone detection templates. For example, if **ca_upperfrq** is 1000 and **ca_lower2frq** is also 1000, an overlap occurs and will result in an error. Also the lower bound of each frequency detection range must be less than the upper bound (e.g., **ca_lower2frq** must be less than **ca_upper2frq**).

Setting Single Tone Frequency Detection Parameters

Setting single tone frequency detection parameters allows you to identify that a SIT sequence was encountered because 1 of the tri-tones in the SIT sequence was detected. But Frequency Detection cannot determine exactly which SIT sequence was encountered, because it is necessary to identify 2 tones in the SIT sequence to distinguish among the 4 possible SIT sequences.

The default frequency detection range is 900 to 1000 Hz, which is set to detect the first tone in any SIT sequence. Because the first tone is often truncated, you may want to increase **ca_upperfrq** to 1800 Hz so that it includes the third tone. If

this results in too many false detections, you can set Frequency Detection to detect only the third tone by setting **ca_lowerfrq** to 1750 and **ca_upperfrq** to 1800.

The following fields in the DX_CAP are used for Frequency Detection. Frequencies are specified in hertz, and time is specified in 10 ms units.

ca_stdely	Start Delay: The delay after dialing has been completed and before starting Frequency Detection. This parameter also determines the start of Cadence Detection. Default: 25 (10 ms units).
ca_infltr	Not used.
ca_lowerfrq	Lower Frequency: Lower bound for tone in Hz. Default: 900.
ca_upperfrq	Upper Frequency: Upper bound for tone in Hz. Default: 1000.
ca_timefrq	Time Frequency: Minimum time for tone to remain in bounds. The minimum amount of time required for the audio signal to remain within the frequency detection range for it to be detected. The audio signal must not be higher than **ca_upperfrq** or lower than **ca_lowerfrq** for at least the time interval specified in **ca_timefrq**, allowing for **ca_rejctfrq**. Default: 5 (10 ms units).
ca_rejctfrq	Not used.

Single Tone Frequency Information Returned

Upon detection of a frequency in the specified range, the Extended Attribute functions can be used to provide the exact frequency that was detected.

The following Extended Attribute functions return information on the single tone detected in the tone detection range specified by the DX_CAP **ca_lowerfrq** and **ca_upperfrq** fields.

ATDX_FRQOUT( )	• Not used.
ATDX_FRQHZ( )	• Frequency Hertz: Frequency in Hz of the tone detected in the tone detection range specified by the DX_CAP ca_lowerfrq and ca_upperfrq parameters; usually the first tone of an SIT sequence.

2.6.3. Cadence Detection in Basic Call Analysis

The Cadence Detection algorithm has been optimized for use in the United States standard network environment.

Cadence Detection as discussed here, and the Call Progress Characterization (CPC) Utility, are relevant to Basic Call Analysis only. For PerfectCall Call Analysis, refer to "Tone Detection in PerfectCall Call Analysis" below. The CPC program, which runs under MS-DOS, can be used to collect call progress data in both Basic Call Analysis and PerfectCall Call Analysis.

If your system is operating in another type of environment (such as behind a PBX), you can customize the Cadence Detection algorithm to suit your system through the adjustment of the Cadence Detection parameters. The Call Progress Characterization (CPC) Utility, which runs under MS-DOS can be used to determine the Cadence Detection parameter requirements for your system.

The following section discusses Cadence Detection and some of the most commonly adjusted Cadence Detection parameters. An entire listing of the DX_CAP, including all Cadence Detection parameters, can be found in the *Voice Programmer's Guide for Windows NT.*

Cadence Detection analyzes the audio signal on the line to detect a repeating pattern of sound and silence, such as the pattern produced by a ringback or a busy signal. These patterns are called *audio cadences*. Once a cadence has been established, it can be classified as a single ring, a double ring or a busy signal by comparing the periods of sound and silence to established parameters.

NOTES: 1. Sound is referred to as *nonsilence*.

2. The algorithm used for cadence detection is disclosed and protected under U.S. patent 4,477,698 of Melissa Electronic Labs, and other patents pending for Dialogic Corporation.

Typical Cadence Patterns

The following figures show some typical cadence patterns.

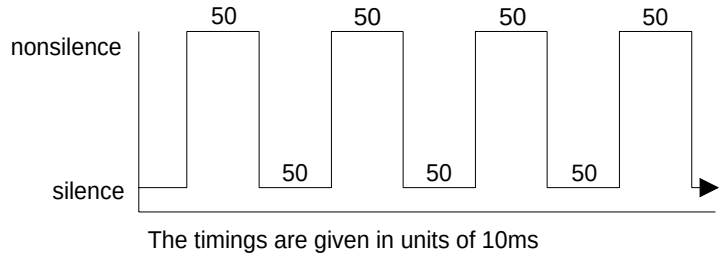

Figure 5. A Standard Busy Signal

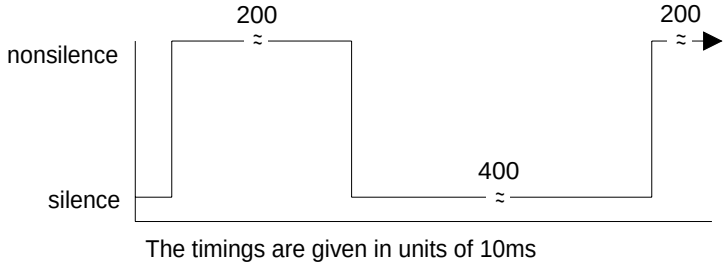

Figure 6. A Standard Single Ring

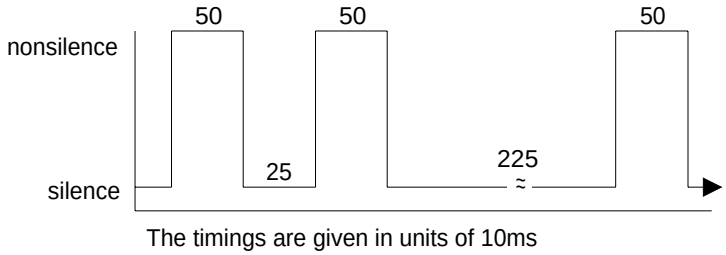

Figure 7. A Type of Double Ring

Elements of a Cadence

From the preceding cadence examples, you can see that a given cadence may contain 2 silence periods with different durations, such as for a double ring, but in general, the nonsilence periods have the same duration. To identify and distinguish between the different types of cadences, the Voice Driver must detect two silence and two nonsilence periods in the audio signal. Figure 8 illustrates cadence detection.

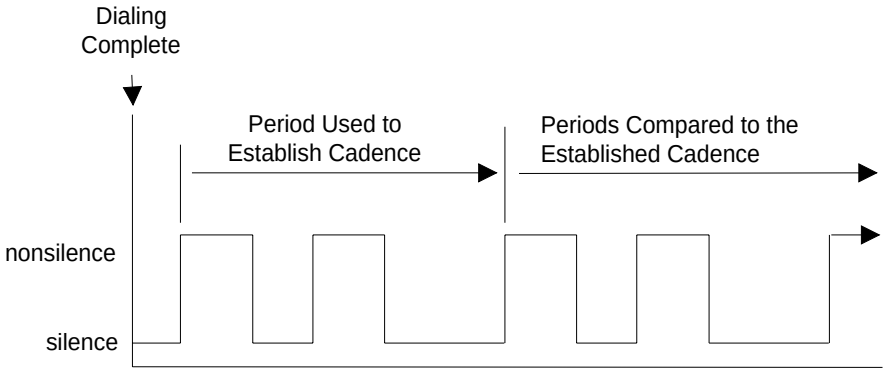

Figure 8. Cadence Detection

Once the cadence is established, the cadence values can be retrieved using the following Extended Attribute functions:

ATDX_SIZEHI( )	Length of the nonsilence period (in 10 ms units) for the detected cadence.
ATDX_SHORTLOW( )	Length of the shortest silence period for the detected cadence (in 10 ms units).
ATDX_LONGLOW( )	Length of the longest silence period for the detected cadence (in 10 ms units).

Only one nonsilence period is used to define the cadence because the nonsilence periods have the same duration. Figure 9 shows the elements of an established cadence.

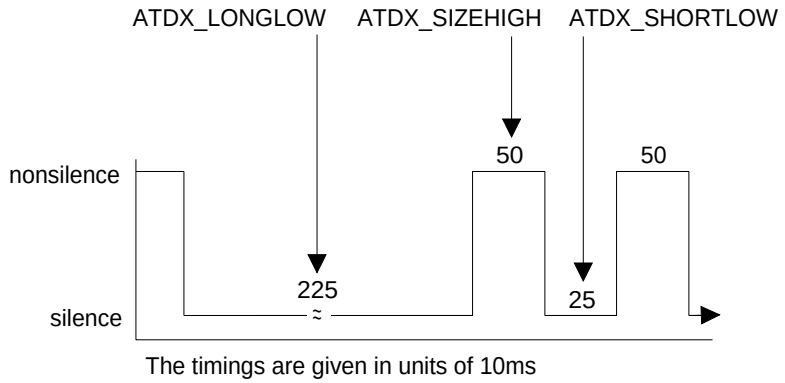

Figure 9. Elements of Established Cadence

The durations of subsequent states are compared with these fields to see if the cadence has been broken.

Outcomes of Cadence Detection

Cadence Detection can identify the following conditions during the period used to establish the cadence or after the cadence has been established:

- No Ringback
- Connect
- Busy
- No Answer

Although Loop Current Detection and Positive Voice Detection provide complementary means of detecting a connect, Cadence Detection provides the only way in Basic Call Analysis to detect a no ringback, busy, or no answer.

Cadence Detection can identify the following conditions during the period used to establish the cadence:

No Ringback While the cadence is being established, Cadence Detection determines whether the signal is continuous silence or nonsilence. In this case, Cadence Detection returns a *no ringback*, indicating there is a problem in completing the call.

Connect While the cadence is being established, Cadence Detection determines whether the audio signal departs from acceptable network standards for busy or ring signals. In this case, Cadence Detection returns a *connect*, indicating that there was a break from general cadence standards.

Cadence Detection can identify the following conditions after the cadence has been established:

Connect After the cadence has been established, Cadence Detection determines whether the audio signal departs from the established cadence. In this case, Cadence Detection returns a *connect*, indicating that there was a break in the established cadence.

No Answer After the cadence has been established, Cadence Detection determines whether the cadence belongs to a single or double ring. In this case, Cadence Detection can return a *no answer*, indicating there was no break in the ring cadence for a specified number of times.

Busy After the cadence has been established, Cadence Detection determines whether the cadence belongs to a slow busy signal. In this case, Cadence Detection can return a *busy*, indicating that the busy cadence was repeated for a specified number of times.

To determine whether the ring cadence is a double or single ring, compare the value returned by the **ATDX_SHORTLOW( )** function to the DX_CAP field **ca_lo2rmin**. If the **ATDX_SHORTLOW( )** value is less than **ca_lo2rmin**, the cadence is a double ring; otherwise it is a single ring.

Setting Selected Cadence Detection Parameters

Only the most commonly adjusted Cadence Detection parameters are discussed here. A listing of the DX_CAP function can be found in *Chapter 4* in the *Voice Programmer's Guide for Windows NT.*

You should only need to adjust Cadence Detection parameters for network environments that do not conform to the U.S. standard network environment (such as behind a PBX).

The Call Progress Characterization Utility (CPC) Utility , which runs under MS-DOS can be used to determine the correct cadence detection parameter settings for your system.

Call Progress Charterization General Cadence Detection Parameters

ca_stdely Start Delay: The delay after dialing has been completed and before starting Cadence Detection. This parameter also determines the start of Frequency Detection and Positive Voice Detection. Default: 25 (10 ms units) = 0.25 seconds. Be careful with this variable. Setting this variable too small may allow switching transients or, if too long, miss critical signaling.

To eliminate audio signal glitches over the telephone line, the parameters **ca_logltch** and **ca_higltch** are used to determine the minimum acceptable length of a valid silence or nonsilence duration. Any silence interval shorter than **ca_logltch** is ignored, and any nonsilence interval shorter than **ca_higltch** is ignored.

ca_higltch High Glitch: The maximum nonsilence period to ignore. Used to help eliminate spurious nonsilence intervals. Default: 19 (in 10 ms units)

ca_logltch Low Glitch: The maximum silence period to ignore. Used to help eliminate spurious silence intervals. Default: 15 (in 10 ms units)..

Cadence Detection Parameters Affecting a No Ringback

After Cadence Detection begins, it waits for an audio signal of nonsilence. The maximum waiting time is determined by the parameter **ca_cnosig** (continuous no signal). If the length of this period of silence exceeds the value of **ca_cnosig**, a *no ringback* is returned (see Figure 10). This usually indicates a dead or disconnected telephone line or some other system malfunction.

