Digital Network Interface Programmer's Guide

for Windows NT

COPYRIGHT NOTICE

Publication Date: April, 1997

Dialogic Corporation
1515 Route Ten
Parsippany, N.J. 07054

Table of Contents

List of Tables

List of Figures

1. Digital Network Interface General Description

Digital Network Interface Overview

The Network library of C functions allows a programmer to design application programs that run on a host PC and work with one or more Dialogic Digital Network Interface boards. The functions provided control the Digital Network Interface device on the PCM Expansion Bus (PEB) or SCbus and the Network external interface to network circuits that meet either the T-1 or E-1 telephony standard.

NOTE: The DTI/211 and DTI/212 boards operate on the PCM Expansion Bus (PEB). The DTI/2xx boards are not SCbus compatible. The Network library of C functions also supports DTI/101 boards. See Section 1.2. for details about the backward compatibility provided by this development package.

The System Release Development Package includes Standard Runtime Library (SRL) functions used in Network Windows NT applications to perform such tasks as event management. SRL functions for Network applications are documented in *Appendix A* of this guide. For a complete explanation of the SRL, see the *Standard Runtime Library Programmer's Guide for Windows NT*, included in the *Voice Software Reference for Windows NT*.

1.1. Digital Network InterfaceTypical Applications

The type of applications supported by your software is dependent on the physical configuration of the host PC system. For instance, a program that will run with a DTI/2xx and other Dialogic devices arranged in **terminate configuration** allows your system to act as a standalone voice processing node. Applications for this configuration include:

- Central-office-based voice mail
- Cellular messaging
- Audiotex

- Service bureaus

A program designed to run with multiple DTI/2xx devices arranged in **drop-and-insert configuration** allows individual channels to terminate at a voice processing device, pass transparently to the network, or both. Applications for this configuration include all the terminate applications plus:

- Operator services such as billing automation, directory assistance, and intercept treatments
- Telemarketing
- Agent automation
- Direct dial-in (DDI) service

Refer to the *SCbus Configuration Planning Guide* for typical applications using the D/240SC-T1 or D/300SC-E1 device. To install and configure your hardware, refer to the appropriate hardware installation card (see *Appendix B*).

1.2. Digital Network Interface Compatibility

This section describes compatibility of the Network software for Windows NT with Dialogic hardware and with existing applications based on the Dialogic Network driver.

NOTE: The DTI/2xx boards are not SCbus-compatible. The SCbus routing functions introduced in this guide do not support the DTI/2xx boards.

The System Release Development Package for UNIX supports all Digital Network Interface hardware. Some functions in the Network function library of C functions may operate differently or not at all on a given Digital Network Interface board type due to differences in the board's usage. This section explains these differences in functionality.

- **dt_dial( )** is not supported by the DTI/211 board or the DTI/212 board. It is supported by the D/240SC-T1 board and the D/300SC-E1 board.

  NOTE: To perform dialing you can instead use a Windows NT Voice library function supported by your D/xxx voice boards. The function name is **dx_dial( )**. If you have a different version, see your Voice Software Reference for UNIX.

- **dt_open( )** opens time slots from 1 to 24 in T-1 applications (DTI/211 and D/240SC-T1 boards) or 1 to 30 in E-1 applications (DTI/212 and D/300SC-E1 boards).
- **dt_route( )** is not supported by DTI/2xx hardware.

  NOTE: To reroute time slots on the PEB, you can instead use a Windows NT Voice library function supported by your D/xxx voice boards. The function name is **dx_route( )**.
- The following functions are new to the network library (*libdti.lib* and *libdtint.lib*) and provide support for routing time slots on the SCbus:
  - **dt_getctinfo( )** is used to return device information for an on-board digital network interface device time slot.
  - **dt_getxmitslot( )** returns the SCbus time slot number connected to the transmit of a digital network time slot.
  - **dt_listen( )** is used to connect the receive of a digital network time slot to an SCbus time slot.
  - **dt_unlisten( )** is used to disconnect the receive of a digital network interface device time slot from the SCbus.
- **dt_setalrm( )** DTA_DROP parameter is not supported by DTI/212 or D/300SC-E1 devices. For these devices, use only DTA_NONE or DTA_TERM.
- **dt_setevtmsk( )** and **dt_getevtmsk( )** functions include the DTG_PDIGEVT parameter, which is not supported by DTI/2xx hardware. These functions also include additional parameters and masks for E-1 alarm handling (DTI/212 and D/300SC-E1 only) and for T-1 alarm handling (DTI/211 and D240/SC-T1 only). See the function descriptions in *Chapter 3. Digital Network Interface* for more information.
- **dt_setsigmod( )** transparent signaling mode is not supported by DTI/212 boards or in SCbus configurations.
- **dt_xmitalrm( )** function uses additional parameters for E-1 alarm transmission (DTI/212 and D/300SC-E1 only).

- **dt_xmittone( )** one-milliwatt tone generation is not supported by DTI/212 or D/300SC-E1 hardware.

The Network device driver also supports the PEB-based MSI and DMX boards. Refer to the MSI and DMX references listed in *Appendix B* of this guide for more information about functions supported on these boards. The MSI and DMX boards are not SCbus-compatible.

1.3. Digital Network Interface SCbus Overview

SCbus is the TDM (Time Division Multiplexed) bus connecting SCSA (Signal Computing System Architecture) 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 which 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, refer to the *SCbus Configuration and Planning Guide* and the *Voice Software Reference for Windows NT*.

Digital Network Interface Products Covered by this Guide

This guide covers the software for the products listed in the table below.

Product Name:	Description
DTI/211	The Dialogic digital telephony interface board for T-1 telephony standards that connects T-1 networks to compatible voice processing boards.
DTI/212	The Dialogic digital telephony interface board for E-1 telephony standards that connects E-1 networks to compatible voice processing boards.
D/240SC-T1	The Dialogic single-slot, high-density voice processing board with a T-1 network interface module.
D/300SC-E1	The Dialogic single-slot, high-density voice

Product Name:	Description
	processing board with an E-1 network interface module. .Cyclic Redundancy Checking (CRC) generation does not work, even when the appropriate download parameter (000[cjs2]Ff) is turned on

In the context of this guide, "Digital Network Interface" is used to refer to the DTI/211 board, the DTI/212 board, the D/240SC-T1, and the D/300SC-E1 board unless otherwise noted.

The DTI/211 and DTI/212 boards operate on the PCM Expansion Bus (PEB). The DTI/2xx boards are not SCbus compatible.

For information on the DTI/240SC and DTI/300SC boards, see the Primary Rate Software Reference for Windows NT.

E-1 is used to refer to the 2.048 Mbps Digital Service with Channel Associated Signaling (CAS) see section *2.2.3. E-1 Signaling*. This service is available in Europe and some parts of Asia.

Digital Network Interface Product Terminology

The following product naming conventions are used throughout this guide:

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 32-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 or **Spancard** 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**.

DMX refers to all Dialogic Digital Matrix Switch boards. These boards provide cross-PEB time slot switching capability for up to four PEB systems.

DTI/xxx refers to any of Dialogic's digital telephony interface expansion boards for the AT-bus architecture. These boards include: **DTI/101**, **DTI/211**, and **DTI/212** boards.

MSI refers to all Dialogic Modular Station Interface boards. These boards connect PEB (PCM Expansion Bus) time slots to analog station devices.

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.

Spancard same as **DIALOG/HD**.

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.

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

For additional information on these products, refer to the Dialogic publications listed in *Appendix B*.

How to Use This Digital Network Interface Guide

This guide is written for users who have purchased a Dialogic DTI/211, DTI/212, D/240SC-T1 or D/300SC-E1 board and the System Release Development Package for installation on a Windows NT PC.

The following steps explain the order in which a Digital Network Interface board and related Dialogic software products for Windows NT should be installed, checked, and programmed.

1. Prepare the Digital Network Interface board for installation using the appropriate hardware installation card (see *Appendix B*).
2. Install the System Release Development Package for Windows NT by following the procedure described in the *System Release Software Installation Reference for Windows NT*.
3. Install the Digital Network Interface board(s) in your PC following the procedures in the appropriate hardware installation card (see *Appendix B*).
4. Refer to this *Digital Network Interface Programmer's Guide for Windows NT* and the *Standard Runtime Library Programmer's Guide for Windows NT* (included in the *Voice Software Reference for Windows NT*) to develop application programs.

To use software for other Dialogic devices, refer to the appropriate software reference for specific instructions (see *Appendix B*).

Organization of This Digital Network Interface Guide

This *Digital Network Interface Programmer's Guide for Windows NT* contains an overview of the Dialogic digital telephony interface and a Network Windows NT library C function reference. It is organized as follows:

Chapter *1. Digital Network Interface General Description* provides a brief description of the Network library of C functions, typical applications using the Digital Network Interface products, and an overview of the SCbus.

Chapter *2. Digital Network Interface Telephony* presents an overview of Dialogic digital telephony interface (DTI) hardware implementation in relation to basic T-1 and E-1 telephony practices.

Chapter *3. Digital Network Interface* provides descriptions of the Network Windows NT library functions that control the Digital Network Interface hardware.

Chapter *4. Digital Network Interface Function Reference* provides guidelines for the design of Dialogic Digital Network Interface applications as well as a detailed alphabetical reference to the Dialogic Network Windows NT library functions, including programming examples for each function.

Chapter *5. Digital Network Interface Application Guidelines* provides advice and suggestions to guide programmers in designing and coding a Dialogic Digital Network Interface application for Windows NT.

Appendix A lists returns and defines associated with the Dialogic Standard Runtime Library (SRL) that are unique to Digital Network Interface devices.

Appendix B lists related publications. This includes a list of Dialogic guides, Dialogic application notes, and non-Dialogic references.

2. Digital Network Interface Telephony

Overview

This chapter provides a brief overview of T-1 and E-1 concepts and a description of how Dialogic hardware works in T-1 and E-1 environments.

It is beyond the scope of this guide to explain all the details of T-1 and E-1 digital telephony. For more detailed information, refer to the related publications listed in *Appendix B*.

2.1. T-1 Digital Network Interface Telephony

A T-1 circuit is used to transfer digital information in a two-way, full duplex connection at a speed of 1.544 megabits per second (Mbps). In a T-1 environment, this rate is known as *digital signal level 1* or DS-1. A T-1 circuit contains 24 voice channels, each operating at a rate of 64,000 bits per second (bps), a rate known as *digital signal level 0* or DS-0. The formula used to calculate the DS-1 rate of 1.544 Mbps includes an extra 8,000 bits that are not part of the voice data but used to synchronize the data received and transmitted on the T-1 circuit.

```
   64,000 bps     (Voice Channel Rate, DS-0)
     x 24         (Number of Voice Channels)

1,536,000 bps
  + 8,000         (Controlling Bits)

1,544,000         (T-1 Circuit Rate, DS-1)
```

The T-1 compatible DTI/211 board and the D/240SC-T1 board demultiplex the 24 voice channels on a T-1 circuit and pass them on to associated hardware (such as a voice board or other resource sharing module).

2.1.1. T-1 Frame Format

Digital data on a T-1 line is organized into D4 frames. A D4 frame consists of a single 8-bit sample from each of the 24 voice channels and one framing bit, for a total of 193 bits. Each 8-bit sample occupies what is known as a time slot within the frame. Figure 1 shows one D4 frame.