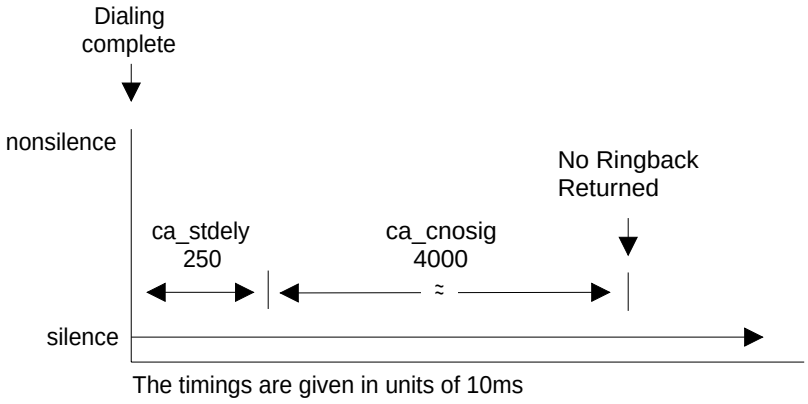

Figure 10. No Ringback Due to Continuous No Signal

ca_cnosig Continuous No Signal: The maximum time of silence (no signal) allowed immediately after Cadence Detection begins. If exceeded, a *no ringback* is returned. Default: 4000 (in 10 ms units), or 40 seconds.

If the length of any period of nonsilence exceeds the value of **ca_cnosil** (continuous nonsilence), a *no ringback* is returned (see Figure 11).

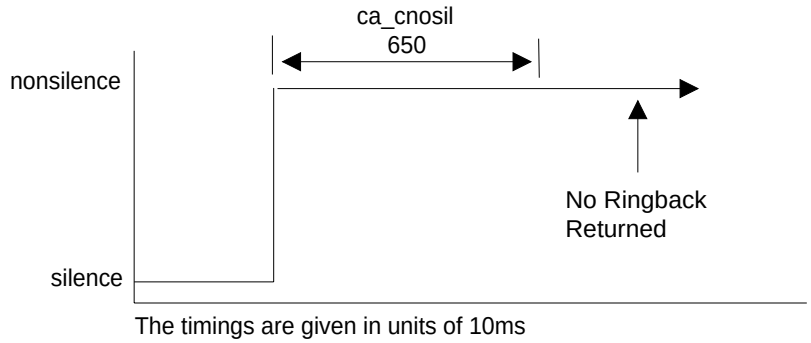

Figure 11. No Ringback Due to Continuous Nonsilence

ca_cnosil Continuous Nonsilence: The maximum length of nonsilence allowed. If exceeded, a *no ringback* is returned. Default: 650 (in 10 ms units), or 6.5 seconds.

Cadence Detection Parameters Affecting a No Answer or Busy

By using the **ca_nbrdna** parameter, you can set the maximum number of ring cadence repetitions that will be detected before returning a *no answer*.

By using the **ca_nbrdna** and **ca_nsbusy** parameters, you can set the maximum number of busy cadence repetitions.

ca_nbrdna Number of Rings Before Detecting No Answer: The number of single or double rings to wait before returning a *no answer*. Default: 4.

ca_nsbusy Nonsilence Busy: The number of nonsilence periods in addition to **ca_nbrdna** to wait before returning a *busy*. Default: 0. **ca_nsbusy** is *added* to **ca_nbrdna** to give the actual number of busy cadences at which to return busy. Note that even though ca_nsbusy is declared as an unsigned variable, it can be a small negative number. Do not allow **ca_nbrdna** + **ca_nsbusy** to equal 2. This is a foible of the 2's complement bit mapping of a small negative number to an unsigned variable.

Cadence Detection Parameters Affecting a Connect

Cadence Detection parameters are used to measure the length of the salutation when the phone is answered. The salutation is the greeting when a person answers the phone, or an announcement when an answering machine or computer answers the phone.

By examining the length of the greeting or salutation you receive when the phone is answered, you may be able to distinguish between an answer at home, at a business, or by an answering machine.

The length of the salutation is returned by the **ATDX_ANSRSIZ( )** function. By examining the value returned, you can estimate the kind of answer that was received.

Normally, a person at home will answer the phone with a brief salutation that lasts about 1 second, such as "Hello" or "Smith Residence." A business will usually answer the phone with a longer greeting that lasts from 1.5 to 3 seconds, such as "Good afternoon, Dialogic Corporation." An answering machine or computer will usually play an extended message that lasts more than 3 or 4 seconds.

This method is not 100% accurate, for the following reasons:

- The length of the salutation can vary greatly.
- A pause in the middle of the salutation can cause a premature connect event.
- If the phone is picked up in the middle of a ringback, the ringback tone may be considered part of the salutation, making the **ATDX_ANSRSIZ( )** return value inaccurate.

In the last case, if someone answers the phone in the middle of a ring and quickly says “Hello”, the nonsilence of the ring will be indistinguishable from the nonsilence of voice that immediately follows, and the resulting **ATDX_ANSRSIZ( )** return value may include both the partial ring and the voice. In this case, the return value may deviate from the actual salutation by 0 to +1.8 seconds. The salutation would appear to be the same as when someone answers the phone after a full ring and says two words.

NOTE: A return value of 180 to 480 may deviate from the actual length of the salutation by 0 to +1.8 seconds.

Cadence Detection will measure the length of the salutation when the **ca_hedge** (hello edge) parameter is set to 2 (the default).

ca_hedge Hello Edge: The point at which a *connect* will be returned to the application, either the rising edge (immediately when a connect is detected) or the falling edge (after the end of the salutation). 1 = rising edge. 2 = falling edge. Default: 2 (*connect* returned on falling edge of salutation). Try changing this if the called party has to say "Hello" twice to trigger the answer event.

Because a greeting might consist of several words, Call Analysis waits for an specified period of silence before assuming the salutation is finished. The **ca_ansrdgl** (answer deglitcher) parameter determines when the end of the salutation occurs. This parameter specifies the maximum amount of silence allowed in a salutation before it is determined to be the end of the salutation. To use **ca_ansrdgl**, set it to approximately 50 (in 10 ms units).

ca_ansrdgl Answer Deglitcher: The maximum silence period (in 10 ms units) allowed between words in a salutation. This parameter should be enabled only when you are interested in measuring the length of the salutation. Default: -1 (disabled).

The **ca_maxansr** (maximum answer) parameter determines the maximum allowable answer size before returning a *connect*.

ca_maxansr Maximum Answer: The maximum allowable length of **ansrsize**. When **ansrsize** exceeds **ca_maxansr**, a *connect* is returned to the application. Default: 1000 (in 10 ms units), or 10 seconds.

Figure 12 shows how the **ca_ansrdgl** parameter works.

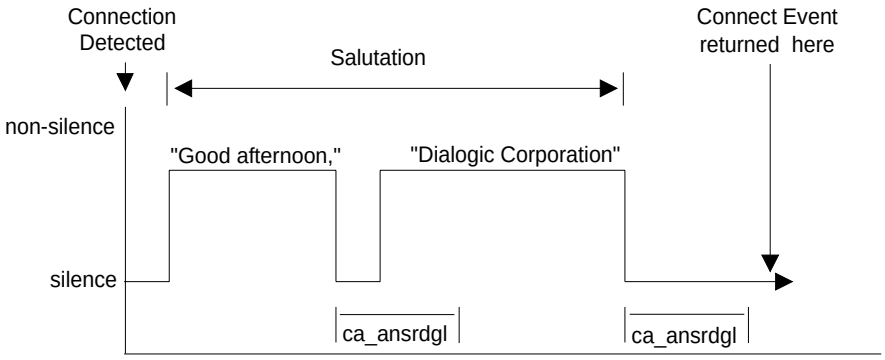

Figure 12. Cadence Detection Salutation Processing

When **ca_hedge** = 2, Cadence Detection waits for the end of the salutation before returning a *connect*. The end of the salutation occurs when the salutation contains a period of silence that exceeds **ca_ansrdgl** or the total length of the salutation exceeds **ca_maxansr**. When the *connect* event is returned, the length of the salutation can be retrieved using the **ATDX_ANSRSIZ( )** function.

After Call Analysis is complete, check call **ATDX_ANSRSIZ( )**. If the return value is less than 180 (1.8 seconds), you have probably contacted a residence. A return value of 180 to 300 is probably a business. If the return value is larger than 480, you have probably contacted an answering machine. A return value of 0 means that a *connect* was returned because excessive silence was detected. This can vary greatly in practice.

NOTE: When a connect is detected through Positive Voice Detection or Loop Current Detection, the DX_CAP parameters ca_hedge, ca_ansrdgl, and ca_maxansr are ignored.

Cadence Information Returned Using Extended Attribute Functions

ATDX_SIZEHI( )	• Size High: Duration of the cadence non-silence period (in 10 ms units).
ATDX_SHORTLOW( )	• Short Low: Duration of the cadence shorter silence period (in 10 ms units).
ATDX_LONGLOW( )	• Long Low: Duration of the cadence longer silence period (in 10 ms units).
ATDX_ANSRSIZ( )	• Answer Size: Duration of answer if a connect occurred (in 10 ms units) (ignored for what for PVD).
ATDX_CONNTYPE( )	• Connection type. If ATDX_CONNTYPE( ) returns CON_CAD, the connect was due to Cadence Detection.

2.6.4. Tone Detection in PerfectCall Call Analysis

PerfectCall Call Analysis uses a combination of Cadence Detection and Frequency Detection to identify certain signals during the course of an outgoing call. Cadence detection identifies repeating patterns of sound and silence, and Frequency Detection determines the pitch of the signal. Together, the cadence and frequency of a signal make up its *tone definition*.

Unlike Basic Call Analysis, which uses fields in the DX_CAP structure to store signal cadence information, PerfectCall Call Analysis uses tone definitions which are contained in the Voice Driver itself. Functions are available to modify these default tone definitions.

Types of Tones

Tone definitions are used to identify several kinds of signals. The following list shows the defined tones and their tone identifiers. Tone identifiers are returned by the **ATDX_CRTNID( )** function.

- TID_DIAL_LCL Local dial tone

- TID_DIAL_INTL International dial tone
- TID_DIAL_XTRA Special (or "extra") dial tone
- TID_BUSY1 Single tone busy signal
- TID_BUSY2 Dual tone busy signal
- TID_RNGBK1 Ringback tone
- TID_FAX1 A fax CNG tone
- TID_FAX2 A fax CED or modem tone

The tone identifiers are used in calls to the functions **dx_chgfreq( )**, **dx_chgdur( )**, and **dx_chgrepcnt( )** to change the tone definitions. Refer to these functions in the Function Reference chapter in the *Voice Programmer's Guide for Windows NT* and the "How to Enable PerfectCall Call Analysis" paragraph of this section for details.

Dial Tone Detection

Wherever PerfectCall Call Analysis is in effect, a dial string for an outgoing call may specify special ASCII characters that instruct the system to wait for a certain kind of dial tone. The following additional special characters may appear in a dial string:

L Wait for a local dial tone
I Wait for an international dial tone
X Wait for a special (extra) dial tone

The tone definitions for each of these dial tones is set for each channel at the time of the **dx_initcallp( )** function. In addition, the following DX_CAP fields identify how long to wait for a dial tone, and how long the dial tone must remain stable.

ca_dtn_pres Dial Tone Present: The length of time that the dial tone must be continuously present (in 10 ms units). If a dial tone is present for this amount of time, dialing of the dial string proceeds. Default value: 100 (one second).

ca_dtn_npres Dial Tone Not Present: The length of time to wait before declaring the dial tone not present (in 10 ms units). If a dial tone of sufficient length (**ca_dtn_pres**) is not found within this period of time, Call Analysis terminates with the reason CR_NODIALTONE. The dial tone character

(**L**, **I**, or **X**) for the missing dial tone can be obtained using **ATDX_DTNFAIL( )**. Default value: 300 (three seconds).

ca_dtn_deboff Dial Tone Debounce: The maximum duration of a break in an otherwise continuous dial tone before it is considered invalid (in 10 ms units). This parameter is used for ignoring short drops in dial tone. If a drop longer than **ca_dtn_deboff** occurs, then dial tone is no longer considered present, and another dial tone must begin and be continuous for **ca_dtn_pres**. Default value: 10 (100 msec).

Ringback Detection

PerfectCall Call Analysis uses the tone definition for ringback to identify the first ringback signal of an outgoing call. At the end of the first ringback (that is, normally, at the beginning of the second ringback), a timer goes into effect. The system continues to identify ringback signals (but does not count them). If a break occurs in the ringback cadence, the call is assumed to have been answered, and Call Analysis terminates with the reason CR_CONN (connect); the connection type returned by the **ATDX_CONNTYPE( )** function will be CON_CAD (cadence break).

However, if the timer expires before a connect is detected, then the call is deemed unanswered, and Call Analysis terminates with the reason CR_NOAN.