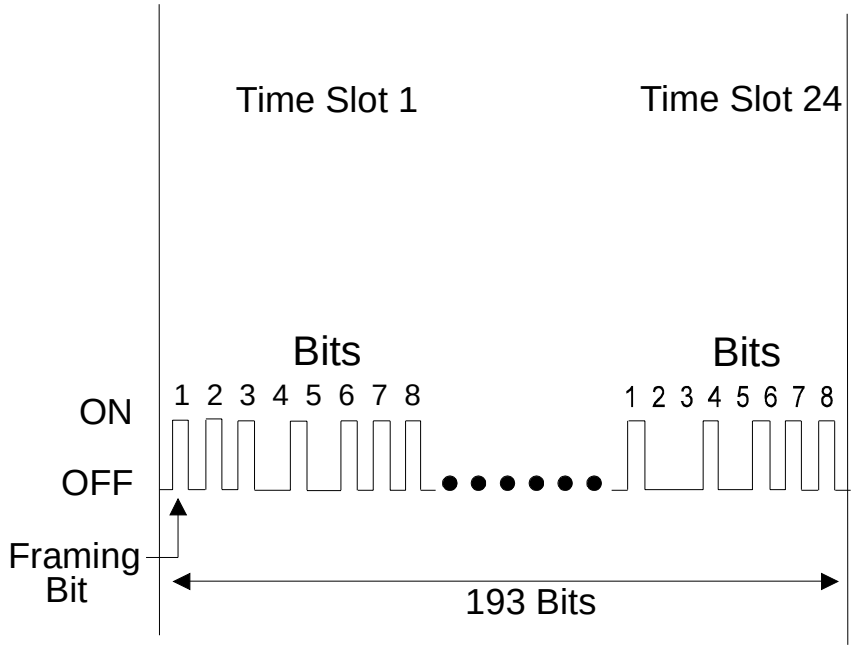

Figure 1. D4 Frame Format

The term *time slot* is derived from the method that is used to multiplex the 24 voice channels in a D4 frame. The channels are byte-interleaved in a frame. That is, each byte is a sample from a different voice channel and occurs in a fixed pattern within the frame (voice channel one in time slot one, voice channel two in time slot two, etc.). All D4 frames have the same pattern. This technique of interleaving is called time division multiplexing.

Twelve D4 frames make up what is known as a D4 superframe. *Figure* 2 shows a single D4 superframe, indicating the framing bit values of the individual D4 frames. The framing bits are used for frame synchronization, which is described in more detail in *Section 2.1.2. T-1 Synchronization*

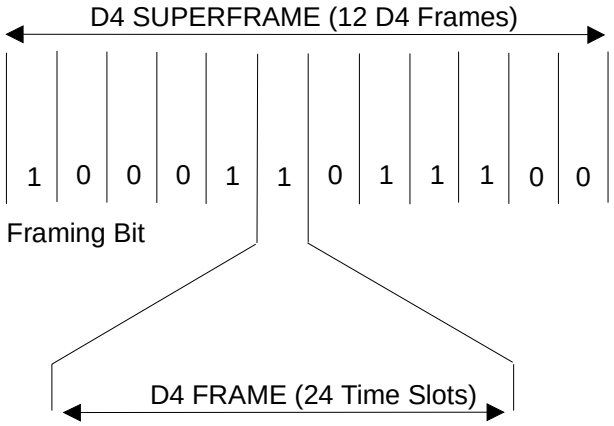

Figure 2. D4 Superframe Format

2.1.2. T-1 Synchronization

To identify DS-0 voice channels for the receiver, the data being transferred must be synchronized. This capability is built into the D4 frame and superframe formats for T-1 systems. Each D4 frame in a superframe begins with a *framing bit*. The 12 framing bits in a D4 superframe are arranged in a predefined pattern: 100011011100. By searching for this pattern, the T-1 compatible DTI/211 or D240SC-T1 hardware can determine the beginning and end of every D4 superframe, D4 frame, and time slot. When this pattern cannot be found, the resulting error is known as Receive Loss of Synchronization (RLOS). See *Section 2.3.7. Loss of Synchronization Alarm Handling* for information on T-1 alarm handling.

2.1.3. T-1 Signaling

T-1 signaling information (on-hook and off-hook states) must be carried on a T-1 line. Signaling is accomplished using two bits called the A-bit and the B-bit. Each time slot in the sixth frame of the D4 superframe has the least significant bit replaced with signaling information. These are the A-bits. Similarly, each time slot in the twelfth frame of the D4 superframe has the least significant bit replaced with signaling information. These are the B-bits. This strategy of replacing the least significant bit with signaling information is called robbed-bit signaling.

For example, in E&M protocol the signaling bits indicate whether the sending party's line is on-hook or off-hook. When the signaling bits are 0s, the line is on-hook, and when the signaling bits are 1s, the line is off-hook.

NOTE: Some T-1 services reverse these values or use them in different patterns or protocols. Check with your T-1 supplier to verify the A-bit and B-bit values for your T-1 service.

2.2. E-1 Digital Network Interface Telephony

An E-1 circuit is a digital two-way connection operating at a speed of 2.048 Mbps. This rate is achieved by combining 32 time slots operating at a rate of 64 Kbps.

```
    64,000 bps     (Individual Voice Channel Rate)
     x  32         (Number of Channels or Time Slots)

2,048,000          (E-1 Circuit Rate)
```

These 32 time slots include 30 time slots available for up to 30 voice channels, one time slot dedicated to carrying frame synchronization information (time slot 0), and one time slot dedicated to carrying signaling information (time slot 16). An E-1 compatible DTI/212 board or D/300SC-E1 board demultiplexes the 30 voice channels and passes them on to E-1 compatible resource modules.

NOTE: E-1 is used to refer to the 2.048 Mbps Digital Service with Channel Associated Signaling (CAS). This service is available in Europe and some parts of Asia.

2.2.1. E-1 Frame Format

On an E-1 circuit, data is organized into frames on a byte-interleaved basis. Data is taken from each voice channel a byte at a time. The resulting E-1 frame contains 32 time slots: one to carry frame synchronization information, one to carry signaling information, and 30 to carry voice channel data. Each time slot contains 8 bits, for a total of 256 bits per frame.*Figure* 3 illustrates the structure of an E-1 frame.

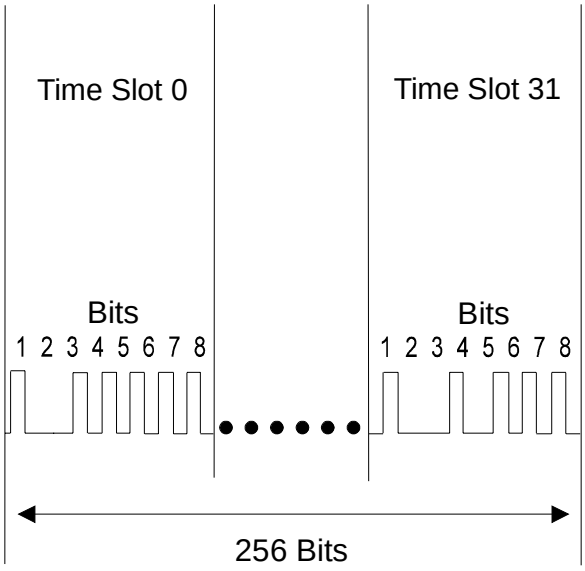

Figure 3. E-1 Frame Format

E-1 frame format numbers time slots from 0 to 31. Dialogic DTI/212, MSI, and DMX products number voice channels from 1 to 30. *Table 1* shows how these channel numbers map to E-1 time slot numbers for a DTI/212 device.

Table 1. E-1 and DTI/212 Time Slot Numbering

E-1 time slot	DTI/212 voice channel	E-1 time slot	DTI/212 voice channel	E-1 time slot	DTI/212 voice channel
00	N/A	11	11	22	21
01	01	12	12	23	22
02	02	13	13	24	23
03	03	14	14	25	24
04	04	15	15	26	25
05	05	16	N/A	27	26
06	06	17	16	28	27
07	07	18	17	29	28
08	08	19	18	30	29
09	09	20	19	31	30
10	10	21	20		

NOTE: Voice channels are not mapped to E-1 time slots 0 and 16. Time slot 0 contains the frame's synchronization information. See*2.2.2. E-1 Synchronization* for more information on E-1 synchronization. Time slot 16 contains the frame's signaling information. See *Section 2.2.3. E-1 Signaling* for more information on E-1 signaling.

E-1 frames 0 through 15 are combined into one *multiframe*. *Figure* 4 illustrates the structure of an E-1 multiframe.

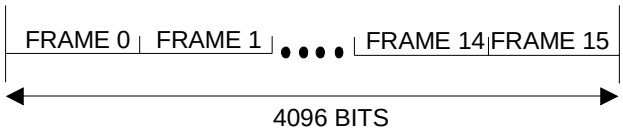

Figure 4. E-1 Multiframe Format

2.2.2. E-1 Synchronization

Time slot 0 of each frame (frames 0 through 15 of a multiframe) carries the information needed to identify voice channels for the receiver on E-1 systems. The pattern carried by time slot 0 alternates between two patterns: the first is a 7-bit pattern (0011011) in bit positions 6 through 0 and the second is a pattern of national and international bits with a single 1-bit in bit position 6. *Figure* 5 shows the alternating bit patterns in odd and even frames.

TIME SLOT 0 OF EACH FRAME

	MSB							LSB
BIT POSITION	7	6	5	4	3	2	1	0
ODD FRAME	I	0	0	1	1	0	1	1
EVEN FRAME	I	1	—	N	N	N	N	N

I - INTERNATIONAL BIT
N - NATIONAL BIT
MSB - MOST SIGNIFICANT BIT
LSB - LEAST SIGNIFICANT BIT

Figure 5. Individual Frame Synchronization

See*2.2.4. E-1 National and International Bits* for an explanation of the E-1 national and international bits pictured in *Figure* 5.

Frame 0 (the first frame within an E-1 multiframe) contains additional synchronization information to identify the beginning of a multiframe. The beginning is identified by a pattern of four zeros in bit positions 7 through 4 of time slot 16, frame 0. illustrates the bit pattern found in time slot 16 of frame 0.

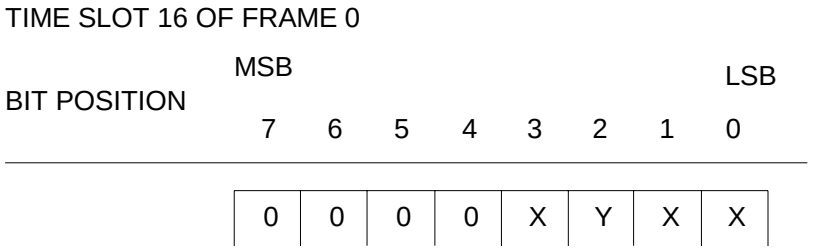

Figure 6. Multiframe Synchronization

If these frame or multiframe bit patterns cannot be found, the resulting error is known as a Frame Sync Error (FSERR) or Multiframe Sync Error (MFSERR). If either an FSERR or MFSERR error is detected, a remote alarm or a distant multiframe alarm is sent to the remote end. The condition exists until synchronization is recovered. See *E-1 Alarm Handling* for information on E-1 alarm handling.

2.2.3. E-1 Signaling

The Conference of European Postal and Telecommunications administrations (CEPT) defines how bits of a PCM carrier system in E-1 areas will be used and in what sequence. E-1 circuits use the Channel Associated Signaling (CAS) protocol. Frames using CAS share time slot 16, which carries signaling information for two time slots or voice channels at a time.

Time slot 16 contains two groups of four bits, known as nibbles, that are designated the upper nibble and the lower nibble. Two channels send their signaling bits in each frame - one using the upper nibble, the other using the lower nibble. As explained in *Section 2.2.1. E-1 Frame Format* on E-1 frame format, it takes 15 frames to carry signaling information for each of the 30 voice channels.

Time slot 16 of frame 0 carries a special pattern. The upper nibble carries a pattern of four 0s, which identifies the frame as frame 0 of an E-1 multiframe.

The lower nibble of time slot 16 in frame 0 carries a pattern of extra bits and an alarm bit. The X bits pictured in *Figure* 7 are the extra bits used for multiframe synchronization (see *2.2.2. E-1 Synchronization*. The Y bit pictured in *Figure* 7 is the distant multiframe alarm bit (see *E-1 Alarm Handling*.