The following DX_CAP fields govern this behavior:

ca_stdely Start Delay: The delay after dialing has been completed before starting Cadence Detection, Frequency Detection, and Positive Voice Detection (in 10 ms units). Default: 25 (0.25 seconds).

ca_cnosig Continuous No Signal: The maximum length of silence (no signal) allowed immediately after the **ca_stdely** period (in 10 ms units). If this duration is exceeded, Call Analysis is terminated with the reason CR_NORNG (no ringback

detected). Default value: 4000 (40 seconds).

ca_noanswer No Answer: The length of time to wait after the first ringback before deciding that the call is not answered (in 10 ms units). If this duration is exceeded, Call Analysis is terminated with the reason CR_NOAN (no answer). Default value: 3000 (30 seconds).

ca_maxintering Maximum Inter-ring: The maximum length of time to wait between consecutive ringback signals (in 10 ms units). If this duration is exceeded, Call Analysis is terminated with the reason CR_CONN (connected). Default value: 800 (8 seconds).

Busy Tone Detection

There are two busy tones defined in PerfectCall Call Analysis, TID_BUSY1 and TID_BUSY2. If either of them is detected while Frequency Detection and Cadence Detection are active, then Call Analysis is terminated with the reason CR_BUSY. **ATDX_CRTNID( )** identifies which busy tone was detected.

No DX_CAP fields affect busy tone detection.

Positive Answering Machine Detection

Positive Answering Machine Detection (PAMD) is available only with PerfectCall Call Analysis. Whenever PAMD is enabled, Positive Voice Detection (PVD) is also enabled. PAMD is enabled by setting the **ca_intflg** field of the DX_CAP structure to one of the following values:

- DX_PAMDENABLE: Enable PAMD and PVD without enabling SIT Frequency Detection.
- DX_PAMDOPTEN: Enable PAMD and PVD, and enable SIT frequency detection.

When enabled, detection of an answering machine will result in the termination of Call Analysis with the reason CR_CONN (connected);the connection type returned by the **ATDX_CONNTYPE( )** function will be CON_PAMD.

To distinguish between a greeting by a live human and one by an answering machine, one of two methods is used:

- PAMD_QUICK: The quick method examines only the events surrounding the connect time and makes a rapid judgment as to whether or not an answering machine is involved.
- PAMD_FULL: The long method looks at the full greeting to determine whether it came from a human or a machine.

The slower method gives a very accurate determination. However in situations where a fast decision is more important than accuracy, PAMD_QUICK might be preferred.

The following DX_CAP fields govern positive answering machine detection:

ca_pamd_spdval PAMD Speed Value: Whether to make a quick decision on Positive Answering Machine Detection. Possible values:

- PAMD_FULL: Look at greeting (long method)
- PAMD_QUICK: Look at connect only (quick method)

Default value: PAMD_FULL.

ca_pamd_qtemp PAMD Qualification Template: The algorithm to use in PAMD. At present there is only one template: PAMD_QUAL1TMP. This parameter must be set to this value.

ca_pamd_failtime PAMD fail time: Maximum time to wait for Positive Answering Machine Detection or Positive Voice Detection after a cadence break. Default Value: 400 (in 10 ms units).

ca_pamd_minring Minimum PAMD ring: Minimum allowable ring duration for Positive Answering Machine Detection.

Default Value: 190 (in 10 ms units).

Fax or Modem Tone Detection

Detection of fax and modem tones is available only with PerfectCall Call Analysis. Two tones are defined, TID_FAX1 and TID_FAX2. If either of them is detected while Frequency Detection and Cadence Detection are active, then Call Analysis is terminated with the reason CR_FAXTONE. **ATDX_CRTNID( )** identifies which fax or modem tone was detected.

No DX_CAP fields affect fax or modem tone detection.

2.6.5. Loop Current Detection

Some telephone systems return a momentary drop in loop current when a connection has been established (*answer supervision*). Loop Current Detection returns a *connect* when a transient loop current drop is detected.

In some environments, including most PBXs, answer supervision is not provided. In these environments, Loop Current Detection will not function. Check with your Central Office or PBX supplier to see if answer supervision based on loop current changes is available.

In some cases, the application may receive one or more transient loop current drops before an actual connection occurs. This is particularly true when dialing long-distance numbers, when the call may be routed through several different switches. Any one of these switches may be capable of generating a momentary drop in loop current.

To disable Loop Current Detection, set **ca_lcdly** to -1.

Loop Current Detection Parameters Affecting a Connect

To prevent detecting a connect prematurely or falsely, due to a spurious loop current drop, you can delay the start of Loop Current Detection by using the parameter **ca_lcdly**.

Loop Current Detection returns a *connect* after detecting a loop current drop. To allow the person who answered the phone to say "hello" before the application proceeds, you can delay the return of the *connect* by using the parameter **ca_lcdly1**.

ca_lcdly	Loop Current Delay: The delay after dialing has been completed and before beginning Loop Current Detection. To disable Loop Current Detection, set to -1. Default: 400 (10 ms units).
ca_lcdly1	Loop Current Delay 1: The delay after Loop Current Detection detects a transient drop in loop current and before Call Analysis returns a *connect* to the application. Default: 10 (10 ms units).

If the **ATDX_CONNTYPE( )** function returns CON_LPC, the connect was due to Loop Current Detection.

NOTE: When a connect is detected through Positive Voice Detection or Loop Current Detection, the DX_CAP parameters ca_hedge, ca_ansrdgl, and ca_maxansr are ignored.

2.6.6. Positive Voice Detection

Positive Voice Detection (PVD) can detect when a call has been answered by determining whether an audio signal is present that has the characteristics of a live or recorded human voice. This provides a very precise method for identifying when a connect occurs.

PVD is especially useful in those situations where answer supervision is not available for Loop Current Detection to identify a connect, and where the cadence is not clearly broken for Cadence Detection to identify a connect (for example, when the nonsilence of the cadence is immediately followed by the nonsilence of speech).

If the **ATDX_CONNTYPE( )** function returns CON_PVD, the connect was due to Positive Voice Detection.

NOTE: When a connect is detected through Positive Voice Detection or Loop Current Detection, the DX_CAP parameters ca_hedge, ca_ansrdgl, and ca_maxansr are ignored.

2.7. Call Analysis Errors

If **ATDX_CPTERM( )** returns CR_ERROR, you can use **ATDX_CPERROR( )** to determine the Call Analysis error that occurred.

CR_MEMERR	Out of memory trying to create temporary SIT tone templates (exceeds maximum number of templates).
CR_TMOUTON	Time-out waiting for an SIT tone.
CR_TMOUTOFF	SIT tone too long (exceeds a **ca_mxtimefrq** parameter).
CR_UNEXPTN	Unexpected SIT tone (the sequence of detected tones did not correspond to the SIT sequence).
CR_MXFRQERR	Invalid ca_mxtimefrq field in DX_CAP. If the ca_mxtimefrq parameter for each SIT tone is non-zero, it must have a value greater than or equal to the ca_timefrq parameter for the same SIT tone.
CR_UPFRQERR	Invalid upper frequency selection. This value must be non-zero for detection of any SIT tone.
CR_LGTUERR	Lower frequency greater than upper frequency.
CR_OVRLPERR	Overlap in selected SIT tones.

3. Global Tone Detection/Generation

3.1. Global Tone Detection/Generation Overview

This chapter provides a description of Global Tone Detection (GTD) and Global Tone Generation (GTG). These features provide the ability to generate and detect single or dual frequency tones.

The functions associated with Global Tone Detection and Global Tone Generation are categorized in *Chapter 2* of the *Voice Programmer's Guide for Windows NT.*

NOTE: See *Chapter 3* for Call Analysis functions that work with Global Tone Detection.

Global Tone Detection and Global Tone Generation is available on all voice boards.

3.2. Global Tone Detection (GTD)

Global Tone Detection is a feature that allows a user to define the characteristics of a tone in order to detect a tone with the same characteristics. The characteristics of a tone are defined using GTD tone templates. The tone templates contain parameters that allow the user to assign frequency bounds and cadence components. Single and dual frequency tones are detected by comparing all incoming sounds to the GTD tone templates.

GTD operates on a channel-by-channel basis and is active when the channel is off-hook, unless the system uses a DTI/xxx, in which case, GTD is always active. GTD works simultaneously with DTMF and MF tone detection.

The driver responds to a detected tone by producing either a *tone event* on the event queue or a *digit* on the digit queue. The particular response depends upon the GTD tone configuration.

Use the Global Tone Detection functions to access tone templates and enable detection of single and dual frequency tones that fall outside those automatically

provided with the Voice Driver. This includes tones outside the standard DTMF set of 0-9, a-d, *, and #, and the standard MF tones 0-9, *, and a-c.

3.2.1. Defining GTD Tones

GTD tones can have an associated ASCII digit (and digit type) specified using the **digit** and **digtype** parameters in the **dx_addtone( )** function. When the tone is detected, the digit is placed in the DV_DIGIT buffer and can be retrieved using the **dx_getdig( )** function. When the tone is detected, either the tone event or the digit associated with the tone can be used as a termination condition to terminate I/O functions.

Termination conditions are set using the DV_TPT data structure. See *Appendix A* in the *Voice Programmer's Guide for Windows NT* for information about the DV_TPT data structure.

NOTE: If you want to terminate on multiple tones (or digits), you need to specify the terminating conditions for each tone in a separate DV_TPT data structure.

3.2.2. Building Tone Templates

When creating the tone template you can define the following:

- single or dual frequency (300-3500 Hz)
- optional ASCII digit associated with the tone template
- cadence components

Adding a tone template to a channel enables detection of a tone on that channel. Although only one tone template can be created at a time, multiple tone templates can be added to a channel. Each channel can have a different set of tone templates. Once created, tone templates can be selectively enabled or disabled.

NOTE: A particular tone template cannot be changed or deleted. A tone template can be disabled on a channel, but to delete a tone template, all tone templates on that channel must be deleted.

The following functions are used to define tone templates:

- **dx_bldst( )** — build a single frequency tone description
- **dx_blddt( )** — build a dual frequency tone description
- **dx_bldstcad( )** — build a single frequency tone cadence description
- **dx_blddtcad( )** — build a dual frequency tone cadence description
- **dx_setgtdamp( )** — set the GTD amplitude

NOTES:

1. GTD build functions define new tone templates, and **dx_addtone( )** adds the tone templates to a channel.
2. Use **dx_addtone( )** to enable detection of the tone template on a channel.
3. After building a tone template using a **dx_bld...( )** function, **dx_addtone( )** must be called to add this tone template to a channel. If the template is not added, the next call to a **dx_bld...( )** function will overwrite the tone definition contained in the previous template.

dx_bldst( ) defines a simple single frequency tone. Subsequent calls to **dx_addtone( )** will use this tone until another tone is defined. Thus, you can build a tone template and add it to several different channels.

dx_blddt( ) defines a simple dual frequency tone. Subsequent calls to **dx_addtone( )** will use this tone until another tone is defined. Thus, you can build a tone template and add it to several different channels.

NOTE: The D/41ESC, D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1, D/320SC, DTI/240SC, DTI/241SC, DTI/300SC, DTI/301SC, LSI/81SC, and LSI/161SC boards cannot detect dual tones with frequency components closer than 65 Hz. Use a single tone description to detect dual tones that are closer together than the ranges specified above.

dx_bldstcad( ) defines a simple single frequency cadence tone. Subsequent calls to **dx_addtone( )** will use this tone until another tone is defined. Thus, you can

build a tone template and add it to several different channels. A single frequency cadence tone has single frequency signals with specific on/off characteristics.

dx_blddtcad( ) defines a simple dual frequency cadence tone. Subsequent calls to **dx_addtone( )** will use this tone until another tone is defined. Thus, you can build a tone template and add it to several different channels. A dual frequency cadence tone has dual frequency signals with specific on/off characteristics.

The minimum on- and off-time for cadence detection is 40 ms on and 40 ms off.

dx_setgtdamp( ) sets the amplitudes used by GTD. The amplitudes set using **dx_setgtdamp( )** will be the default amplitudes that will apply to all tones built using the **dx_bld...( )** functions. The amplitudes will remain valid for all tones built until **dx_setgtdamp( )** is called again and the amplitudes are changed. The following table lists some standard Bell System Network Call Progress Tones. The frequencies are useful when creating the tone templates.

Tone	Frequency(Hz)	On Time (ms)	Off Time (ms)
Dial	350 + 440	Continuous	
Busy	480 + 620	500	500
Congestion (Toll)	480 + 620	200	300
Reorder (Local)	480 + 620	300	200
Ringback	440 + 480	2000	4000

3.2.3. Working with Tone Templates

Use the following functions to add/delete tone templates or to enable/disable tone detection:

- **dx_addtone( )** add a tone template
- **dx_deltones( )** delete tone templates
- **dx_distone( )** disable detection of a tone
- **dx_enbtone( )** enable detection of a tone

dx_addtone( ) adds the tone template that was defined by the most recent GTD build-tone function call to the specified channel. Adding a tone template to a channel downloads it to the board and enables detection of tone-on and tone-off events for that tone template.

Cautions

1. Each tone template must have a unique identification.
2. Errors will occur if you use **dx_addtone( )** to change a tone template that has previously been added.

dx_deltones( ) removes all tone templates previously added to a channel with **dx_addtone( )**. If no tone templates were previously enabled for this channel, the function has no effect.