Time slot 16 of frame 1 in an E-1 multiframe carries signaling information for the first and sixteenth channels. Time slot 16 of frame 2 in an E-1 multiframe carries signaling information for the second and the seventeenth channels. This continues until frame 15 which carries signaling information for the fifteenth and thirtieth channels.

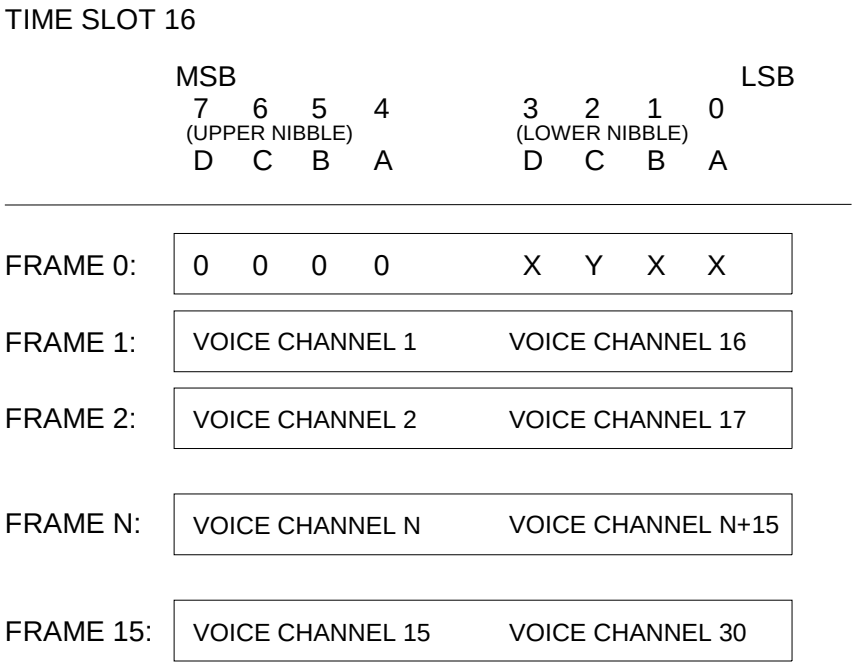

Figure 7. Channel Associated Signaling (CAS) Protocol

Caution
Do not set signaling bits ABCD to 0000. As explained in *Section 2.2.2. E-1 Synchronization* on E-1 synchronization, this setting is used to identify frame 0 of an E-1 multiframe.

2.2.4. E-1 National and International Bits

National and international bits are set in time slot 0. The most significant bit (bit position 7) in time slot 0 of each frame contains the international bit. The national bits occupy bit positions 0 through 4 of time slot 0 of every second frame. *Figure* 8 shows national and international bit settings.

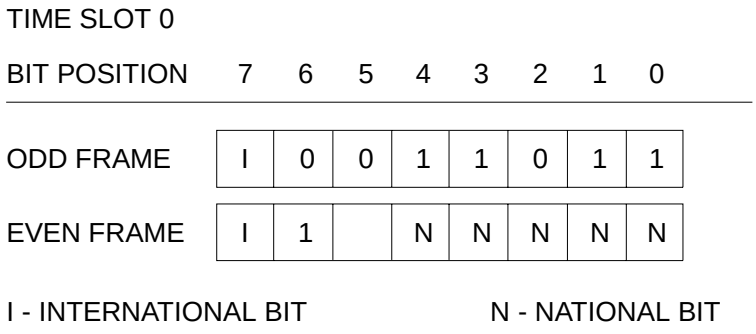

Figure 8. E-1 National and International Bits

2.3. Digital Network Interface Hardware Implementation

The following sections describe features of the Dialogic Digital Network Interface hardware implementation that are important to note for purposes of application development.

2.3.1. Intelligent Network Interfaces

The Intelligent Network Interfaces products support the following features:

- Clear channel TS16
- Call progress features
- Tone features
- Signaling features

2.3.2. Clear Channel TS16

Added a download parameter that allows the use of time slot 16 for data on E-1 interface boards.

2.3.3. Modifying Network Parameters

When modifying the default network parameters, ensure that you perform the following:

- If you have Country-Specific Parameter software installed, modify your parameter file which is located in the <*install drive*>:\<*install directory*>\dialogic\data. The file name is in the form of xx_240.prm or xx_300.prm, where xx is the country code. For example, the parameter file for Australia is au_240.prm or au_300.prm.
- If you do not have Country-Specific Parameter software installed, edit the SPANDTI.PRM file which is located in the <*install drive*>:\<*install directory*>\dialogic\data directory.

2.3.4. Signaling Features

The Intelligent Network Interface boards in this release support a combination of the signaling features listed below:

- Channel Associated Signaling (CAS)
- Robbed Bit or Loop Signaling
- Pulse Dialing
- Wink or Flash

For more information on voice, call progress, and tone features, refer to the *Voice Software Reference for Windows NT*. For signaling features, refer to the *Digital Network Interface Software Reference for Windows NT.*

The features for each intelligent network interface product are listed in the following features matrix:

Product	Call Progress Features	Tone Features	Signaling Features
DTI/240SC			✓
DTI/241SC	✓	✓	✓
DTI/300SC			✓
DTI/301SC	✓	✓	✓
LSI/81SC	✓	✓	✓
LSI/161SC	✓	✓	✓

NOTES: 1. Network signaling event detection is not passed through the SCbus. In SCbus applications, network signaling events cannot be detected by voice resource devices connected via the SCbus. For example, a voice channel on a board without an attached analog network interface cannot automatically terminate a play as a result of a drop in loop current. The analog network interface board can generate an event indicating that a loop current drop occurred; the application must then explicitly terminate the play.

2. The DTI/301SC boards are available in a 75 Ohm or 120 Ohm version.

2.3.5. PEB Device Channels and Routing Functions

A time slot can be routed to another network device or be dropped to a directly connected PEB-compatible resource board. In turn, information originating at the resource board can be inserted into the transmit bitstream. All routing is done by the Voice board or other resource module. The DTI/2xx board has no control over the routing of time slots.

2.3.6. SCbus Routing

Data is transmitted over the SCbus in 1024 time slots. At system initiation and download, the number of devices (analog interface, voice, digital network interface, facsimile, etc.) on each board and the number of SCbus time slots required to service these devices are determined. Only one digital network interface device time slot can transmit on a specific SCbus time slot at a time. To assure this, the transmit of all devices are assigned to a specific and unique SCbus time slot at system initialization. This transmit assignment cannot be changed by the application.

When both voice devices and telephone network digital interface devices (T-1/E-1) are on a single SCbus board, these resources may be treated as separate and independent devices.

2.3.7. Loss of Synchronization Alarm Handling

The most critical error condition that can occur on a T-1 or E-1 line is Receive Loss of Synchronization (RLOS). This section describes the alarm conditions and signals associated with Digital Network Interface alarm handling and how they are indicated on a Dialogic Digital Network Interface board.

T-1 Alarm Handling

For T-1 applications, the DTI/211 and D/240SC-T1 boards generate three alarm conditions to indicate RLOS:

- Red alarm
- Yellow alarm
- Blue alarm

A **red alarm condition** occurs when RLOS has existed for 2.5 seconds (default) on incoming data. This condition will exist until the synchronization has been recovered and remains recovered for 12 seconds (default).

A **yellow alarm** is sent by the receiving T-1 Digital Network Interface device to the transmitter device. The yellow alarm indicates to the transmitter device that a

red alarm condition exists at the receiver device. The yellow alarm is sent for as long as the red alarm condition exists at the receiver device.

NOTE: A yellow alarm is sent by the T-1 Digital Network Interface receiver device by inserting a zero in bit 2 of all time slots.

The **blue alarm** is a "keep alive" signal. When the T-1 Digital Network Interface device is used in a drop and insert configuration and it receives an RLOS for 2.5 seconds, a red alarm condition is entered on the T-1 Digital Network Interface side that received the RLOS. The configuration then transmits a blue alarm signal from the other Digital Network Interface connected via the PEB cable to its T-1 span. The blue alarm signal informs the receiving station that there is a problem on the line and allows the receiving station to continue to derive its transmit clock from the received signal.

NOTE: The blue alarm signal causes an RLOS on the T-1 Digital Network Interface device that receives the blue signal. A blue alarm consists of an unframed pattern of 1s.

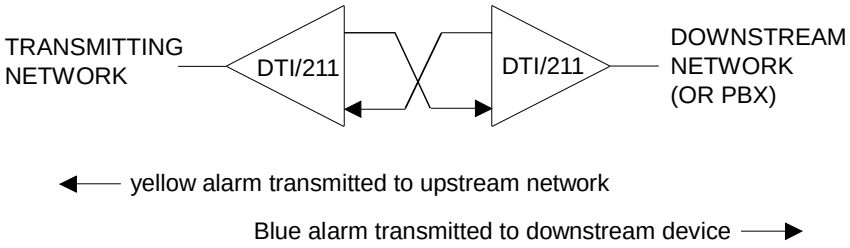

Figure 9. T-1 Alarm Conditions

E-1 Alarm Handling

For E-1 applications, the DTI/212 and D/300SC-E1 boards generate four alarm conditions to indicate loss of synchronization (FSERR or MFSERR):

- Remote alarm
- Unframed all 1s alarm
- Distant multiframe alarm
- Signaling all 1s alarm

A **remote alarm** is generated by the DTI/212 or D/300SC-E1 device to indicate it has detected a loss of frame synchronization on the receive line (FSERR condition). The remote alarm is transmitted to the E-1 network. A remote alarm is returned to the network by setting bit 3 of time slot 0 in non-alignment frames to 1. ("Non-alignment frames" are those frames not carrying the 7-bit frame-sync pattern 0011011 in time slot 0.)

If the DTI/212 or D/300SC-E1 device is in a drop-and-insert configuration, it also generates an **unframed all 1s alarm**. The unframed all 1s alarm is transmitted to the downstream device to indicate that the data it is receiving is unsynchronized at the frame level and is therefore unreliable. The downstream device must then transmit this alarm to the downstream network.

When the DTI/212 or D/300SC-E1 device detects a recovery of frame synchronization, it will stop transmitting the remote and unframed all 1s alarms.

A **distant multiframe alarm** is generated by the DTI/212 or D/300SC-E1 device to indicate it has detected a loss of multiframe synchronization on the receive line (MSFERR condition). The distant multiframe alarm is transmitted to the E-1 network. The Digital Network Interface device returns a distant multiframe alarm by setting the bit in position 2 of time slot 16 in frame 0 to 1.

If the DTI/212 device is in a drop-and-insert configuration, it also generates a **signaling all 1s alarm**. A signaling all 1s alarm is generated by inserting all 1s in time slot 16. The signaling all 1s alarm is transmitted to the downstream device to indicate that the data it is receiving is unsynchronized at the multiframe level and is therefore unreliable. The downstream device must then transmit this alarm to the downstream network.

When the DTI/212 or D/300SC-E1 device detects a recovery of multiframe synchronization, it will stop transmitting the distant multiframe and signaling all 1s alarms.