NOTE: **dx_deltones( )** does not affect tones defined by build-tone template functions and tone templates not yet defined. If you have added tones for PerfectCall Call Analysis, these tones are also deleted.

dx_distone( ) disables the detection of DX_TONEON and/or DX_TONEOFF events on a channel. Detection capability for user-defined tones is enabled on a channel by default when **dx_addtone( )** is called.

dx_enbtone( ) enables the detection of DX_TONEON and/or DX_TONEOFF events on a channel. Detection capability for tones is enabled on a channel by default when **dx_addtone( )** is called. The function can re-enable tones disabled by **dx_distone( )**.

DX_TONEON and DX_TONEOFF events are Call Status Transition (CST) events.

3.2.4. Tone Event Retrieval

To retrieve events, perform the following:

1. Call **dx_addtone( )** or **dx_enbtone( )** to enable tone-on/off detection.
2. Call **dx_getevt( )** to wait for CST event(s). Events are returned in the DX_EBLK structure.

NOTE: These procedures are the same as the retrieval of any other CST event, except that **dx_addtone( )** or **dx_enbtone( )** are used to enable event detection instead of **dx_setevtmsk( )**.

You can optionally specify an associated ASCII digit (and digit type) with the tone template. In this case, the tone template is treated like DTMF tones. When the digit is detected, it is placed in the digit buffer and can be used for termination. When an associated ASCII digit is specified, tone events will not be generated for that tone.

3.2.5. Maximum Number of Tone Templates

Guidelines for maximum number of tone templates are listed below and in Table 2:

- The D/21D or D/21E board allows a maximum of 33 tone templates per board, which results in 16 tone templates per channel if apportioned evenly among the 2 channels.
- The D/4xD, D/41E, or D/41ESC board allows a maximum of 33 tone templates per board, which results in 8 tone templates per channel if apportioned evenly among the 4 channels.
- The D/81A board allows a maximum of 100 tone templates per board, which results in 12 tone templates per channel if apportioned evenly among the 8 channels.
- The D/12x board allows a maximum of 166 tone templates per board, which results in 13 tone templates per channel if apportioned evenly among the 12 channels.
- The D/160SC-LS board has a maximum of 240 tone templates per board or 15 per channel if evenly apportioned.

- The D/240SC and D/240SC-T1 boards have a maximum of 300 tone templates per board or 15 per channel if evenly apportioned.
- The D/300SC-E1 and D/320SC boards have a maximum of 450 tone templates per board or 15 per channel if evenly apportioned.
- The DTI/240SC and DTI/241SC boards have a maximum of 300 tone templates per board or 15 per channel if evenly apportioned.
- The DTI/300SC and DTI/301SC boards have a maximum of 450 tone templates per board or 15 per channel if evenly apportioned.
- The LSI/81SC and LSI/161SC boards have a maximum of 240 tone templates per board or 15 per channel if evenly apportioned.
- The maximum number of tone templates is based on tone templates that define a *dual* tone with a frequency detection range (bandwidth) of 62 Hz. (The detection range is the difference between the minimum and maximum defined frequencies for the tone.) For dual tones and tones that exceed 100 Hz bandwidth, see Specific Guidelines in this section.
- If you use Call Analysis to identify the tri-tone Special Information Tone (SIT) sequences, Call Analysis will create tone templates internally, and this will reduce the number of tone templates that can be created by the user. Call Analysis creates one tone template for each single-frequency tone that is defined using DX_CAP. For example, if detecting the SIT tri-tone sequences per channel, the number of tone templates that can be defined by the user will be reduced by 3 per channel.

See *Chapter* 2. Call Analysis*2. Call Analysis* for more information about Call Analysis.

- If you create R2MF user-defined tones using **r2_creatfsig( )**, the voice boards will usually be able to create all 15 R2MF user-defined tones due to the overlap in frequencies for the R2MF signals (see next heading on Specific Guidelines). If these boards do not have sufficient memory, they may be able to support R2MF signaling through a reduced number of R2MF user-defined tones.
- If you use **dx_blddt( )** (or one of the **dx_bld...( )** build tone functions) or **r2_creatfsig( )** to define a user-defined tone that alone or with existing user-defined tones exceeds the available memory, you will get an EDX_MAXTMPLT error.

- If you initiate Call Analysis and there is not enough memory to create the SIT tones internally, you will get a CR_MEMERR.
- The tone detection range should be limited to a maximum of 200 Hz per tone to reduce the chance of exceeding the available memory.

NOTE: The **dx_deltones( )** function deletes all tone templates from a specified channel and releases the memory that was used for those tone templates. When an associated ASCII digit is specified, tone events will not be generated for that tone.

Table 2. Maximum Memory and Tone Templates (for Dual Tones)

Hardware	Tone Templates Per Board	Tone Templates Per Channel
D/21D	33	16
D/4xD	33	8
D/21E	33	16
D/41E	33	8
D/41ESC	33	8
D/81A	100	12
D/121A	166	13
D/121B	166	13
D/160SC-LS	240	15
D/240SC	300	15
D/240SC-T1	300	15
D/300SC-E1	450	15
D/320SC	450	15

Hardware	Tone Templates Per Board	Tone Templates Per Channel
DTI/240SC	300	15
DTI/241SC	300	15
DTI/300SC	450	15
DTI/301SC	450	15
LSI/81SC	240	15
LSI/161SC	240	15

3.2.6. Applications

Two sample applications for Global Tone Detection are described in this section. The first application describes how to use GTD to detect a fast-busy signal to determine when a disconnect occurs. The second application describes how to use GTD for leading edge detection of a tone using debounce time.

Disconnect Supervision

Global Tone Detection can be used for disconnect supervision. When a telephone call terminates, the central office may send a momentary drop in loop current to signal the disconnect. In configurations where the voice board is connected to a Private Branch Exchange (PBX), it is likely that there will be no drop in loop current for the voice board to detect. Instead, the PBX may initiate a fast-busy signal to indicate the disconnect. Global Tone Detection can be used to detect this fast-busy signal. Perform the following to detect the signal:

1. Determine the frequencies of the signal.
2. Characterize the on/off durations and tolerances of the signal cadence.
3. Use a build-tone function to define the characteristics of a single or dual tone with cadence in a tone template.

4. Use the **dx_addtone( )** function to add the GTD tone template for Global Tone Detection on each channel.

Leading Edge Detection Using Debounce Time

Rather than detecting a signal immediately, an application may want to wait for a period of time (debounce time) before the DX_TONEON event is generated indicating the detection of the signal. The **dx_bldstcad( )** and **dx_blddtcad( )** functions can detect leading edge debounce on-time. A tone must be present at a given frequency and for a period of time (debounce time) before a DX_TONEON event is generated. The debounce time is specified using the ontime and ondev parameters in the **dx_bldstcad( )** or **dx_blddtcad( )** functions.

To use this application specify the following values for the **dx_bldstcad( )** or **dx_blddtcad( )** function parameters listed below:

ontime	1/2 of the desired debounce time
ondev	-1/2 of the desired debounce time
offtime	0
offdev	0
repcnt	0

NOTE: This application cannot work with the functions **dx_blddt( )** and **dx_bldst( )** since these functions do not have timing field parameters.

3.3. Global Tone Generation (GTG)

Global Tone Generation enables the creation of user-defined tones. The Tone Generation template, TN_GEN, is used to define the tones with the following information:

- Single or dual tone
- Frequency fields
- Amplitude for each frequency
- Duration of tone

3.3.1. Global Tone Generation Functions

The following functions are used to generate tones:

- **dx_bldtngen( )** build a tone generation template
- **dx_playtone( )** play a tone

dx_bldtngen( ) is a convenience function that sets up the tone generation template data structure (TN_GEN) by allowing the assignment of specified values to the appropriate fields. The tone generation template is placed in the user's return buffer and can then be used by the **dx_playtone( )** function to generate the tone.

dx_playtone( ) plays a tone specified by the tone generation template (pointed to by **tngenp**). Termination conditions are set using the DV_TPT structure. The reason for termination is returned by the **ATDX_TERMMSK( )** function. **dx_playtone( )** returns a 0 to indicate that it has completed successfully.

3.3.2. Building and Implementing a Tone Generation Template

The tone generation template defines the frequency, amplitude, and duration of a single or dual frequency tone to be played. You can use the convenience function **dx_bldtngen( )** to set up the structure. Use **dx_playtone( )** to play the tone.

The TN_GEN data structure is shown below:

```
typedef struct {
   unsigned short tg_dflag;       /* dualt tone - 1, single tone - 0 */
   unsigned short tg_freq1;       /* frequency of tone 1 (in Hz) */
   unsigned short tg_freq2;       /* frequency of tone 2 (in Hz) */
   short int      tg_ampl1;       /* amplitude of tone 1 (in dB) */
   short int      tg_ampl2;       /* amplitude of tone 2 (in dB) */
   short int      tg_dur;         /* duration (in 10 ms) */
} TN_GEN;
```

After you build the TN_GEN data structure, there are two ways to define each tone template:

1. Include the values in the structure, or
2. Pass the values to TN_GEN using the **dx_bldtngen( )** function.

After defining the template, pass TN_GEN to **dx_playtone( )** to play the tone.

If you include the values in the structure, you must create a structure for each tone template. If you pass the values using the **dx_playtone( )** function, then you can reuse the structure. If you are only changing one value in a template with many variables, it may be more convenient to use several structures in the code instead of reusing just one.

4. R2MF Signaling

This chapter provides a description of R2MF signaling protocol and the use of R2MF signaling with voice boards.

NOTE: R2MF support is available on all voice boards, D/21D, D/41D, D/41E, D/81A, and D/12x boards).

4.1. R2MF Overview

R2MF signaling is an international signaling system that is used in Europe and Asia to permit the transmission of numerical and other information relating to the called and calling subscribers' lines.

R2MF signals that control the call set-up are referred to as *interregister signals*. In the case of the signals sent between the central office (CO) and the customer premises equipment (CPE), the CO is referred to as the *outgoing register* and the CPE as the *incoming register*. Signals sent from the CO are *forward* signals; signals sent from the CPE are *backward* signals. The outgoing register (CO) sends forward interregister signals and receives backward interregister signals. The incoming register (CPE) receives forward interregister signals and sends backward interregister signals. See Figure 13.

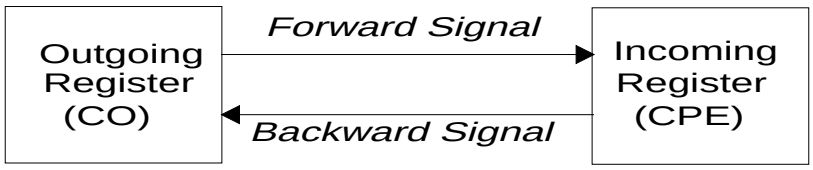

Figure 13. Forward and Backward Interregister Signals

The focus of this section is on the forward and backward interregister signals, and more specifically, the *address* signals, which can provide the telephone number of the called subscriber's line. For national traffic, the address signals can also provide the telephone number of calling subscriber's line for automatic number identification (ANI) applications.

R2MF signals that are used for supervisory signaling on the network are called *line signals*. Line signals are beyond the scope of this document.

4.1.1. Direct Dialing-In Service

Since R2MF address signals can provide the telephone number of the called subscriber's line, the signals may be used by applications providing direct dialing-in (DDI) service, also known as direct inward dialing (DID), and dialed number identification service (DNIS).

DDI service allows an outside caller to dial an extension within a company without requiring an operator to transfer the call. The central office (CO) passes the last 2, 3, or 4 digits of the dialed number to the customer premises equipment (CPE) and the CPE completes the call.

4.1.2. R2MF Multifrequency Combinations

R2MF signaling uses a multifrequency code system based on six fundamental frequencies in the forward direction (1380, 1500, 1620, 1740, 1860, and 1980 Hz) and six frequencies in the backward direction (1140, 1020, 900, 780, 660, and 540 Hz).

Each signal is composed of two out of the six fundamental frequencies, which results in 15 different tone combinations in each direction. Although R2MF is designed for operation on international networks with 15 multifrequency combinations in each direction, in national networks it can be used with a reduced number of signaling frequencies (e.g., 10 multifrequency combinations). See the following tables for a list of the signal tone pairs:

- Table 3: Forward Signals
- Table 4: Backward Signals

Table 3. Forward Signals CCITT Signaling System R2MF Tones

R2MF TONES			DIALOGIC INFORMATION		
Tone Number	**Tone Pair Frequencies (Hz)**		**Group I DefineI**	**Group II Define**	**Tone Detect. ID**
1	1380	1500	SIGI_1	SIGII_1	101
2	1380	1620	SIGI_2	SIGII_2	102
3	1500	1620	SIGI_3	SIGII_3	103
4	1380	1740	SIGI_4	SIGII_4	104
5	1500	1740	SIGI_5	SIGII_5	105
6	1620	1740	SIGI_6	SIGII_6	106
7	1380	1860	SIGI_7	SIGII_7	107
8	1500	1860	SIGI_8	SIGII_8	108
9	1620	1860	SIGI_9	SIGII_9	109
10	1740	1860	SIGI_0	SIGII_0	110
11	1380	1980	SIGI_11	SIGII_11	111
12	1500	1980	SIGI_12	SIGII_12	112
13	1620	1980	SIGI_13	SIGII_13	113
14	1740	1980	SIGI_14	SIGII_14	114
15	1860	1980	SIGI_15	SIGII_15	115

Table 4. Backward Signals CCITT Signaling System R2MF Tones

R2MF TONES			DIALOGIC INFORMATION	
Tone Number	**Tone Pair Frequencies (Hz)**		**Group A Define**	**Group B Define**
1	1140	1020	SIGA_1	SIGB_1
2	1140	900	SIGA_2	SIGB_2
3	1020	900	SIGA_3	SIGB_3
4	1140	780	SIGA_4	SIGB_4
5	1020	780	SIGA_5	SIGB_5
6	900	780	SIGA_6	SIGB_6
7	1140	660	SIGA_7	SIGB_7
8	1020	660	SIGA_8	SIGB_8
9	900	660	SIGA_9	SIGB_9
10	780	660	SIGA_0	SIGB_0
11	1140	540	SIGA_11	SIGB_11
12	1020	540	SIGA_12	SIGB_12
13	900	540	SIGA_13	SIGB_13
14	780	540	SIGA_14	SIGB_14
15	660	540	SIGA_15	SIGB_15

4.1.3. R2MF Signal Meanings

There are 2 groups of meanings associated with each set of signals. Group I meanings and Group II meanings are associated with the 15 forward signals. Group A meanings and Group B meanings are associated with the 15 backward signals. See Figure 14.

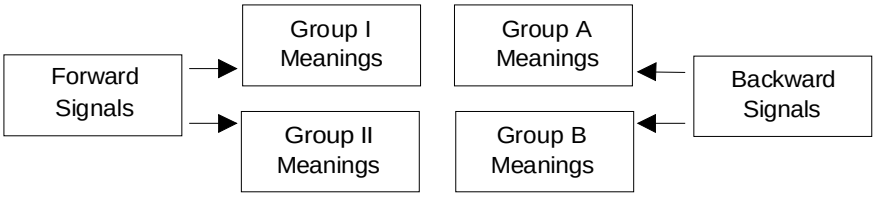

Figure 14. Multiple Meanings for R2MF Signals

In general, Group I forward signals and Group A backward signals are used to control the call set-up and to transfer address information between the outgoing register (CO) and the incoming register (CPE). The incoming register can then signal to the outgoing register to change over to the Group II and Group B meanings.

Group II forward signals provide the calling party's category, and Group B backward signals provide the condition of the called subscriber's line. For further information, see Table 5 describing the purpose of the signal groups and the changeover in meanings.

Signaling must always begin with a Group I forward signal followed by a Group A backward signal that serves to acknowledge the signal just received and also has its own meaning. Each signal then requires a response from the other party. Each response becomes an acknowledgement of the event and an event for the other party to respond to.

Backward signals serve to indicate certain conditions encountered during call set-up or to announce switch-over to changed meanings of subsequent backward signals. Changeover to Group II & Group B meanings allows information about the state of the called subscriber's line to be transferred.

Table 5. Purpose of Signal Groups and Changeover in Meaning

Signal	Purpose
Group I	Group I signals control the call set-up and provide address information.
Group A	Group A signals acknowledge Group I signals (see exception under signal A-5 below) for call set-up, and can also request address and other information. Group A signals also control the changeover to Group II and Group B meanings through the following signals: A-3 Address Complete — Changeover to Reception of Group B Signals: Indicates the address is complete and signals a changeover to Group II/B meanings; after signal A-3 is sent, signaling cannot change back to Group I/A meanings. A-5 Send Calling Party's Category: Requests transmission of a single Group II signal providing the calling party's category. Signal A-5 requests a Group II signal but does not indicate changeover to Group B signals. When the Group II signal requested by A-5 is received, it is acknowledged by a Group A signal; this is an exception to the rule that Group A signals acknowledge Group I signals.
Group II	Group II signals acknowledge signal A-3 or A-5 and provide the calling party category (national or international call, operator or subscriber, data transmission, maintenance or test call).
Group B	Group B signals acknowledge Group II signals and provide the condition of the called subscriber's line. Before Group B signals can be transmitted, the preceding backward signal must have been A-3. Signals cannot change back to Group I/A.

The Incoming Register Backward Signals Can Request:

- Transmission of address
  - Send next digit
  - Send digit previous to last digit sent
  - Send second digit previous to last digit sent
  - Send third digit previous to last digit sent
- Category of the call (the nature and origin)
  - National or international call
  - Operator or subscriber
  - Data transmission
  - Maintenance or test call
- Whether or not the circuit includes a satellite link
- Country code and language for international calls
- Information on use of an echo suppressor

The Incoming Register Backward Signals Can Indicate:

- Address complete — send category of call
- Address complete — put call through
- International, national, or local congestion
- Condition of subscriber's line
  - Send SIT to indicate long term unavailability
  - Line busy
  - Unallocated number
  - Line free — charge on answer
  - Line free — no charge on answer (only for special destinations)
  - Line out of order

NOTE: The meaning of certain forward multifrequency combinations may also vary depending upon their position in the signaling sequence. For example, with terminal calls the first forward signal transmitted in international working is a language or discriminating digit (signals I-1 through I-10). When the same signal is sent as other than the first signal, it usually means a numerical digit.

See the following tables for the signal meanings:

- Table 6. Meanings for R2MF Group I Forward Signals
- Table 7. Meanings for R2MF Group II Forward Signals
- Table 8. Meanings for R2MF Group A Backward Signals
- Table 9. Meanings for R2MF Group B Backward Signals

Table 6. Meanings for R2MF Group I Forward Signals

Tone Number	Dialogic Define	(A) Primary Meaning (B) Secondary Meaning
1	SIGI_1	**(A)** Digit 1 **(B)** Language digit-French
2	SIGI_2	**(A)** Digit 2 **(B)** Language digit-English
3	SIGI_3	**(A)** Digit 3 **(B)** Language digit-German
4	SIGI_4	**(A)** Digit 4 **(B)** Language digit-Russian
5	SIGI_5	**(A)** Digit 5 **(B)** Language digit-Spanish
6	SIGI_6	**(A)** Digit 6 **(B)** Spare (language digit)
7	SIGI_7	**(A)** Digit 7 **(B)** Spare (language digit)
8	SIGI_8	**(A)** Digit 8 **(B)** Spare (language digit)
9	SIGI_9	**(A)** Digit 9 **(B)** Spare (discriminating digit)
10	SIGI_0	**(A)** Digit 0 **(B)** Discriminating digit
11	SIGI_11	**(A)** Access to incoming operator (Code 11) **(B)** Country code indicator: outgoing half-echo suppressor required
12	SIGI_12	**(A)** Access to delay operator (code 12); request not accepted **(B)** Country code indicator: no echo suppressor required
13	SIGI_13	**(A)** Access to test equipment (code 13); satellite link not included **(B)** Test call indicator
14	SIGI_14	**(A)** Incoming half-echo suppressor required; satellite link included **(B)** Country code indicator: outgoing half-echo suppressor inserted
15	SIGI_15	**(A)** End of pulsing (code 15); end of identification **(B)** Signal not used

Table 7. Meanings for R2MF Group II Forward Signals

Tone Number	Dialogic Define	Meaning
1	SIGII_1	National: Subscriber without priority
2	SIGII_2	National: Subscriber with priority
3	SIGII_3	National: Maintenance equipment
4	SIGII_4	National: Spare
5	SIGII_5	National: Operator
6	SIGII_6	National: Data transmission
7	SIGII_7	International: Subscriber, operator, or maintenance equipment (without forward transfer)
8	SIGII_8	International: Data transmission
9	SIGII_9	International: Subscriber with priority
10	SIGII_0	International: Operator with forward transfer facility
11	SIGII_11	Spare for national use
12	SIGII_12	Spare for national use
13	SIGII_13	Spare for national use
14	SIGII_14	Spare for national use
15	SIGII_15	Spare for national use

Table 8. Meanings for R2MF Group A Backward Signals

Tone Number	Dialogic Define	Meaning
1	SIGA_1	Send next digit (n+1)
2	SIGA_2	Send last but one digit (n-1)
3	SIGA_3	Address complete; change to Group B signals; no change back
4	SIGA_4	Congestion in the national network
5	SIGA_5	Send calling party's category; change to Group II; can change back
6	SIGA_6	Address complete; charge; set-up speech conditions
7	SIGA_7	Send last but two digit (n-2)
8	SIGA_8	Send last but three digit (n-3)
9	SIGA_9	Spare for national use
10	SIGA_0	Spare for national use
11	SIGA_11	Send country code indicator
12	SIGA_12	Send language or discriminating digit
13	SIGA_13	Send nature of circuit (satellite link only)
14	SIGA_14	Request for information on use of an echo suppressor (Is an incoming half-echo suppressor required?)
15	SIGA_15	Congestion in an international exchange or at its output

Table 9. Meanings for R2MF Group B Backward Signals

Tone Number	Dialogic Define	Meaning
1	SIGB_1	Spare for national use
2	SIGB_2	Send special information tone to indicate long-term unavailability
3	SIGB_3	Subscriber line busy
4	SIGB_4	Congestion encountered after change to Group B
5	SIGB_5	Unallocated number
6	SIGB_6	Subscriber line free; charge on answer
7	SIGB_7	Subscriber line free; no charge (only for calls to special destinations)
8	SIGB_8	Subscriber line out of order
9	SIGB_9	Spare for national use
10	SIGB_0	Spare for national use
11	SIGB_11	Spare for national use
12	SIGB_12	Spare for national use
13	SIGB_13	Spare for national use
14	SIGB_14	Spare for national use
15	SIGB_15	Spare for national use

4.1.4. R2MF Compelled Signaling

R2MF interregister signaling uses forward and backward compelled signaling. Simply put, with compelled signaling each signal is sent until it is responded to by a return signal, which in turn is sent until responded to by the other party. Each

signal stays on until the other party responds, thus compelling a response from the other party.

Reliability and speed requirements for signaling systems are often in conflict, the faster the signaling, the more unreliable it is likely to be. Compelled signaling provides a balance between speed and reliability because it adapts its signaling speed to the working conditions with a minimum loss of reliability.

The R2MF signal is composed of two significant events, tone-on and tone-off. Each tone event requires a response from the other party. Each response becomes an acknowledgement of the event and an event for the other party to respond to.

Compelled signaling must always begin with a Group I forward signal.

- The CO starts to send the first forward signal.
- As soon as the CPE recognizes the signal, it starts to send a backward signal that serves as an acknowledgement and at the same time has its own meaning.
- As soon as the CO recognizes the CPE acknowledging signal, it stops sending the forward signal.
- As soon as the CPE recognizes the end of the forward signal, it stops sending the backward signal.
- As soon as the CO recognizes the CPE end of the backward signal, it may start to send the next forward signal.

The CPE responds to a tone-on with a tone-on and to a tone-off with a tone-off. The CO responds to a tone-on with a tone-off and to a tone-off with a tone-on.