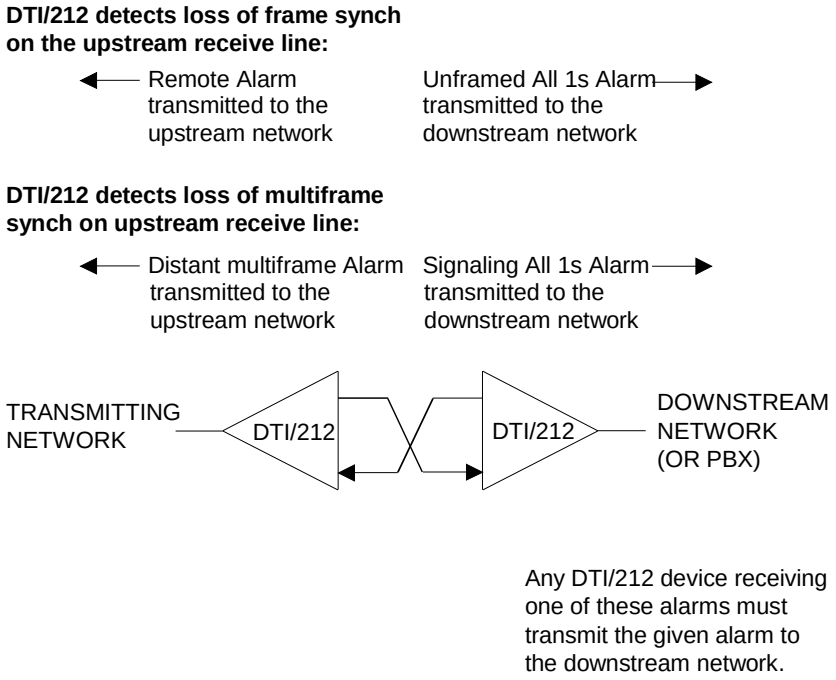

Figure 10. E-1 Loss of Synchronization Alarm Requirements

2.3.8. Digital Network Interface Hardware Alarm Indicators

The three LEDs on the rear bracket of the Digital Network Interface board indicate the state of the signal being received. All LED indicators will remain lit until the Digital Network Interface firmware is downloaded to the device.

Red LED: The red LED lights up whenever the Digital Network Interface device detects RLOS.

Yellow LED: A yellow LED lights up whenever the Digital Network Interface device receives an alarm indicating that a network span is receiving unsynchronized data from the Digital Network Interface board.

Green LED: A green LED is lit whenever the Digital Network Interface board is receiving a signal.

NOTES:

1. Red, yellow, and green LEDs will be lit when the system is powered up, regardless of whether or not a signal is being received.
2. No alarm handling is performed until Digital Network Interface boards are downloaded.
3. Once the firmware is downloaded, the default alarm handling mode for Digital Network Interface boards is terminate alarm handling.

3. Digital Network Interface

Function Overview

This chapter describes the Network Windows NT library functions that control the Digital Network Interface hardware.

3.1. Digital Network Interface Library Function Categories

NOTE: In the context of this guide, "Digital Network Interface" is used to refer to the DTI/211 board, the DTI/212 board, the D/240SC-T1 board and the D/300SC-E1 board unless otherwise noted.

The Network library functions provide the necessary building blocks to create voice applications using T-1 or E-1 lines. These functions can be divided into the following categories:

Alarm functions	Control T-1 or E-1 alarm handling
Compatibility functions	
Diagnostic functions	Test Digital Network Interface hardware
Extended Attribute functions	Retrieve device-specific attribute data
Parameter Request functions	Request device parameters
Parameter Setting functions	Set device parameters
Resource Management functions	Open and close Digital Network Interface devices
SCbus Routing functions	Generate communication between devices connected to SCbus time slots
Time Slot Audio functions	Generate audio signals on time slots
Time Slot Signaling functions	Alter signaling portion of time slot

NOTE: Many Network Windows NT library functions can operate in either *synchronous* mode or *asynchronous* mode. Synchronous functions do not

return control to the calling process until the function call is completed. To operate a function in *asynchronous* mode, your application must include an event handler to trap and process the completion event.

Each category and its functions are briefly described in the following sections.

3.1.1. Alarm Functions

dt_setalrm( )	• set alarm handling mode
dt_xmitalrm( )	• start/stop alarm transmission

The Alarm functions allow your application to control the way T-1 or E-1 alarms are handled. The **dt_setalrm( )** function sets the alarm-handling mode. The **dt_xmitalrm( )** function starts and stops the transmission of alarms.

For a detailed discussion of T-1 and E-1 alarm handling, refer to *Chapter 2. Digital Network Interface Telephony* .

3.1.2. Compatibility Functions

dt_libinit( )	• initializes the Network Library DLL
dt_GetD11Version( )	• returns the Network DLL Version Number

The **xx_libinit( )** function calls the **LoadLibrary( )** function to load a specific Dialogic technology DLL. If the DLL does not exist, all its functions are set up as default Not Implemented Functions. If the DLL does exist, the **xx_libinit( )** function performs a series of **GetProcAddress( )** function calls that. set up the address pointers for the functions.

3.1.3. Diagnostic Functions

dt_rundiag( )	• run diagnostics on Network firmware
dt_tstcom( )	• test board Interface communications
dt_tstdat( )	• run data test on board device

The Diagnostic functions check the Network firmware and hardware. The **dt_rundiag( )** function runs diagnostics on the Network firmware and the other two functions test the hardware. The **dt_tstcom( )** function tests communication between the PC and the Digital Network Interface device. The **dt_tstdat( )** function tests the reliability of data transfer between the PC and the Digital Network Interface device.

3.1.4. Extended Attribute Functions

ATDT_BDMODE( )	• board signaling mode (all time slots)
ATDT_BDSGBIT( )	• board signaling bits (all time slots)
ATDT_DNLDVER( )	• downloaded Network firmware version
ATDT_IDLEST( )	• time slot idling state
ATDT_ROMVER( )	• EPROM version
ATDT_STATUS( )	• time slot status
ATDT_TSMODE( )	• get time slot signaling mode
ATDT_TSSGBIT( )	• get time slot signaling bits

Standard Attribute functions, which are contained in the Dialogic Standard Runtime Library (SRL, see *Appendix A*), provide generic information about a device, such as its name or the status of the last function call of the device. **Extended Attribute** functions return device specific information. The Network Windows NT Library Extended Attribute functions return information about Digital Network Interface logical board and time slot devices.

Extended Attribute function error handling is similar to that of other Network library functions. Most Extended Attribute functions return AT_FAILURE on error. One Extended Attribute function, **ATDT_BDSGBIT( )**, returns the value AT_FAILUREP on error. Refer to *Section 3.2. Digital Network Interface Error Handling* for information about retrieving errors.

3.1.5. Parameter Request Functions

dt_getparm( )	• get device parameter
dt_getevt ( )	• blocks and returns control after event
dt_getevtmsk( )	• get device event bitmask

Parameter Request functions are used to check the status of Network parameter and event mask settings.

3.1.6. Parameter Setting Functions

dt_setparm( )	• change device parameter
dt_setevtmsk( )	• change device event mask

The Parameter Setting functions set Network device parameters and masks used for event management.

- When the application is first invoked after a Dialogic Service Startup, if the application enables signaling transition notification via the dt_setevtmsk( ) function, a DTG_SIGEVT is posted automatically. The state of the receiving signaling is typically DTMM_AOFF and DTMM_BOFF, if enabled, as if the transition had just occurred. Subsequent generation of these events is only on signaling state change.

3.1.7. Resource Management Functions

dt_open( )	• open board or time slot device
dt_close( )	• close board or time slot device

Resource Management functions open and close devices. Before you can perform an operation on a Dialogic device, the device must be opened. The **dt_open( )** function returns a unique device handle. All subsequent operations on the device must use this handle.

NOTES: **1.** A device handle is NOT the same as a Windows NT system file handle.

2. Opening or closing a Digital Network Interface device does not affect other processes using the device but a command can only be issued while the device is idle. (See *Chapter 5. Digital Network* Interface Application Guidelines, for more information on opening and using DTI/2xx devices.)

3. The value returned by **dt_open( )** for a Digital Network Interface logical board is referred to as a *DTI/2xx logical board device handle* in this guide. The value returned by **dt_open( )** for a DTI/2xx logical time slot device is referred to as a *DTI/2xx logical time slot device handle*.

3.1.8. SCbus Routing Functions

dt_getctinfo( )	• get information about the digital network interface device time slot connected to theSCbus
dt_getxmitslot( )	• returns SCbus time slot connected to the digitalnetwork interface device time slot
dt_listen( )	• connects the receive of a digital network nterface device time slot to an SCbus time slot
dt_unlisten( )	• disconnects the receive of a digital network interface device time slot from an SCbus time slot

SCbus routing functions enable the application to make or break a connection between voice, telephone network interface and other resource channels connected via SCbus time slots.

3.1.9. Time Slot Audio Functions

dt_setidle( )	• enable/disable time slot idle state
dt_xmittone( )	• enable/disable transmission of test tone

A Time Slot Audio function affects only the transmitted audio portion of a time slot. It replaces the normal voice data on the audio portion of a time slot with other data. The **dt_setidle( )** function transmits an idle pattern (digital equivalent of silence) on the selected digital network interface time slot. The specific idle pattern transmitted can be specified via the download configuration file or by using the **dt_setparm( )** function. The **dt_xmittone( )** function transmits a fixed-frequency test tone to the PEB on the selected DTI/211 time slot.

NOTE: The **dt_xmittone( )** function is not supported by and will not operate on DTI/212 or D/300SC-E1 devices.

3.1.10. Time Slot Signaling Functions

dt_dial( )	• dial a pulse digit string
dt_mtfcn( )	• initiate or stop a multitasking function
dt_setsigmod( )	• change time slot transmit signaling mode
dt_settssig( )	• change time slot signaling bits
dt_settssigsim( )	• clear and set signaling bits simultaneously
dt_xmitwink( )	• transmit wink signaling

Time Slot Signaling functions affect the transmitted signaling portion of a time slot. The **dt_setsigmod( )** function selects the origin of the signaling information. The signaling information can either be inserted by the Digital Network Interface hardware or derived (by way of the PCM Expansion Bus) from a PEB-compatible resource device (such as a D/12x) or another network device. The **dt_settssig( )** function sets the state of the signaling bits when the signaling information is inserted by the DTI/2xx board (signaling insertion mode). The **dt_xmitwink( )** function transmits wink signaling to the network on any of the available signaling bits (for T-1, bit A or B; for E-1, bit A, B, C, or D).

NOTES: 1. The signaling bit and polarity used for wink signaling are only configurable through the download parameter file. See the *System Release Software Installation Reference for Windows NT* for details.

2. Dialing, supported on the DTI/101 device and the D/240SC-T1, is not supported by and will not operate on DTI/2xx devices. If your configuration includes Voice boards, you can use a Windows NT Voice library function instead. The function name is **dx_dial( )**.

3.2. Digital Network Interface Error Handling

All Network Windows NT library functions return a value that indicates the success or failure of the function call. Generally, Network library functions return the following values:

• 0	function success
• -1	general error
• AT_FAILURE	Extended Attribute function error from a function that returns a value
• AT_FAILUREP	Extended Attribute function error from a function that returns a pointer

If a function fails, the error code can be retrieved using the Dialogic Standard Runtime Library (SRL) **ATDV_LASTERR( )** function. Each function description in *Chapter 3. Digital Network Interface* includes a list of the errors that can occur for that function. These error codes are defined in *dtilib.h* and listed in *Table 2*.

NOTES: 1. The Network **dt_open( )** function call returns a Dialogic device handle if the function call is successful. A device handle is a positive non-zero value. If **dt_open( )** fails, the return code is -1 and the specific error is a Windows NT system error which can be found in the global variable **errno**, contained in *errno.h*.

2. The **ATDT_BDSGBIT( )** function call returns the value AT_FAILUREP on error. All other Extended Attribute functions return AT_FAILURE on error.

3. The SRL Standard Attribute functions **ATDV_LASTERR( )** and **ATDV_ERRMSGP( )** can be used to obtain the status of the last function call of the device. Refer to *Appendix A* for more information.

4. If the error returned by **ATDV_LASTERR( )** is EDT_SYSTEM, a Windows NT system error has occurred; you must check the global variable **errno** contained in *errno.h*.