Refer to the following figures for more information:

- Figure 15: R2MF Compelled Signaling Cycle
- Figure 16: Example of R2MF Signals for 4-digit DDI Application

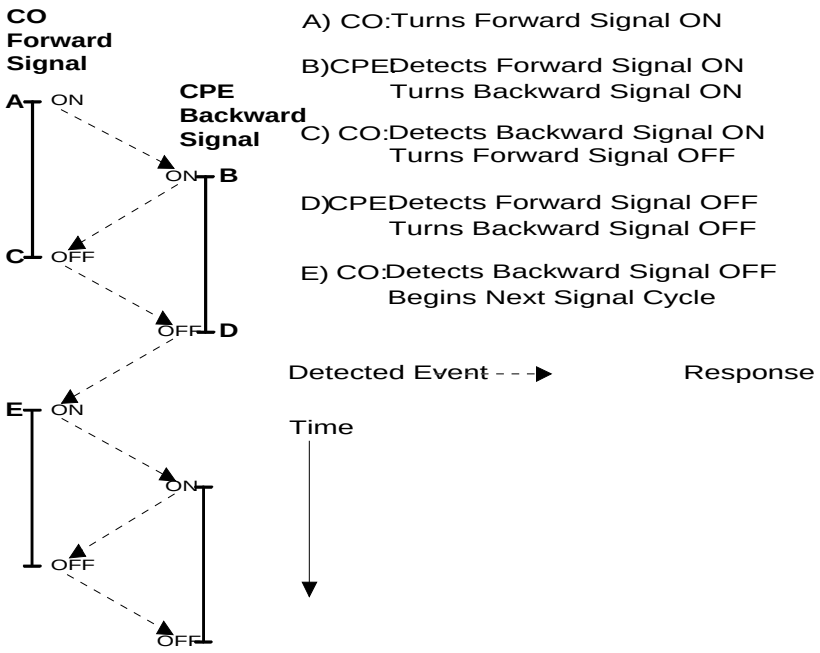

Figure 15. R2MF Compelled Signaling Cycle

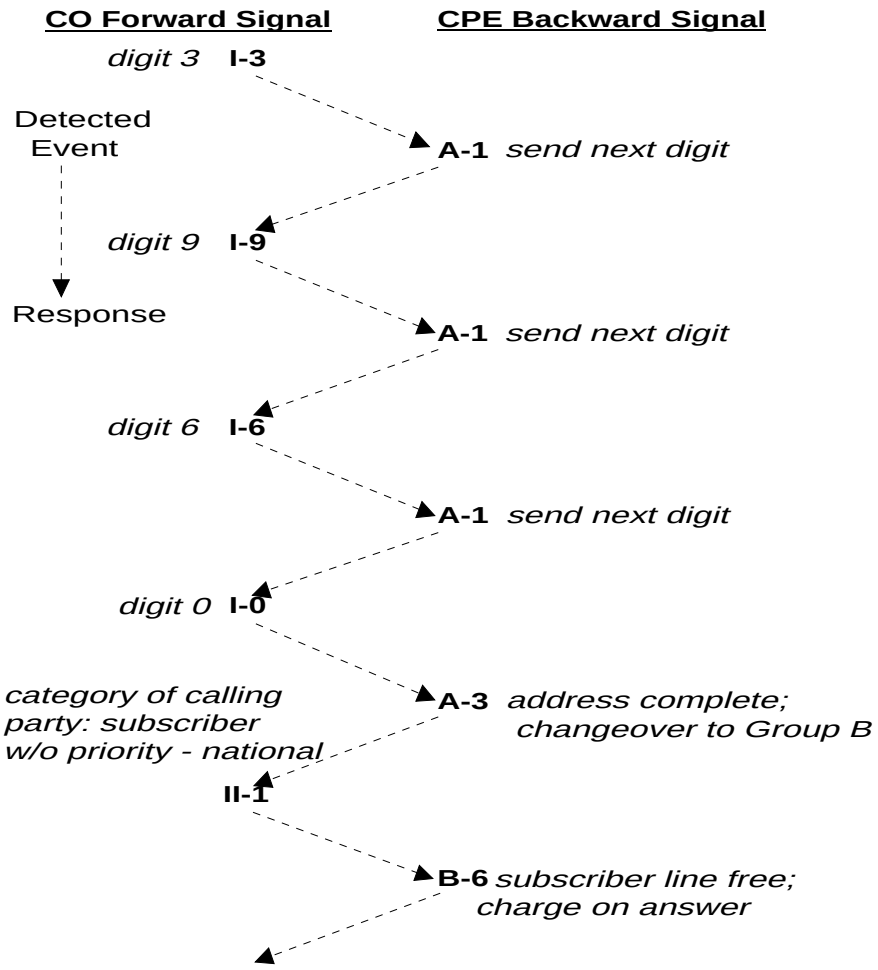

Figure 16. Example of R2MF Signals for 4-digit DDI Application

4.1.5. Related Publications

For more information on R2MF signaling, you can refer to the following publications:

Specifications of Signalling Systems R1 and R2, International Telegraph and Telephone Consultative Committee (CCITT), Blue Book Volume VI, Fascicle VI.4, ISBN 92-61-03481-0

General Recommendations on Telephone Switching and Signalling, International Telegraph and Telephone Consultative Committee (CCITT), Blue Book Volume VI, Fascicle VI.1, ISBN 92-61-03451-9

4.2. Using R2MF Signaling with Voice Boards

The Voice Software support for R2MF signaling is based upon Global Tone Detection and Global Tone Generation.

The following R2MF functions allow you to define R2MF tones for detecting the forward signals and to play the backward signals in the correct timing sequence required by the compelled signaling procedure:

r2_creatfsig( ): Create R2MF Forward Signal Tone
r2_playbsig( ): Play R2MF Backward Signal

See *Chapter 3* in the *Voice Programmer's Guide for Windows NT* for a detailed description of these functions.

Four sets of defines are provided to specify the 15 Group I and 15 Group II forward signals, and the 15 Group A and 15 Group B backward signals. Table 6, Table 7, Table 8, and Table 9 in *Section 4.1.3. R2MF Signal Meanings* provide a list of these defines.

4.2.1. R2MF Tone Detection Template Memory Requirements

To implement R2MF signaling the board must have sufficient memory blocks to create the number of user-defined tones required by your application. Your application may not need to detect all 15 forward signals, especially if you do not need to support R2MF signaling for international calls. If that is the case, your application can work with a reduced number of R2MF tones.

The D/12x, D/81A, and DIALOG/HD boards (e.g., D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1, and D/320SC) normally contain sufficient memory to

create the necessary R2MF tones. However, you should be aware of the maximum number of user-defined tones (including non-R2MF tones) allowed on the board. Refer to *Section 3.2.5. Maximum Number of Tone Templates*, for more information.

The D/21D, D/4xD, D/21E, D/41E, and D/41ESC boards may also be able to create all 15 R2MF tones due to the overlap in frequencies for the R2MF signals. If creating these tones exceeds the maximum number of tones allowed, you may be able to support R2MF signaling through a reduced number of R2MF user-defined tones. Refer to *Section 3.2.5. Maximum Number of Tone Templates*.

5. Speed and Volume Control

5.1. Speed and Volume Control Overview

This chapter describes how to control play-speed and play-volume.

5.2. Voice Software Speed and Volume Support

Speed and Volume control are available on D/21D, D/21E, D/41D, D/41E, D/41ESC, D/81A, D/121B, D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1 and D/320SC boards only. Do not use the Speed and Volume control functions to control speed on the D/120, D/121, or D/121A boards.

The Voice Software contains functions and data structures to control the speed and volume of play on a channel. This allows an end user to control the speed or volume of a message by entering a DTMF tone, for example.

NOTE: Speed can be controlled on playbacks using 24 kbps or 32 kbps ADPCM only. Volume can be controlled on all playbacks regardless of the encoding algorithm.

5.2.1. Speed and Volume Convenience Functions

The convenience functions set a digit that will adjust speed or volume, but do not use any data structures. These convenience functions will only function properly if you use the default settings of the Speed or Volume Modification Tables. These functions assume that the Modification Tables have not been modified. The speed or volume convenience functions are:

- **dx_addspddig( )** - adds a digit that will modify speed by a specified amount.
- **dx_addvoldig( )** - adds a digit that will modify volume by a specified amount.

5.2.2. Speed and Volume Adjustment Functions

Speed or volume can be adjusted explicitly or can be set to adjust in response to a pre-set condition, such as a specific digit. For example, speed could be set to increase a certain amount when "1" is pressed on the telephone keypad. The functions used for speed and volume adjustment are:

- **dx_setsvcond( )** - sets conditions that adjust speed or volume. Use this if you want to adjust speed or volume in response to a DTMF digit, or start of play.
- **dx_adjsv( )** - adjusts speed or volume explicitly. Use this if your adjustment condition is not a digit or start of play. For example, the application could call this function after detecting a spoken word (voice recognition) or a certain key on the keyboard.

See *Chapter 3* in the *Voice Programmer's Guide for Windows NT* for detailed information about these functions.

5.2.3. Speed and Volume Modification Tables

Each channel has a Speed or Volume Modification Table for play speed or play volume adjustments. Except for the value of the settings, the table is the same for speed and volume.

Each Speed or Volume Modification table has 21 entries, 20 entries that allow for a maximum of 10 increases and decreases in speed or volume; the entry in the middle of the table is referred to as the "origin" entry that represents normal speed or volume. The normal speed or volume is how playback occurs when the Speed and Volume Control feature is not used. The normal speed or volume is the basis for all settings in the table.

Typically, the origin is set to 0. The normal speed or volume is the basis for all settings in the table. Speed and volume increases or decreases by moving up or down the tables.

Other entries in the table specify a speed or volume setting in terms of a deviation from normal. For example, if a Speed Modification Table (SMT) entry is -10, this value represents a 10% decrease from the normal speed.

Although the origin is typically set to normal speed/volume, changing the setting of the origin does not affect the other settings, because all values in the SVMT are based on a deviation from normal speed/volume.

Speed and Volume Control adjustments are specified by moving the current speed/volume pointer in the table to another SVMT table entry; this translates to increasing or decreasing the current speed/volume to the value specified in the table entry.

A speed/volume adjustment stays in effect until the next adjustment on that channel or until a system reset.

The SVMT is like a 21-speed bicycle. You can select the gear ratio for each of the 21 speeds before you go for a ride (by changing the values in the SVMT). And you can select any gear once you are on the bike, like adjusting the current speed/volume to any setting in the SVMT.

The Speed or Volume Modification Table can be set or reset using the **dx_setsvmt( )** function which uses the DX_SVMT data structure. The current values of these tables can also be returned to the DX_SVMT structure using **dx_getsvmt( )**. The DX_SVCB data structure uses this table when setting adjustment conditions. See *Chapter 4* in the *Voice Programmer's Guide for Windows NT* for information about the DX_SVMT and DX_SVCB data structures.

Adjustments to speed or volume are made by **dx_adjsv( )** and **dx_setsvcond( )** according to the Speed or Volume Modification Table settings. These functions adjust speed or volume to one of the following:

- a specified level (i.e., to a specified absolute position in the speed table or volume table)
- a change in level (i.e., by a specified number of steps up or down in the speed table or volume table)

For example:

By default, each entry in the Volume Modification Table is equivalent to 2 decibels from the origin. Volume could be decreased by 2 decibels by specifying position 1 in the table, or by moving 1 step down.

The Speed Modification Table is shown in Table 10 Each entry in the table is a percentage deviation from the default play speed ("origin"). For example, the decrease[6] position reduces speed by 40%. This is four steps from the origin.

Table 10. Speed Modification Table

Table Entry	Default Value (%)	Absolute Position
decrease[0]	-128 (80h)	-10
decrease[1]	-128 (80h)	-9
decrease[2]	-128 (80h)	-8
decrease[3]	-128 (80h)	-7
decrease[4]	-128 (80h)	-6
decrease[5]	-50	-5
decrease[6]	-40	-4
decrease[7]	-30	-3
decrease[8]	-20	-2
decrease[9]	-10	-1
origin		0
increase[0]	+10	1
increase[1]	+20	2
increase[2]	+30	3
increase[3]	+40	4
. .	. .	.
. .	. .	.
increase[9]	+100	10

NOTE: In this table, the lowest position utilized is the decrease[5] position. The remaining decrease fields are set to -128 (80h). If these "non-adjustment" positions are selected, the default action is to play at the decrease[5] speed. These fields can be reset, as long as no values lower than -50 are

used (e.g., you could spread the 50% speed decrease over 10 steps rather than 5).

The Volume Modification Table is shown in Table 11 Each entry in the table is a deviation of *n* dB from the starting point or volume ("origin"). For example, the increase[1] position increases volume by 4 dB. This is two steps from the origin.

Table 11. Volume Modification Table

Table Entry	Default Value (dB)	Absolute Position
decrease[0]	-20	-10
decrease[1]	-18	-9
decrease[2]	-16	-8
decrease[3]	-14	-7
decrease[4]	-12	-6
decrease[5]	-10	-5
decrease[6]	-08	-4
decrease[7]	-06	-3
decrease[8]	-04	-2
decrease[9]	-02	-1
origin	0	
increase[0]	+02	1
increase[1]	+04	2
increase[2]	+06	3
increase[3]	+08	4
increase[4]	+10	5
increase[5]	-128 (80h)	
.	.	
.	.	
increase[9]	-128 (80h)	

NOTE: In this table, the highest position utilized is the increase[4] position. The remaining increase fields are set to -128 (80h). If these "non-adjustment" positions are selected, the default action is to play at the increase[4] volume. These fields can be reset, as long as no values higher than +10 are used (e.g., you could spread the 10 dB volume increase over 10 steps rather than 5).

5.2.4. Play Adjustment Digits

The Voice Software processes play adjustment digits differently from normal digits:

- If a play adjustment digit is entered during playback, it causes a play adjustment only and has no other effect. This means that the digit is not added to the digit queue, it cannot be retrieved with the **dx_getdig( )** or **dx_getdigbuf( )** function, and it does not affect any termination condition.
- If the digit queue contains adjustment digits when a play begins and play adjustment is set to be level sensitive, the digits will affect the speed or volume and then be removed from the queue.

5.3. Using Speed and Volume Control

This section describes the steps for the following:

- Setting play adjustment conditions
- Adjusting play explicitly

5.3.1. Setting Adjustment Conditions

Adjustment conditions are set in the same way for speed or volume. The following steps describe how to set conditions upon which volume should be adjusted:

1. Set up the Volume Modification Table (if you do not want to use the defaults):
    - Set up the DX_SVMT structure to specify the size and number of the steps in the table.
    - Call the **dx_setsvmt( )** function, which points to the DX_SVMT structure, to modify the Volume Modification Table (**dx_setsvmt( )** can also be used to reset the table to its default values).

2. Set up the DX_SVCB structure to specify the condition, the size and type of adjustment.
3. Call **dx_setsvcond( )**, which points to an array of DX_SVCB structures. All subsequent plays will adjust volume as required whenever one of the conditions specified in the array occur.

See *Chapter 3* in the *Voice Programmer's Guide for Windows NT* for more information on **dx_setsvcond( )** and **dx_setsvmt( )**. See *Chapter 4* in the *Voice Programmer's Guide for Windows NT* for information on the DX_SVMT and DX_SVCB structures.

5.3.2. Explicitly Adjusting Speed and Volume

Speed and volume adjustments are made in the same way. The following is an example of the steps you should take to adjust speed, but you can use exactly the same procedure for volume:

1. Set up the Speed Modification Table (if you do not want to use the defaults):
    - Set up the DX_SVMT structure to specify the size and number of the steps in the table.
    - Call the **dx_setsvmt( )** function, which points to the DX_SVMT structure, to modify the Speed Modification Table (**dx_setsvmt( )** can also be used to reset the table to its default values).
2. When required, call **dx_adjsv( )** to adjust the Speed Modification Table, by specifying the size and type of the adjustment.

See *Chapter 3* in the *Voice Programmer's Guide for Windows NT* for more information on **dx_adjsv( )** and **dx_setsvmt( )**. See *Chapter 4* in the *Voice Programmer's Guide for Windows NT* for information about the DX_SVMT structure.

6. Voice Features Demonstration Programs

This chapter provides instructions for using the demonstration programs provided. There are two demonstration programs supplied with this System Release for Windows NT:

- Multithreaded Text Based Application Program
- Multithreaded GUI Based Application Program

6.1. Multithreaded Text Based Application Program

The Dialogic Multithreaded Text Based Application Program performs asynchronous plays and records on up to forty-eight voice channels, using one thread per channel.

To run the program, change to the directory where the demo is located (<install drive>:\<install directory>\dialogic\samples\voice). Then type **testmt -b***x* **-t***y* to perform plays and records on *y* channels, where *x* is the first board the program tests (default is 1) and *y* is the number of threads to create in the system (default is 4).

NOTE: **testmt -?** will display demo options.

NOTE: To exit the demo program prior to completion of the demonstration, press CTRL C.

6.2. Multithreaded GUI Based Voice Features Application Program

The Multithreaded GUI Based Application Program is a multichannel Dialogic voice GUI based demonstration program for Windows NT. The files are located in the base directory *<install drive>:\<install directory>\dialogic\samples\voice.*

The purpose of the Multithreaded GUI Based Application Program is to show how to use the API functions in an asynchronous voice application within the

Windows programming environment. As is the case with most small Windows programs, the majority of the code is for graphical user interface support.

The Multithreaded GUI Based Application Program demonstrates the following functions:

- Set hook state
- Play VOX and WAV files
- Adjust playback volume
- Record to VOX and WAV files
- Play a user defined tone
- Make a call with or without PerfectCall
- Wait for digits

The Multithreaded GUI Based Application Program is a MDI application running with two threads. The main thread creates a child window for each voice device that is opened. The main thread will also initiate the selected voice function on the active voice device (foreground child window). This thread will print a brief message in the child window and will gray out menu items that should not be selected while a channel is busy.

The purpose of the second thread is to poll for device events using **sr_waitevt( )**. Upon receiving an event, a brief message is displayed in the child window and menu items are enabled for the next function selection.

The 'Options' Menu allows the user to select PerfectCall options, adjust playback volume, define tone for playback, and select WAVE recording options. These settings are on a per channel basis and are stored in the structure CHINFO. When a device is opened, its CHINFO structure is initialized with default settings.

NOTES: **1.** Linear PCM is not supported on the D/41D

2. Volume for playback can be adjusted at any time

6.3. Running the Multithreaded GUI Based Application Program for Voice Boards

Before running the Multithreaded GUI Based Application Program for Voice Boards, ensure that you perform the following:

1. Connect a Dialogic PromptMaster, a Central Office Simulator, or PBX to the Dialogic voice board.
2. **Start** the Dialogic boards using the Dialogic Configuration Manager.

To run the Multithreaded GUI Based Application Program for Voice Boards:

1. Select the *Multithreaded GUI Based Application* icon in the Dialogic Development Package folder, or change to the directory where the demo is located and type SAMPLE.EXE.
2. Select Open from the File Menu.
3. Click OK to open the channel (default: DxxxB1C1 - board 1, channel 1). You can open multiple channels concurrently.
4. Select Off-hook from the Function Menu.

You can now select any of the operations shown on the Function menu and modify their operation using the Options Menu.

Appendix A

Related Voice Publications

Voice Publications Overview

For more information on Voice hardware and software products see the following Dialogic publications:

Voice Publications Listing

- For information about the SCbus and SCbus routing, see the *SCbus Routing Guide* and the *SCbus Routing Function Reference for Windows NT*.
- For information about the Voice and Diagnostic Libraries and about library data structures, see the *Voice Programmer's Guide for Windows NT* in this reference.
- For information about the Standard Runtime Library, see the *Standard Runtime Library Programmer's Guide for Windows NT* in this reference.
- For information about installing Voice software, see the *System Release Software Installation Reference for Windows NT*.
- For information about the D/2x, D/4x, D/81A, D/12x and D/xxxSC (D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1, and D/320SC) Voice boards, see the *Voice Hardware Reference*.
- For information about the digital interface device functions for the D/240SC-T1 and D/300SC-E1 boards, see the *Digital Network Interface Software Reference for Windows NT*.
- For information about the primary rate functions, see the *Primary Rate Software Reference for Windows NT*.

Glossary

Adaptive Differential Pulse Code Modulation: See *ADPCM*.

ADPCM: Adaptive Differential Pulse Code Modulation. A sophisticated compression algorithm for digitizing audio that stores the differences between successive samples rather than the absolute value of each sample. This method of digitization also reduces storage requirements from 64K bits/second to as low as 24K bits/second.

AGC: Automatic Gain Control. An electronic circuit used to maintain the audio signal volume at a constant level.

analog: 1. A method of telephony transmission in which the signals from the source (for example, speech in a human conversation) are converted into an electrical signal that varies continuously over a range of amplitude values analogous to the original signals. 2. Not digital signaling. 3. Used to refer to applications that use loop start signaling.

ANI: Automatic Number Identification.

Antares: A Dialogic open platform for easily incorporating speech recognition, Text-To-Speech, fax and many other DSP technologies. Dialogic PC-based expansion board with four TI floating point DSPs, SPOX DSP operating system, and the Antares board downloadable firmware and device driver.

ASCIIZ string: A null-terminated string of ASCII characters.

asynchronous function: A function that allows program execution to continue without waiting for a task to complete. To implement an asynchronous function, an application-defined event handler must be enabled to trap and process the completed event. See *synchronous function*.

AT: Used to describe an IBM or IBM-compatible Personal Computer (PC) containing an 80286 or higher microprocessor, a 16-bit bus architecture, and a compatible BIOS.

AT bus: The common communication channel in a PC AT. The channel uses a 16-bit data path architecture, which allows up to 16 bits of data transfer. This bus architecture includes the standard PC bus plus a set of 36 lines for additional data transmission, addressing, and interrupt request handling.

Automatic Gain Control: See *AGC*.

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

bit mask: A pattern which selects or ignores specific bits in a bit mapped control or status field.

bitmap: An entity of data (byte or word) in which individual bits contain independent control or status information.

board device: A board-level object that can be manipulated by a physical library. Board devices can be real physical devices, such as a D/4x board, or emulated devices, such as one of the D/4x boards that is emulated by a D/81A, D/12x or D/xxxSC board.

Board Locator Technology: Operates in conjunction with a rotary switch to determine and set non-conflicting slot and IRQ interrupt-level parameters, thus eliminating the need to set confusing jumpers or DIP switches.

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

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

busy device: A device that is stopped, being configured, has a multitasking or non-multitasking function, or I/O function active on it.

cadence: A rhythmic sequence or pattern. Once established, it can be classified as a single ring, a double ring, or a busy signal by comparing the periods of sound and silence to establish parameters.

cadence detection: A voice driver feature that analyzes the audio signal on the line to detect a repeating pattern of sound and silence.

Call Progress Analysis: The process used to automatically determine what happened after an outgoing call is dialed.

Call Status Transition Event Functions: Functions that set and monitor events on devices.

CCITT: Comite Consultatif Internationale de Telegraphique et Telephonique. One of the four permanent parts of the International Telecommunications Union, a United Nations agency based in Geneva. The CCITT is divided into three

sections: 1. Study Groups set up standards for telecommunications equipment, systems, networks, and services. 2. Plan Committees develop general plans for the evolution of networks and services. 3. Specialized Autonomous Groups produce handbooks, strategies, and case studies to support developing countries.

channel device: A channel-level object that can be manipulated by a physical library, such as an individual telephone line connection. A channel is also a subdevice of a board. See *subdevice*.

channel: 1. When used in reference to a Dialogic expansion board that is analog, an audio path, or the activity happening on that audio path (for example, when you say the channel goes off-hook). 2. When used in reference to a Dialogic expansion board that is digital, a data path, or the activity happening on that data path. 3. When used in reference to a bus, an electrical circuit carrying control information and data.

CO: Central Office. A local phone exchange. In general, "CO' refers to the phone network exchange that provides your phone lines. The term "Central Office" is used in North America. The rest of the world calls it PTT, for Post, Telephone and Telegraph. The telephone company facility where subscriber lines are linked, through switches, to other subscriber lines (including local and long distance lines).

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

configuration file: An unformatted ASCII file that stores device initialization information for an application.

Configuration Functions: Functions that alter the configuration of devices.

Convenience Functions: Functions that simplify application writing.

D/81A: 8 port DSP-based voice board that runs SpringWare firmware. Connects via PEB to a standalone telephone network interface board.

D/120: A 12-channel voice board from Dialogic that consists of a SpringBoard-based expansion device and downloaded software. On the PEB bus, the D/120 serves as a resource module to the installed network module.

D/121 A 12-channel voice-store-and-forward product from Dialogic with all the features of the D/120 plus patented call analysis algorithms for outbound applications and multifrequency (MF) tone capability.

D/12x System: A Voice System that uses D/12x boards. See *Voice System.*

D/121A: A 12-channel voice board from Dialogic with all the features of the D/121 plus additional RAM, increased performance and reliability, and improved downstream compatibility.

D/121B: 12 port DSP-based voice board that runs SpringWare firmware. Connects via PEB to a standalone telephone network interface board.

D/12x: Any model of the Dialogic series of 12-channel voice-store-and-forward expansion boards for the AT-bus architecture. Includes: D/120 and D/121 boards.

D/160SC-LS: 16 port DSP-based voice board that runs SpringWare firmware and has onboard analog loop start telephone interfaces and an SCbus interface.

D/21D, D/41D: 2 and 4 port DSP-based voice boards with onboard analog telephone interface; runs SpringWare downloadable firmware.

D/21E, D/41E: 2 and 4 port DSP-based voice boards with onboard analog telephone interface; runs SpringWare downloadable firmware.

D/2x: A term used to refer to any 2-channel voice-store-and-forward expansion board made by Dialogic.

D/40: A model of 4-channel voice-store-and-forward expansion board by Dialogic with an on-board processor and shared RAM. The D/40 features real-time digitization, compression and playback of audio, DTMF reception, automatic answering, DTMF or rotary pulse dialing, and direct connection to telephone lines.

D/41: A model of the four-channel voice-store-and-forward expansion boards by Dialogic that has all of the features of a D/40 plus patented call analysis algorithms for outbound applications.

D/4x: Any model of the Dialogic series of 4-channel voice-store-and-forward expansion boards for the AT-bus architecture. Includes D/4xD and D/41E boards.

D/240SC: 24 port DSP-based voice board that runs SpringWare firmware and has an onboard SCbus interface. Connects to a standalone telephone network interface board.

D/240SC-T1: 24 port DSP-based voice board that runs SpringWare firmware and has an onboard digital T-1 telephone interface and an SCbus interface.

D/300SC-E1: 30 port DSP-based voice board that runs SpringWare firmware and has an onboard digital E-1 telephone interface and an SCbus interface.

D/320SC: 30 port DSP-based voice board that runs SpringWare firmware and has an onboard SCbus interface. Connects to a standalone telephone network interface board.

data structure: Programming term for a data element consisting of fields, where each field may have a different type definition and length. A group of data structure elements usually share a common purpose or functionality.

debouncing: Eliminating false signal detection by filtering out rapid signal changes. Any detected signal change must last for the minimum duration as specified by the debounce parameters before the signal is considered valid. Also known as deglitching.

deglitching: Eliminating false signal detection by filtering out rapid signal changes. Any signal change shorter than that specified by the deglitching parameters is ignored.

device: A computer peripheral or component controlled through a software device driver. A Dialogic vioce and/or network interface expansion board is considered a physical board containing one or more logical board devices, and each channel or time slot on the board is a device.

device channel: A Dialogic voice data path that processes one incoming or outgoing call at a time (equivalent to the terminal equipment terminating a phone line). There are 4 device channels on a D/4x, 12 on a D/12x, 16 on a

D/160SC-LS, 24 on a D/240SC or D/240SC-T1, 30 on a D/300SC-E1, and 32 on a D/320SC board.

device driver: Software that acts as an interface between an application and hardware devices.

device handle: Numerical reference to a device, obtained when a device is opened using **xx_open( )**, where *xx* is the prefix defining the device to be opened. The device handle is used for all operations on that device.