Table 2. Error Types Defined in *dtilib.h*

Error Returned	Description
EDT_ABORT	abort received response
EDT_ADDRS	bad address
EDT_BADBRDERR	DTI/2xx missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_BADCNT	count of bytes requested is bad
EDT_BADDEV	bad device error
EDT_BADGLOB	bad global (device) parameter number
EDT_BADPORT	1st byte appeared on reserved port
EDT_BADVAL	invalid parameter value passed in value pointer
EDT_BITBSY	bit is already set
EDT_CHKSUM	bad checksum
EDT_DATTO	data reception timed out
EDT_DTTSTMOD	in test mode; cannot set DTI/2xx mode
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid DTI/2xx logical board device handle
EDT_INVCFG	invalid configuration area or EEPROM configuration data
EDT_INVMSG	invalid message
EDT_INVSIGST	invalid signaling state
EDT_INVTS	invalid DTI/2xx logical time slot device handle
EDT_MBFMT	wrong number of bytes for multiple byte request
EDT_MBIMM	received an immediate termination
EDT_MBINV	1st byte appeared on data port
EDT_MBOVR	message was too long, overflow

Error Returned	Description
EDT_MBPORT	received multiple byte data on port other than 0 or 1
EDT_MBTERM	terminating byte other than FEH or FFH
EDT_MBUND	under the number of bytes for a multibyte request
EDT_MSGCNT	count received did not match actual count
EDT_MTSIG	cannot disable insertion
EDT_NOCLK	no PEB clocking source
EDT_NOIDLEERR	time slot is not in idle/closed state
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_NOTDNLD	not downloaded
EDT_NOTPEBMODE	not in PEB mode
EDT_NOTSACS	cannot use tsacs on the device
EDT_NOWTCALL	not waiting for a call
EDT_PARAMERR	invalid parameter
EDT_PDOFFHK	wink bit not in correct initial state
EDT_PDSIG	cannot disable insertion
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SH_BADEXTTS	external time slot unsupported at current clock rate
EDT_SH_BADINDX	invalid switching handler index number
EDT_SH_BADLCLTS	invalid local time slot number
EDT_SH_BADMODE	invalid bus mode
EDT_SH_BADTYPE	invalid local time slot type
EDT_SH_LCLDSCNCT	local time slot already disconnected from SCbus
EDT_SH_LCLTSCNCT	local time slot already connected to Scbus
EDT_SH_LIBBSY	switching handler library is busy
EDT_SH_LIBNOTINIT	switching handler library has not been

Error Returned	Description
	initialized
EDT_SH_MISSING	switching handler is not present
EDT_SH_NOCLK	clock fallback failed
EDT_SIGINS	signaling insertion not enabled
EDT_SIGTO	transmit/receive did not update in time
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_STARTED	cannot start when already started
EDT_SUCC	no error
EDT_SYSTEM	Windows NT system error. Check the global variable errno for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware
EDT_TSASN	time slot already assigned
EDT_TSBSY	time slot is busy

3.3. Include Files

The Network Windows NT library function prototypes and defines are listed in the *dtilib.h* file supplied with the System Release Development Package for Windows NT. Applications that use Network Windows NT library functions must include the following statements:

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
```

To perform error handling in your routines, your source code must include the following line:

```
#include <errno.h>
```

Code that uses Voice devices and version 3.00 of the Windows NT Voice Driver with Digital Network Interface devices must include the following statements, in the following order:

```
#include <windows.h>
#include <srllib.h>
#include <dxxxlib.h>
#include <dtilib.h>
#include <errno.h>
```

4. Digital Network Interface Function Reference

Digital Network Interface Function Overview

This chapter contains an alphabetical listing of all Dialogic Network Windows NT library functions. Extended Attribute functions, also contained in the Network Windows NT library, are described here as well (because the functions appear alphabetically, the Extended Attribute functions are located together near the front of the reference). For information about Standard Attribute functions, refer to *Appendix A*.

NOTE: In the context of this guide, "Digital Network Interface" is used to refer to the DTI/211 board, the DTI/212 board, the D/240SC-T1 board, and the D/300SC-E1 board unless otherwise noted.

Name:	long ATDT_BDMODE(devh)
Inputs:	int devh • Dialogic Digital Network Interface logical board device handle
Returns:	signaling mode of all Digital Network Interface time slots AT_FAILURE if failure
Includes:	srllib.h dtilib.h
Category:	Extended Attribute
Mode:	synchronous

■ Description

The **ATDT_BDMODE( )** function returns the current mode of every time slot on the specified Digital Network Interface device.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.

For T-1 applications, the mode is returned as a long integer where bits 0 to 23 represent the mode of Digital Network Interface time slots 1 to 24.

For E-1 applications, the mode is returned as a long integer where bits 0 to 29 represent the mode of Digital Network Interface time slots 1 to 30.

The following signaling mode defines are provided in *dtilib.h*:

- DTM_SIGINS - signaling insertion mode (Digital Network Interface board generates signaling to network)
- DTM_TRANSP - transparent signaling mode (D/12x or other Voice board generates signaling to network)

To determine the mode of a time slot, compare the returned value with the provided defines.

Cautions

1. This function will fail if an invalid Digital Network Interface logical board device handle is specified.
2. Unless DTI/2xx signaling is set to transparent (DTM_TRANSP), wink signaling cannot be transmitted on a voice device channel (see the *Voice Software Reference for Windows NT*).

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;           /* Board device handle */
   long modebits;      /* Mode of all time slots */
   int i;              /* Loop counter */

   /*
    * Open board 1 device
    */

   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get the signaling mode of all E-1 time slots (1 to 30)
    */

   if ( ( modebits = ATDT_BDMODE( devh ) ) == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Display it
    */for ( i = 0; i < 30; i++ ) {
      switch( ( modebits >> i ) & 1 ) {
      case DTM_TRANSP:
         printf( "Time slot %d on board 1 is in transparent mode\n", i + 1 );
         break;
      case DTM_SIGINS:
         printf( "Time slot %d on board 1 is in insertion mode\n", i + 1 );
         break;
      }
   }
            .
            .
```

```
}
```

■ Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_INVMSG	invalid message
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDSIGBIT( )**
- **ATDT_TSMODE( )**
- **ATDT_TSSGBIT( )**
- **dt_setsigmod( )**
- **dt_settssig( )**

Name:	char * ATDT_BDSGBIT(devh)
Inputs:	int devh • Dialogic Digital Network Interface logical board device handle
Returns:	pointer to signaling bit states of all device time slots AT_FAILUREP if failure
Includes:	srllib.h dtilib.h
Category:	Extended Attribute
Mode:	synchronous

■ Description

The **ATDT_BDSGBIT( )** function returns the current state of the transmit and receive , bits for all time slots on the Digital Network Interface device specified in **devh**.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.

For T-1 applications, the returned value is a pointer to a 24-byte buffer. Bytes 0 to 23 represent PEB time slots 1 to 24.

For E-1 applications, the returned value is a pointer to a 30-byte buffer. Bytes 0 to 29 represent PEB time slots 1 to 30.

The following symbols represent each signaling bit and are defined in *dtilib.h*:

- DTSG_RCVA - “A” receive signaling bit
- DTSG_RCVB - “B” receive signaling bit
- DTSG_RCVC - “C” receive signaling bit (E-1 only)
- DTSG_RCVD - “D” receive signaling bit (E-1 only)
- DTSG_XMTA - “A” transmit signaling bit
- DTSG_XMTB - “B” transmit signaling bit
- DTSG_XMTC - “C” transmit signaling bit (E-1 only)
- DTSG_XMTD - “D” transmit signaling bit (E-1 only)

To determine the state of the signaling bits, perform a logical AND operation on the byte buffer and the defines, as demonstrated in the example below.

■ Cautions

1. This function will fail if an invalid Digital Network Interface logical board device handle is specified. AT_FAILUREP will be returned.
2. The transmit signaling bits are only valid when the device is in signaling insertion mode.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                    /* Board device handle */
   char *sigbits;               /* Pointer to signaling bits array */
   int i;                       /* Loop counter */
   int arcv, brcv, axmt, bxmt;  /* Bit mask values */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get current transmit and receive signaling bits of all time slots
    */
   if ( ( sigbits = ATDT_BDSGBIT( devh ) ) == AT_FAILUREP ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Display it
    */
   for ( i = 0; i < 24; i++ ) {
      arcv = ( sigbits[ i ] & DTSG_RCVA ) ? 1 : 0;
      brcv = ( sigbits[ i ] & DTSG_RCVB ) ? 1 : 0;
      axmt = ( sigbits[ i ] & DTSG_XMTA ) ? 1 : 0;
      bxmt = ( sigbits[ i ] & DTSG_XMTB ) ? 1 : 0;
      printf( "tslot #%d arcv = %d, brcv = %d, axmt = %d, bxmt = %d\n",
              i + 1, arcv, brcv, axmt, bxmt );
   }
```

```
                .
                .
                .
}
```

■ Errors

If the function returns AT_FAILUREP, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_INVMSG	invalid message
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDMODE( )**
- **ATDT_TSMODE( )**
- **ATDT_TSSGBIT( )**
- **dt_setsigmod( )**
- **dt_settssig( )**

Name:	long ATDT_DNLDVER(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
Returns:	version of firmware used by the device AT_FAILURE if failure	
Includes:	srllib.h dtilib.h	
Category:	Extended Attribute	
Mode:	synchronous	

■ Description

The **ATDT_DNLDVER( )** function returns the firmware version downloaded to the device specified in **devh**. This number is returned in the standard Dialogic version numbering format.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.

■ Dialogic Version Numbering

A Dialogic version number consists of two parts that provide:

- The release TYPE
  Example: Production or Beta
- The release NUMBER, which consists of different elements depending on the type of release.

Example: 1.00 Production
1.00 Beta 5

NOTE: The examples above are shown in the convention used by Dialogic to display version numbers.

This function returns the version number as a long integer (32 bits) in BCD (binary coded decimal) format.

Nibble 1 returns the type of release in BCD numbers. A converted value of 0 indicates a Production release and a converted value of 1 indicates a Beta release.

Nibbles 2, 3, and 4 return the Production Release Number.

NOTE: Nibbles 2 through 4 are used in all version numbers. Nibbles 5 through 8 only contain values if the release is not a production release.

Nibbles 5, 6, 7, and 8 return the Internal Release Number used for pre-production product releases. Nibbles 5 and 6 hold the product's Beta number. Nibbles 7 and 8 hold additional information used for internal releases.

Table 3 displays a breakdown of the values returned by each nibble in the long integer.

Table 3. ATDT_DNLDVER( ) Return Values

Nibble (4 bits)				
1	**2**	**3 & 4**	**5 & 6**	**7 & 8**
TYPE	**PRODUCTION RELEASE NUMBER**		**INTERNAL NUMBER**	
Production	Major Release No.	Minor Release No.	N/A	N/A
Beta	Major Release No.	Minor Release No.	Beta Number	N/A

■ Major and Minor Release Numbers

Major and minor release numbers distinguish major revisions from minor revisions to Production releases. The major number converts to a single digit

integer that increments with each major revision to the release. The minor number converts to a two digit integer that increments with each minor revision to the release.

In decimal number format, the major number is the number before the decimal point, and the minor number is the number after the decimal point.

The following list gives examples of each type of release. The values used in these examples have been converted from the binary coded decimal numbers returned in the long integer and are displayed according to Dialogic convention.

1.00 Production
1.00 Beta 5

■ Cautions

This function will fail if an invalid Digital Network Interface logical board device handle is specified.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;      /* Board device handle */
   long version;      /* Version number of firmware */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get the version number of the firmware
    */
   version = ATDT_DNLDVER( devh );
   if ( version == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
```

```
   /*
    * Display it
    */
   printf( "DTI/2xx Download version number is %d.%02x\n",
           (int)((version>>24)&0x0F), ((version >>16)&0xFF) );

               .
               .
               .
}
```

■ Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_INVMSG	invalid message
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_ROMVER( )**

Name:	long ATDT_IDLEST(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
Returns:	idling state of time slot AT_FAILURE if failure	
Includes:	srllib.h dtilib.h	
Category:	Extended Attribute	
Mode:	synchronous	

■ Description

The **ATDT_IDLEST( )** function returns the current idle state of the Digital Network Interface time slot specified in **devh**. "Idling" transmits silence to the network for the selected time slot.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.

The following defines are provided in *dtilib.h*.

- DTIS_ENABLE - silence insertion is enabled
- DTIS_DISABLE - silence insertion is disabled

To determine if a time slot is idling, compare the value of the returned integer with the provided defines.

■ Cautions

This function will fail if an invalid Digital Network Interface logical time slot device handle is specified.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                  /* Time slot device handle */
   long mode;                 /* Time slot idle state mode */

   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get silence insertion mode
    */
   if ( ( mode = ATDT_IDLEST( devh ) ) == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   switch ( mode ) {
   case DTIS_ENABLE:
      printf( "Time slot 1 on board 1 has silence insertion enabled\n" );
      break;
   case DTIS_DISABLE:
      printf( "Time slot 1 on board 1 has silence insertion disabled\n" );
      break;
   }

               .
               .
               .
}
```

Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver

Equate	Returned When
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_INVMSG	invalid message
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*

■ See also

- **dt_setidle( )**

Name:	long ATDT_ROMVER(devh)
Inputs:	int devh • Dialogic Digital Network Interface logical board device handle
Returns:	version of EPROM installed on Digital Network Interface device AT_FAILURE if function fails
Includes:	srllib.h dtilib.h
Category:	Extended Attribute
Mode:	synchronous

Description

The **ATDT_ROMVER( )** function returns the version of the EPROM that is installed on the Digital Network Interface device specified in **devh**. This number is returned in the standard Dialogic version numbering format.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.

Dialogic Version Numbering

A Dialogic version number consists of two parts that provide:

- The release TYPE
  Example: Production or Beta
- The release NUMBER, which consists of different elements depending on the type of release.

Example: 1.00 Production
1.00 Beta 5

NOTE: The examples above are shown in the convention used by Dialogic to display version numbers.

This function returns the version number as a long integer (32 bits) in BCD (binary coded decimal) format.

Nibble 1 returns the type of release in BCD numbers. A converted value of 0 indicates a Production release and a converted value of 1 indicates a Beta release.

Nibbles 2, 3, and 4 return the Production Release Number.

NOTE: Nibbles 2 through 4 are used in all version numbers. Nibbles 5 through 8 only contain values if the release is not a production release.

Nibbles 5, 6, 7, and 8 return the Internal Release Number used for pre-production product releases. Nibbles 5 and 6 hold the product's Beta number. Nibbles 7 and 8 hold additional information used for internal releases.

Table 4 displays a breakdown of the values returned by each nibble in the long integer.

Table 4. ATDT_ROMVER( ) Return Values

Nibble (4 bits)				
1	**2**	**3 & 4**	**5 & 6**	**7 & 8**
TYPE	**PRODUCTION RELEASE NUMBER**		**INTERNAL NUMBER**	
Production	Major Release No.	Minor Release No.	N/A	N/A
Beta	Major Release No.	Minor Release No.	Beta Number	N/A

■ Major and Minor Release Numbers

Major and minor release numbers distinguish major revisions from minor revisions to Production releases. The major number converts to a single digit

integer that increments with each major revision to the release. The minor number converts to a two digit integer that increments with each minor revision to the release.

In decimal number format, the major number is the number before the decimal point, and the minor number is the number after the decimal point.

The following list gives examples of each type of release. The values used in these examples have been converted from the binary coded decimal numbers returned in the long integer and are displayed according to Dialogic convention.

1.00 Production
1.00 Beta 5

■ Cautions

This function will fail if an invalid Digital Network Interface logical board device handle is specified.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;      /* Board device handle */
   long version;      /* Version number of EPROM */


   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }


   /*
    * Get the version number of the EPROM
    */
   version = ATDT_ROMVER( devh );
   if ( version == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
```

```
   /*
    * Display it
    */
   printf( "DTI/2xx EPROM version number is %d.%02x\n",
           (int)((version>>24)&0x0F), ((version >>16)&0xFF) );

               .
               .
               .
}
```

■ Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h.*

■ See also

- **ATDT_DNLDVER( )**

Name:	long ATDT_STATUS(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
Returns:	status of time slot AT_FAILURE if failure	
Includes:	srllib.h dtilib.h	
Category:	Extended Attribute	
Mode:	synchronous	

■ Description

The **ATDT_STATUS( )** function returns the current status of the Digital Network Interface time slot specified in **devh**.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.

The following defines are provided:

- DTST_INACTIVE - time slot is idle
- DTST_BUSY - time slot is not idle

To determine the status of the time slot, compare the value of the returned integer with the defines listed above.

The time slot is considered busy if it is currently executing a multitasking function, for example, wink signaling.

■ Cautions

This function will fail if an invalid Digital Network Interface logical time slot device handle is specified.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   long mode;         /* Current status of time slot */

   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get current wink status of time slot
    */
   if ( ( mode = ATDT_STATUS( devh ) ) == AT_FAILURE ) {
      printf( "Error message = %s.", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Display it
    */
   switch ( mode ) {
   case DTST_INACTIVE:
      printf( "Time slot 1 on board 1 is idle\n" );
      break;
   case DTST_BUSY:
      printf( "Time slot 1 on board 1 is busy\n" );
      break;
   }

               .
               .
               .
}
```

■ Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVMSG	invalid message
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **dt_xmitwink( )**

Name:	long ATDT_TSMODE(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
Returns:	time slot signaling mode AT_FAILURE if failure	
Includes:	srllib.h dtilib.h	
Category:	Extended Attribute	
Mode:	synchronous	

■ Description

The **ATDT_TSMODE( )** function returns the current signaling mode of the time slot specified in **devh**.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.

The following defines are provided in *dtilib.h*.

- DTM_SIGINS - signaling insertion mode (Digital Network Interface board generates signaling to network)
- DTM_TRANSP - transparent signaling mode (D/12x or other Voice board generates signaling to network)

To determine the signaling mode of a specified time slot, compare the returned value with the defines listed above.

■ Cautions

This function will fail if an invalid Digital Network Interface logical time slot device handle is specified.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   long mode;         /* Time slot signaling mode */

   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get current time slot signaling mode
    */
   if ( ( mode = ATDT_TSMODE( devh ) ) == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Display it
    */
   switch ( mode ) {
   case DTM_SIGINS:
      printf( "Time slot 1 on board 1 has signaling insertion\n" );
      break;
   case DTM_TRANSP:
      printf( "Time slot 1 on board 1 has signaling transparent\n" );
      break;
   }

            .
            .
            .
}
```

Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDSGBIT( )**
- **ATDT_BDMODE( )**
- **dt_setsigmod( )**
- **dt_settssig( )**

Name:	long ATDT_TSSGBIT(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
Returns:	state of time slot signaling bits AT_FAILURE if failure	
Includes:	srllib.h dtilib.h	
Category:	Extended Attribute	
Mode:	synchronous	

■ Description

The **ATDT_TSSGBIT( )** function retrieves the current state of the transmit and receive signaling bits for the time slot specified by **devh**.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.

The returned bitmask represents the following signaling bits:

- DTSG_RCVA - “A” receive signaling bit
- DTSG_RCVB - “B” receive signaling bit
- DTSG_RCVC - “C” receive signaling bit (E-1 only)
- DTSG_RCVD - “D” receive signaling bit (E-1 only)
- DTSG_XMTA - “A” transmit signaling bit
- DTSG_XMTB - “B” transmit signaling bit
- DTSG_XMTC - “C” transmit signaling bit (E-1 only)
- DTSG_XMTD - “D” transmit signaling bit (E-1 only)

To determine the state of the signaling bits for the specified time slot, perform a logical AND operation on the byte buffer and the defines, as demonstrated in the example below.

Cautions

1. This function will fail if an invalid Digital Network Interface logical time slot device handle is specified.
2. The transmit signaling bits are only valid when the device is in signaling insertion mode.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


main()
{
   int devh;         /* Time slot device handle */
   long tsbits;         /* Time slot signaling bits */
   int arcv, brcv, axmt, bxmt;   /* Bit mask values */
   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Get time slot signaling bits
    */
   tsbits = ATDT_TSSGBIT( devh );
   if ( tsbits == AT_FAILURE ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Display it
    */
   arcv = ( tsbits & DTSG_RCVA ) ? 1 : 0;
   brcv = ( tsbits & DTSG_RCVB ) ? 1 : 0;
   axmt = ( tsbits & DTSG_XMTA ) ? 1 : 0;
   bxmt = ( tsbits & DTSG_XMTB ) ? 1 : 0;
   printf( "tslot 1 arcv = %d, brcv = %d, axmt = %d, bxmt = %d\n",
           arcv, brcv, axmt, bxmt );
              .
              .
              .
}
```

Errors

If the function returns AT_FAILURE, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to

obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid command parameter to driver
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_RANGERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDSGBIT( )**
- **ATDT_BDMODE( )**
- **ATDT_TSMODE( )**
- **dt_setsigmod( )**
- **dt_settssig( )**

Name:	int dt_close(devh)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board or Digital Network Interface logical time slot device handle
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Resource Management	
Mode:	synchronous	

Description

The **dt_close( )** function closes Digital Network Interface devices opened previously by a call to **dt_open( )**. The specified device may be either a Digital Network Interface logical board or time slot device. The **dt_close( )** function releases the handle and breaks the link between the calling process and the device.

Parameter	Description
devh::	Specifies the valid Digital Network Interface logical board or Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.

Cautions

1. This function will fail if the device handle is invalid.
2. The **dt_close( )** function affects only the link between the calling process and the device. Other processes are unaffected by **dt_close( )**.
3. If event notification is active for the device to be closed, call the SRL **sr_dishdlr( )** function prior to calling **dt_close( )**.
4. A call to **dt_close( )** does not affect the configuration of the device.
5. Dialogic devices should **never** be closed using the Windows NT **close( )**.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Board device handle */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    *   Continue processing
    */
               .
               .
               .
   /*
    * Done processing - close device.
    */

   if ( dt_close( devh ) == -1 ) {
      printf( "Cannot close board dtiB1.  errno = %d", errno );
   }
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the following error value:

EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.

See also

- **dt_open( )**

Name:	int dt_dial(devh,digstr,tmo)	
Inputs:	int devh	• Dialogic time slot device handle
	char *digstr	• pointer to an ASCIIZ string of digits
	unsigned int tmo	• timeout value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous/asynchronous	

■ Description

The **dt_dial( )** function allows the application to pulse dial an ASCIIZ string of digits on a specified D/240SC-T1 or D/300SC-E1 time slot. The function can operate in either the synchronous (blocking) or asynchronous (non-blocking) mode.

Parameter	Description
devh:	Specifies the valid time slot device handle returned by a call to **dt_open( )**. The specified time slot must be in the offhook, idle state when **dt_dial( )** is called.
digstr:	Pointer to the ASCIIZ string of digits to dial. The maximum length of the string is 32 digits.
tmo:	Specifies the maximum number of seconds that the function will block while awaiting a dial status response from the D/240SC-T1 or D/300SC-E1 .

■ Asynchronous Mode

To operate this function in asynchronous (non-blocking) mode, specify 0 for **tmo**. This allows the application to continue processing while awaiting a completion event. If event handling is set up properly for your application, DTEV_PDDONE

will be returned by the **sr_getevttype( )** function included in the SRL when the dial is successfully completed. See *Appendix A* for information on event handling.

Synchronous Mode

To operate the function in synchronous (blocking) mode, specify a length of time in seconds that the function will block for **tmo**. This causes the application to wait for a return from the function before performing any other processing. A suggested **tmo** setting for this function is 60.

Cautions

1. This function will fail under the following conditions:
   - A logical board or invalid time slot device handle is specified.
   - More than a 32 digit buffer is passed.
   - There is insufficient memory.
   - Signaling insertion is not enabled.
   - The time slot is already pulse dialing.
   - The time slot is not in the offhook idle state.
2. This function is not supported by DTI/211 and DTI/212 devices. To pulse dial a string of ASCIIZ digits on a system using DTI/211, DTI/212, and Voice devices, you can use the Voice library function **dx_dial( )** (see the *Voice Software Reference for Windows NT*).
3. The time slot must be in signaling insertion mode before this function is called. Signaling insertion mode is enabled using the **dt_setsigmod( )** function.
4. To use this function in asynchronous mode, you must use the **dt_setevtmsk( )** and SRL **sr_enbhdlr( )** functions to enable trapping the completion event and create an event handler to process the event. See *Appendix A* for more information on Digital Network Interface event management.
5. Make sure adequate time is given to the function to complete the dial if the synchronous mode is used.

Example