Device Management Functions: Functions that open and close devices.

device name: Literal reference to a device, used to gain access to the device via an **xx_open( )** function, where *xx* is the prefix defining the device to be opened.

DIALOG/HD Series: Dialogic High Density products, including the D/160SC-LS, D/240SC, D/240SC-T1, D/300SC-E1, and D/320SC, provide a powerful set of advanced computer telephony features that developers can use to create cost-efficient, high-density systems.

digitize: The process of converting an analog waveform into a digital data set.

download: The process where board level program instructions and routines are loaded during board initialization to a reserved section of shared RAM.

downloadable SpringWare firmware: Software features loaded to Dialogic voice hardware. Features include voice recording and playback, enhanced voice coding, tone detection, tone generation, dialing, call progress analysis, voice detection, answering machine detection, speed control, volume control, ADSI support, automatic gain control, and silence detection.

driver: A software module which provides a defined interface between an application program and the firmware interface.

DSP-based boards: 1. Digital signal processor. A specialized microprocessor designed to perform speedy and complex operations with digital signals. 2. Digital signal processing.

DTI/: (Digital Telephony Interface) The naming convention used with Dialogic boards such as the DTI/211. This interface is designed to work with the T-1 telephony standard used in North American and Japanese markets. A general term used to refer to any Dialogic digital telephony interface device.

DTI/211: 24 port standalone telephone network interface for use with voice-only boards; digital T-1 interface.

DTI/212:: 24 port standalone telephone network interface for use with voice-only boards; digital E-1 interface.

DTI/2xx Refer's to Dialogic's DTI/211, DTI/212, DTI/240SC, and DTI/300SC digital telephony interface boards.

DTI/xxx: Refers to any of Dialogic's second-generation digital telephony interface boards.

DTMF digits: Dual Tone Multi Frequency. Push button or touch tone dialing based on transmitting a high and a low frequency tone identify each digit on a telephone keypad. The tones are (Hz):

1: 697,1209	2: 697,1336	3:697,1477
4: 770,1209	5: 770,1336	6: 770,1477
7: 852,1209	8: 852,1336	9: 852,1477
0: 941,1336	*: 941,1209	#: 941,1477

emulated device: A virtual device whose software interface mimics the interface of a particular physical device, such as a D/4x boards that is emulated by a D/12x or a D/xxxSC board. On a functional level, a D/12x board is perceived by an application as three D/4x boards. See *physical device*.

event: An unsolicited or asynchronous message from a hardware device to an operating system, application, or driver. Events are generally attention-getting messages, allowing a process to know when a task is complete or when an external event occurs.

event handler: A portion of a Dialogic application program designed to trap and control processing of device-specific events. The rules for creating a DTI/1xx event handler are the same as those for creating a Windows NT signal handler.

Event Management functions: Class of device-independent functions (contained in the Standard Runtime Library) that connect events to application-specified event handlers, allowing users to retrieve and handle events that occur on the device. See *Standard Runtime Library*.

Extended Attribute functions: Class of functions that take one input parameter (a valid Dialogic device handle) and return device-specific information. For instance, a Voice device's Extended Attribute function returns information

specific to the Voice devices. Extended Attribute function names are case-sensitive and must be in capital letters. See Standard Runtime Library.

firmware: A set of program instructions that reside on an expansion board.

flash: A signal which consists of a momentary on-hook condition used by the Voice hardware to alert a telephone switch. This signal usually initiates a call transfer.

frequency detection: A voice driver feature that detects the tri-tone Special Information Tone (SIT) sequences and other single-frequency tones for call progress analysis.

Global Tone Detection: A feature that allows the creation and detection of user-defined tone descriptions on a channel by channel basis.

hook state: A general term for the current line status of the channel: either on-hook or off-hook. A telephone station is said to be on-hook when the conductor loop between the station and the switch is open and no current is flowing. When the loop is closed and current is flowing the station is off-hook. These terms are derived from the position of the old fashioned telephone set receiver in relation to the mounting hook provided for it.

hook switch: The name given to the circuitry which controls on-hook and off-hook state of the Voice device telephone interface.

I/O Functions: Functions that transfer data to and from devices.

I/O: Input-Output

idle device: A device that has no functions active on it.

interrupt request level: A signal sent to the central processing unit (CPU) to temporarily suspend normal processing and transfer control to an interrupt handling routine. Interrupts may be generated by conditions such as completion of an I/O process, detection of hardware failure, power failures, etc.

IRQ: Interrupt ReQuest. A signal sent to the CPU to temporarily suspend normal processing and transfer control to an interrupt handling routine. A means of toggling between applications so that your system does not crash.

kernel: A set of programs in an operating system that implement the system's functions.

loop: The physical circuit between the telephone switch and the D/xxx board.

loop current: The current that flows through the circuit from the telephone switch when the Voice device is off-hook.

loop current detection: A voice driver feature that returns a connect after detecting a loop current drop.

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

LSI/120: A Dialogic 12-line loop start interface expansion board.

off-hook: The state of a telephone station when the conductor loop between the station and the switch is closed and current is flowing. When a telephone handset is lifted from its cradle (or equivalent condition), the telephone line state is said to be off-hook.

PC: Personal Computer. In this manual, the term refers to an IBM Personal Computer or compatible machine.

PCM Expansion Bus: See PEB.

PEB: PCM Expansion Bus. A Dialogic open platform, digital voice bus for electrically and digitally connecting different voice processing components. Information on the PEB is encoded using the Pulse Code Modulation (PCM) method. Non-Dialogic products using PCM encoding may interface with Dialogic products by using this bus.

PerfectDigit: Dialogic SpringWare DTMF or MF signaling.

PerfectLevel: Dialogic SpringWare Volume control.

PerfectPitch: Dialogic SpringWare Speed control**.**

PerfectVoice: Dialogic SpringWare Enhanced voice coding.

physical device: A device that is an actual piece of hardware, such as a D/4x board; not an emulated device. See emulated device.

polling: The process of repeatedly checking the status of a resource to determine when state changes occur.

resource: Functionality (e.g. voice-store-and-forward) that can be assigned to call. Resources are **shared** when functionality is selectively assigned to a call (usually via a PEB time slot) and may be shared among multiple calls. Resources are **dedicated** when functionality is fixed to the one call.

RFU: Reserved for future use.

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

route: Assign a resource to a time slot.

routing functions: For SCbus, functions that assign analog and digital channels to specific SCbus time slots; these SCbus time slots can then be connected to transmit or listen to other SCbus time slots. For PEB, functions that change the routing of channels to the time slots on the PCM Expansion Bus (PEB).

sampling rate: Frequency with which a digitizer takes measurements of the analog voice signal.

SCbus: Signal Computing Bus. Third generation TDM (Time Division Multiplexed) resource sharing bus that allows information to be transmitted and received among resources over multiple data lines.

SCSA: See Signal Computing System Architecture.

Signal Computer System Architecture: SCSA. A Dialogic standard open development platform. An open hardware and software standard that incorporates virtually every other standard in PC-based switching. All signaling is out of band. In addition, SCSA offers time slot bundling and allows for scalability.

silence threshold: The level that sets whether incoming data to the Voice board is recognized as silence or non-silence.

Special Information Tones: SIT. (1) Standard Information Tones. Tones sent out by a central office to indicate that the dialed call has been answered by the distant phone. (2) Special Information Tone. Detection of a SIT sequence indicates an operator intercept or other problem in completing the call.

speed and volume control: Voice software that contains functions and data structures to control the speed and volume of play on a channel. The end user controls the speed or volume of a message by entering a DTMF tone.

speed and volume modification table: Each channel on a voice board has a table with twenty entries that allow for a maximum of ten increases and decreases in speed or volume, and one "origin" entry that represents regular speed or volume.

SpringBoard: A Dialogic expansion board using digital signal processing to emulate the functions of other products. SpringBoard is a development platform for Dialogic products such as the D/120 and D/121.

SpringBoard functions: Functions used on SpringBoard devices only.

SpringWare: Software algorithms build into the downloadable firmware that provides the voice processing features available on all Dialogic voice boards.

SRL: See *Standard Runtime Library*.

Standard Attribute functions: Class of functions that take one input parameter (a valid Dialogic device handle) and return generic information about the device. For instance, Standard Attribute functions return IRQ and error information for all device types. Standard Attribute function names are case-sensitive and must be in capital letters. Standard Attribute functions for all Dialogic devices are contained in the Dialogic SRL. See *Standard Runtime Library*.

Standard Runtime Library: A Dialogic software resource containing Event-Management and Standard Attribute functions and data structures used by all Dialogic devices, but which return data unique to the device. See the *Standard Runtime Library Programmer's Guide for Windows NT*.

string: An array of ASCII characters.

subdevice: Any device that is a direct child of another device. Since "subdevice" describes a relationship between devices, a subdevice can be a device that is a direct child of another subdevice, as a channel is a child of a board.

synchronous function: Blocks program execution until a value is returned by the device. Also called a blocking function. See asynchronous function.

System Release Development Package: The software and user documentation provided by Dialogic that is required to develop applications.

termination condition: An event or condition which, when present, causes a process to stop.

termination event: An event that is generated when an asynchronous function terminates. See *asynchronous function*.

time slot: In a digital telephony environment, a normally continuous and individual communication (for example, someone speaking on a telephone) is (1) digitized, (2) broken up into pieces consisting of a fixed number of bits, (3) combined with pieces of other individual communications in a regularly repeating, timed sequence (multiplexed), and (4) transmitted serially over a single telephone line. The process happens at such a fast rate that, once the pieces are sorted out and put back together again at the receiving end, the speech is normal and continuous. Each individual pieced-together communication is called a time slot.

time slot assignment: The ability to route the digital information contained in a time slot to a specific analog or digital channel on an expansion board. See *device channe*l.

Universal Dialogic Diagnostic program: Software diagnostic routines for testing board-level functions of Dialogic hardware.

voice processing: Science of converting human voice into data that can be reconstructed and played back at a later time. Dialogic equipment can place 2-30 ports in one PC slot. They also use common API's for scalability and the SCbus to connect to a broad range of technologies.

Voice System: A combination of expansion boards and software that let you develop and run high-density voice processing applications.

Index

A

B

C

D

E

F

G

P

R

S

T

U

V

Dialogic Sales Offices

North American Sales
1-800-755-4444

Corporate Headquarters
1515 Route 10
Parsippany, NJ 07054
1-201-993-3030
fax: 1-201-993-3093

Northeastern US
70 Walnut Street
Wellesley, MA 02181

Southeastern US
1040 Crown Pointe Pkwy.
Suite 360
Atlanta, GA 30338

Central US
3307 Northland Drive
Suite 270
Austin, TX 78731

Western US
1314 Chesapeake Terrace
Sunnyvale, CA 94089

Northwestern US
19125 Northcreek Parkway #120
Bothell, WA 98011

Israel
Dialogic Israel Ltd.
P.O. Box 58, Building 2
Migdal Tefen Israel 24959

Latin America and the Caribbean
Roque Saenz Pena
730 Tercer Piso
Oficina 34 y 37
1035 Buenos Aires, Argentina
54-1-328-1531
fax: 54-1-328-5425

Europe, Middle East, and Africa
Dialogic Telecom Europe N.V.-S.A.
Belgicastraat 4
Keibergpark 3
B-1930 Zaventem
Belgium
32-2-725-0890
fax: 32-2-725-4275

Canada
Dialogic Corporation
1033 Oak Meadow Road
Oakville, Ontario
L6M 1J6
Canada

United Kingdom and Scandinavian Countries
Dialogic Telecom U.K. Ltd.
Dialogic House
Dairy Walk
Hartley Wintney
Hampshire
RG27 8XX
United Kingdom
44-1252-844000
fax: 44-252-844525

Japan
Dialogic Systems K.K.
Suntowers Center Building 18F
2-11-22 Sangenjaya
Setagayaku, Tokyo 154
Japan
81-3-5430-3252
fax: 81-3-5430-3373

Pacific
Dialogic (N.Z.) Ltd.
Level 6, Tower 2
Shortland Towers
55-63 Shortland Street
Auckland
New Zealand
64 9 366-1133
fax: 64 9 302-1793

Southeast Asia
Dialogic SEA Ltd.
17-08 Gateway West Building
150 Beach Road
Singapore 0718
65-298-8208
fax: 65-298-1820

France
Dialogic Telecom France
42 Avenue Montaigne
75008 Paris
France
33-1-53-67-52-80
fax: 33-1-53-67-82-79

Germany
Dialogic Telecom Deutschland GMBHIG
Ridlerstrabe 11
D80339 Munchen
Germany
49-89-5020-0914
fax: 49-89-5024-540

NOTES

NOTES

NOTES