```
#include <windows.h>
```

```
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


/*
 * Basic error handler
 */
do_error( devh, funcname )
   int devh;
   char *funcname;
{
   int errorval = ATDV_LASTERR( devh );
   printf( "Error while calling function %s.\n", funcname );
   printf( "Error value = %d.  Error message = %s.", errorval,
          ATDV_ERRMSGP( devh ) );
   if ( errorval == EDT_SYSTEM ) {
      printf( "errno = %d.\n", errno );
   } else {
      printf( "\n" );
   }
}


main()
{
   int tsdev;         /* Time Slot device handle */
   /*
    * Open time slot 1 on board 1
    */
   if ( ( tsdev = dt_open( "dtiB1T1", 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }
   /*
    * Set signaling mode to signaling insertion
    */


if ( dt_setsigmod( tsdev, DTM_SIGINS ) == -1 ) {
      do_error( tsdev, "dt_setsigmod()" );
      exit( 1 );
   }

   /*
    * Disable silence transmission
    */
   if ( dt_setidle( tsdev, DTIS_DISABLE ) == -1 ) {
      do_error( tsdev, "dt_setidle()" );
      exit( 1 );
   }
   /*
    * Go offhook
    */
   if ( dt_settssig( tsdev, DTB_ABIT | DTB_BBIT, DTA_SETMSK ) == -1 ) {
      do_error( tsdev, "dt_settssig()" );
      exit( 1 );
   }
   /*
    * Dial number with 60 second timeout.  Note that this is the blocking
    * mode dial.
    */
   if ( dt_dial( tsdev, "7223689", 60 ) == -1 ) {
      do_error( tsdev, "dt_dial()" );
      exit( 1 );
   }
```

```
   /*
    *   Continue processing
    *           .
    *           .
    *           .
    */
   /*
    * Done processing - close device.
    */
   if ( dt_close( tsdev ) == -1 ) {
      do_error( tsdev, "dt_close()" );
   }
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code. See *Appendix A* for more information on SRL functions. The error codes returned by ATDV_LASTERR are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware
EDT_INVTS	invalid time slot device handle
EDT_SIGINS	signaling insertion not enabled
EDT_TSBSY	time slot is busy
EDT_PDOFFHK	not in offhook idle state

EDT_PDSIG cannot disable insertion when pulse dialing

Error defines can be found in the file *dtilib.h*.

See also

In *Voice Software Reference for Windows NT*:

- **dx_dial( )**

Name:	int dt_getctinfo(devh,ct_devinfop)	
Inputs:	int devh	• D/240SC-T1 or D/300SC-E1 Digital network interface device time slot handle
	CT_DEVINFO *ct_devinfop	• pointer to device information structure
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	SCbus routing	
Mode:	Synchronous	

■ Description

The **dt_getctinfo( )** function returns information about the Digital network interface device associated with the specified digital channel (time slot) (dtiBxTx) on a D/240SC-T1 or D/300SC-E1 board.

Parameter	Description
devh:	Specifies the valid digital network interface time slot device handle returned by a call to **dt_open( )**.
ct_devinfop:	Specifies the pointer to the data structure CT_DEVINFO.

On return from the function, the CT_DEVINFO structure contains the relevant information and is declared as follows:

```
typedef struct {
   unsigned long   ct_prodid;
   unsigned char   ct_devfamily;
   unsigned char   ct_devmode;
   unsigned char   ct_nettype;
   unsigned char   ct_busmode;
   unsigned char   ct_busencoding;
   unsigned char   ct_rfu[7];
} CT_DEVINFO;
```

Valid values for each member of the CT_DEVINFO structure are defined in dtilib.h. Possible return values are:

ct_prodid:	field contains a valid Dialogic product identification number for the device.	
ct_devfamily:	specifies the device family and contains:	
	CT_DFSPAN	specifies a D/240SC-T1 or D/300SC-E1 digital interface device.
ct_devmode:	not valid for D/240SC-T1 or D/300SC-E1 devices.	
ct_nettype:	specifies the type of network interface for the device. The two valid values are:	
	CT_NTT1	specifies a D/240SC-T1 T-1 digital channel.
	CT_NTE1	specifies a D/300SC-E1 E-1 digital channel.
ct_busmode:	specifies the bus architecture used to communicate with other devices in the system. The two valid values are:	
	CT_BMPEB	specifies PEB (PCM Expansion Bus) architecture.
	CT_BMSCBUS	specifies SCbus architecture.
ct_busencoding:	describes the PCM encoding used on the bus. Valid values are:	
	CT_BEULAW	specifies Mu-law encoding.
	CT_BEALAW	specifies A-law encoding.

■ Cautions

This function will fail if an invalid time slot device handle is specified.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main( )
{

   int devh;                  /* Digital network interface device handle */
   CT_DEVINFO ct_devinfo;     /* Device information structure */
```

```
   /* Open board 1 time slot 1 on Digital network interface device */
   if ((devh = dt_open("dtiB1T1", 0)) == -1) {
      printf("Cannot open time slot dtiB1T1.  errno = %d", errno);
      exit(1);
   }

   /* Get Device Information */
   if (dt_getctinfo(devh, &ct_devinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(devh));
      exit(1);
   }

   printf("%s Product Id = 0x%x, Family = %d, Network = %d, Bus mode = %d,
   Encoding = %d", ATDV_NAMEP(devh), ct_devinfo.ct_prodid,
   ct_devinfo.ct_devfamily, ct_devinfo.ct_nettype, ct_devinfo.ct_busmode,
   ct_devinfo.ct_busencoding);
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Board missing or defective
EDT_BADCMDERR	Invalid command parameter to driver
EDT_FWERR	Firmware returned an error
EDT_INVTS	Invalid time slot device handle
EDT_INVMSG	Invalid message
EDT_SH_BADLCLTS	Invalid local time slot number
EDT_SH_BADINDX	Invalid Switch Handler library index number
EDT_SH_BADTYPE	Invalid local time slot type
EDT_SH_LIBBSY	Switch Handler library busy
EDT_SH_LIBNOTINIT	Switch Handler library is uninitialized
EDT_SH_MISSING	Switch Handler is not present
EDT_SH_NOCLK	Switch Handler clock fallback failed
EDT_SYSTEM	Windows NT system error
EDT_TMOERR	Timed out waiting for reply from firmware

See also

In *Voice Software Reference for Windows NT*:

- **ag_getctinfo( )**

Name:	dt_GetDllVersion (dwfileverp, dwprodverp)	
Inputs:	LPDWORD dwfileverp	• Network DLL Version Number
	LPDWORD dwprodverp	• Product version of this release
Returns:	0 if success	
	-1 if failure	
Includes:	srllib.h	
	dxxxlib.h	
	dtilib.h	

■ Description

The **dt_GetDllVersion( )** function returns the Network DLL Version Number for the file and product.

This function has the following parameters:

Parameter	Description
dwfileverp	pointer to where to return file version information
dwprodverp	pointer to where to return product version information

■ Cautions

If using older DLL's with no version number stamps, a default version 4.10 (0x0004000A) is returned.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dxxxlib.h>
#include <dtilib.h>

int InitDevices()
{
   DWORD dwfilever, dwprodver;

   //
   // Initialize all the DLLs required. This will cause the DLLs to be
   // loaded and entry points to be resolved. Entry points not resolved
   // are set up to point to a default not implemented function in the
   // 'C' library. If the DLL is not found all functions are resolved
   // to not implemented.
```

```
   //

   if (sr_libinit(DLGC_MT) == -1) {
      // Must be already loaded, only reasoon if sr_libinit() was
     // already called

 }

   //
   // Call technology specific dt_libinit() functions to load Network DLL
   //
   if (dt_libinit(DLGC_MT) == -1) {
      // Must be already loaded, only reasoon if dx_libinit() was
      // already called
   }


   //
   // Network library initialised so all other DTI/ISDN/MSI functions may be called
   // as normal. Display the version number of the DLL  //

 dt_GetDllVersion(&dwfilever, &dwprodver);
  printf("File Version for network DLL is %d.%02d\n",
                              HIWORD(dwfilever), LOWORD(dwfilever));
   printf("Product Version for network DLL is %d.%02d\n",
                              HIWORD(dwprodver), LOWORD(dwprodver));
   //


   // Now open all the network devices
   //
}
```

■ Errors

None.

■ See Also

- **dx_GetDllVersion()**
- **fx_GetDllVersion()**
- **sr_GetDllVersion()**
- **vr_GetDllVersion()**

Name:	int dt_getevt(devh,eblkp,timeout)	
Inputs:	int devh	• D/240SC-T1 or D/300SC-E1 Digital network interface device time slot handle
	EV_EBLK *eblkp	• pointer to Event Block Structure
	int timeout	• timeout value in seconds
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Parameter Request	
Mode:	Synchronous	

■ Description

This **dt_getevt( )** function blocks and returns control to the program after one of the events set by **dt_setevtmsk( )** occurs on the channel specified in the **devh** parameter, or a timeout occurs. **dt_getevt( )** is used with multi-threaded applications only.

Parameter	Description
devh:	Specifies the valid digital network interface time slot device handle returned by a call to **dt_open( )**.
***eblkp:**	Points to the Event Block Structure DX_EBLK, which will contain the event that ended the blocking.
timeout:	Specifies the maximum amount of time in seconds to wait for an event to occur. **timeout** can have one of the following values: • # of seconds: maximum length of time to wait for an event. When time has elapsed, the function will terminate and return an error.

Parameter	Description
	• -1: block until an event occurs. The function will not timeout. • 0: return -1 immediately if no event is present.

NOTE: When the time specified expires, **dt_getevt( )** will terminate and return an error. The Standard Attribute function **ATDV_LASTERR( )** can be used to determine the cause of the error, which in this case is EDX_TIMEOUT.

On successful return from the function the event block structure will have the following information.

eblk.ev_dev:	device on which the event occured. This will be the same as the devh parameter passed in.
eblk.ev_event:	DTEV_SIG indicates signaling transition event. DTEV_T1ERRC indicates alarm.
eblk.ev_data[]:	DTEV_SIG contains information about the signalling event. ev_data[] is an array of bytes where ev_data[0] and ev_data[1] contain the signaling information. Retrieve the signaling information in a short variable and see the example below to get the signaling information from ev_data[0] and ev_data[1]. DTEV_T1ERRC contains information about the type of alarm occuring.

The event block structure is defined as follows:

```
typedef struct ev_eblk {
   long ev_dev;            /* Device on which event occured */
   unsigned long ev_event; /* Event type */
   long ev_len;            /* Length of data associated with event */
   char ev_data[8];        /* 8 byte data buffer */
   void * ev_datap;        /* Variable pointer if more than 8 bytes of
                              data */
} EV_EBLK;
```

■ Cautions

dt_getevt( ) is only used for multithreaded applications.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

EV_EBLK eblk;
main()
{
   int devh;                    /* Board device handle */
   unsigned short bitmaskp;     /* Bitmask variable */
   unsigned short sigmsk = DTMM_AON | DTMM_AOFF | DTMM_BON | DTMM_BOFF;
   short sig, indx;

   /*
    * Open Timeslot 1 device
    */
   if ( ( devh = dt_open( "dtiB1T1", 0 ) ) == -1 ) {
      printf( "Cannot open timeslot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }
    if (dt_setevtmsk(ddd, DTG_SIGEVT, sigmsk, DTA_SETMSK) == -1) {
       printf("%s: dt_setevtmsk DTG_SIGEVT DTA_SETMSK ERROR %d: %s: Mask =
       0x%x\n",ATDV_NAMEP(ddd),ATDV_LASTERR(ddd),ATDV_ERRMSGP(ddd),sigmsk);
       dt_close(ddd);
         exit(1);
      }
   /*
    *  Wait for events on this timeslot
    */
   while(1) {
      dt_getevt ( devh, &eblk, -1 );    /* Wait for ever */
      sig = eblk.ev_data[0]  |  ( (short) eblk.ev_data[1] << 8 );

   for (indx = 0; indx <4; indx++) {
      if (!(sig & (0x1010 << indx))) {
         continue;
      }
      switch (sig & (0x1111 << indx)) {
         case DTMM_AOFF:
             fprintf(stderr,"A-OFF ");
             break;

         case DTMM_AON:
            fprintf(stderr,"A-ON ");
            break;
         case DTMM_BOFF:
            fprintf(stderr,"B-OFF ");
            break;
         case DTMM_BON:
            fprintf(stderr,"B-ON ");
            break;
         }    /* End of switch Statement */
      }  /* end of for statement */
   } /* end of while statement */
```

```
                .
                .
                .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADPARM	Invalid parameter
EDT_SYSTEM	Windows NT system error
EDT_TMOERR	Timed out waiting for reply from firmware

■ See also

- **dt_getevtmsk( )**

Name:	int dt_getevtmsk(devh,event,bitmaskp)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board or Digital Network Interface logical time slot device handle
	int event	• event to retrieve
	unsigned short *bitmaskp	• pointer to bitmask variable
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Parameter Request	
Mode:	synchronous	

■ Description

The **dt_getevtmsk( )** function retrieves the current event bitmask(s) for the specified event type and Digital Network Interface logical board or time slot device. The function can be used to find which bitmask was set by the **dt_setevtmsk( )** function.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board or Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
event:	Specifies which event's bitmask will be retrieved. The possible values for **event** are: • DTG_T1ERREVT - get T-1 error bitmask (board level event) • DTG_E1ERREVT - get E-1 error bitmask (board level event) • DTG_SIGEVT - get signaling bitmask (time slot event) • DTG_PDIGEVT - determine if pulse digit detection is enabled or disabled for the selected time slot device
bitmaskp:	Variable that will contain the value of the bitmask.

Table 5. dt_getevtmsk( ) Return Values

event	return	description
DTG_T1ERREVT	DTEC_LOS	loss of T-1 digital signal mask
	DTEC_DPM	driver performance monitor mask
	DTEC_RED	receive red alarm mask
	DTEC_BPVS	bipolar violation count saturation mask
	DTEC_ECS	error count saturation mask
	DTEC_RYEL	receive yellow alarm mask
	DTEC_RCLX	receive carrier loss mask
	DTEC_FERR	frame bit error mask
	DTEC_B8ZSD	bipolar 8 zero substitution detect mask
	DTEC_RBL	receive blue alarm mask
	DTEC_RLOS	receive loss of sync mask
	DTEC_OOF	out of frame error mask
DTG_E1ERREVT	DEEC_RLOS	receive loss of sync mask
	DEEC_RUA1	receive unframed all ones alarm mask
	DEEC_FSERR	frame sync error mask
	DEEC_RRA	receive remote alarm mask
	DEEC_BPVS	bipolar violation count saturation mask
	DEEC_CECS	CRC error count saturation mask
	DEEC_ECS	error count saturation mask
	DEEC_LOS	loss of E-1 digital signal detected mask
	DEEC_DPM	driver performance monitor mask
	DEEC_MFSERR	multiframe sync error mask
	DEEC_RSA1	receive signaling all ones alarm mask
	DEEC_RDMA	receive distant multiframe alarm

event	return	description
		mask
DTG_SIGEVT	DTMM_AON	signaling bit A ON event mask
	DTMM_AOFF	signaling bit A OFF event mask
	DTMM_BON	signaling bit B ON event mask
	DTMM_BOFF	signaling bit B OFF event mask
	DTMM_WINK	receive wink signaling event mask
(E-1 only)	DTMM_CON	signaling bit C ON event mask
" "	DTMM_COFF	signaling bit C OFF event mask
" "	DTMM_DON	signaling bit D ON event mask
" "	DTMM_DOFF	signaling bit D OFF event mask
DTG_PDIGEVT	DTIS_ENABLE	pulse digit detection enabled
	DTIS_DISABLE	pulse digit detection disabled

NOTE: When the DTG_T1ERREVT, DTG_E1ERREVT, DTG_SIGEVT, or DTG_PDIGEVT event is generated, call the sr_getevtdatap( ) function in the event handler to get a pointer to the event value. The pointer should be cast to an unsigned short pointer and the event retrieved as an unsigned short value.

Refer to Appendix A for more information on SRL data structures and functions.

■ Cautions

This function will fail under the following conditions:

- The board or time slot device handle is invalid.
- The event field is invalid.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                          /* Board device handle */
   unsigned short bitmaskp;           /* Bitmask variable */
```

```
   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Get current T1 error mask
    */
   if ( dt_getevtmsk( devh, DTG_T1ERREVT, &bitmaskp ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Check for loss of T-1 digital signal
    */
   if ( bitmaskp & DTEC_LOS) {
      printf( "Loss of T-1 digital signal will be reported \n");
   }


            .
            .
            .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range

EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

In this guide:

- **dt_setevtmsk( )**

In *Appendix A*:

- **sr_enbhdlr( )**
- **sr_dishdlr( )**

Name:	int dt_getparm(devh,param,valuep)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned long param	• device parameter defined name
	void *valuep	• pointer to integer variable for parameter value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Parameter Request	
Mode:	synchronous	

■ Description

The **dt_getparm( )** function gets the current value of the selected Digital Network Interface device parameter.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
param:	Specifies the parameter to be examined.
valuep:	Points to the variable to which the value of the parameter will be assigned.

Table 6 lists each parameter name, its default value, and a brief description.

Table 6. dt_getparm( ) Parameters

#DEFINE	VALUE	DESCRIPTION
DTG_RDEBON	0-255 5 (default)	debounce value for receive signaling transitions from logical 0 to 1 and 1 to 0 (in 10 ms units). DTG_RDEBON is used for only the debounce on value for DTI/1xx boards, but is used for both debounce on and debounce off for Digital Network Interface boards.
DTG_RDEBOFF	0-255 5 (default)	included only for DTI/1xx boards, debounce off value for receive signaling transitions from logical 1 to 0 and 0 to 1 (in 10 ms units).
DTG_CABTYPE	- - -	line interface unit (LIU) cable length and type (T-1 only):
DTI/211 only		
	DTLL_G703	CCITT recommendation G.703, 2.048 Mhz.
	DTLL_FCC68	FCC part 68 option A, CSU
	DTLL_ANSIT1	ANSI T1.403, CSU
	DTLL_133ABAM	0-133 feet DSX-1 ABAM (default)
	DTLL_266ABAM	133-266 feet DSX-1 ABAM
	DTLL_399ABAM	266-399 feet DSX-1 ABAM
	DTLL_533ABAM	399-533 feet DSX-1 ABAM
	DTLL_655ABAM	533-655 feet DSX-1 ABAM
D/240SC-T1 only		
	DTLL_000	000-110 feet
	DTLL_110	110-220 feet
	DTLL_220	220-330 feet

#DEFINE	VALUE	DESCRIPTION
	DTLL_330	330-440 feet
	DTLL_440	440-550 feet
	DTLL_550	550-655 feet
	DTLL_655	655 feet or greater
	DTLL_SQUARE	square pulse
DTG_CODESUPR	- - -	bipolar format suppression value (T-1 only).
	DTSP_TRAN	transparent (default).
	DTSP_B8ZS	binary 8 zero suppression.
	DTSP_BIT7	bit 7 stuffing.
DTG_IDLTYP	- - -	gets IDLE value:
	IDLE_7F (default)	T-1 IDLE value is 7FH
	IDLE_54	E-1 IDLE value is 54H
	IDLE_FF	T-1 IDLE value is FFH
	IDLE_D5	E-1 IDLE value is D5H
DTG_SETBDMD	- - -	get device mode value. This parameter will NOT change the device mode if the Digital Network Interface remote loopback test switch is set to ON.
	DTMD_NORMAL	normal mode (default if Digital Network Interface remote loopback test switch is set to OFF).
	DTMD_XCVRLB	transceiver local loopback mode (used for Digital Network Interface testing).
	DTMD_LIULLB	line interface unit local loopback mode (used for Digital Network Interface testing).
	DTMD_LIURLB	line interface unit remote loopback mode (used by

#DEFINE	VALUE	DESCRIPTION
		network for network testing).
DTG_SETCLK	- - -	get clock source (see the System Release Software Installation Reference for Windows NT):
	DTC_LOOP (default)	loop timing (clock derived from receive sync; if RLOS detected, falls back to DTC_IND).
	DTC_IND	1.544 Mhz (T-1) or 2.048 Mhz (E-1) independent timing.
	DTC_NOCLK	no clock.
	DTC_EXT	external (clock derived from PEB).
DTG_OOFMAX	0 (default)	(T-1 only) number of out-of-frame errors to allow before sending an alarm (maximum <= 15). For the default value, an alarm is sent after first detected frame error.
DTG_PCDEAD	- - -	This parameter is provided only for backward compatibility with DTI/1xx applications. The following masks tell the DTI/1xx what to do when the DTI/1xx firmware cannot communicate with the PC (default = 0 for all masks):
	DTD_SNDIDLE	transmit IDLE (0 = NO, 1 = YES).
	DTD_IDLEVAL	IDLE value to transmit (0 = 7F for T-1, 54 for E-1; 1 = FF for T-1, D5 for E-1).
	DTD_STXSIG	(0 = NO, 1 = YES).
	DTD_SIGVAL	transmit signaling value (0 =

#DEFINE	VALUE	DESCRIPTION
		0, 1 = 1).
DTG_ECRRSTTM	10 (default)	(E-1 only) rate, in 100 ms units, to reset the following 3 error-count registers.
DTG_BPVCMAX	0 - 255 (255 default)	bipolar violation count saturation.
DTG_CECRMAX	0 - 255 (255 default)	(E-1 only) CRC error count saturation.
DTG_FECRMAX	0 - 255 (4 default)	(E-1 only) frame sync error count saturation.
DTG_FECSMAX	0 (default)	(T-1 only) frame error count saturation.
DTG_PREWINK	0 (default)	prewink transmit delay in 10 ms units.
DTG_WINKLEN	15 (default)	transmit wink duration in 10 ms units.
DTG_WINKMIN	10 (default)	minimum receive wink time in 10 ms units.
DTG_WINKMAX	32 (default)	maximum receive wink time in 10 ms units.
DTG_REDTIME	250 (default)	(T-1 only) time in 10 ms units during which loss of sync (LOS) must exist before declaring a red alarm.
DTG_RCOVRTM	DTI/211 - 1200 (default)	(T-1 only) time in 10 ms units after recovery of red alarm that a yellow alarm must still be transmitted.
	DTI/212 - 300 (default)	(E-1 only) time in 10 ms units after recovery of FECS alarm.
DTG_RXTXIDLE	0x0F0E (default)	used to set the receive and transmit idle patterns that must be present prior to waiting for a seizure. The upper byte represents the

#DEFINE	VALUE	DESCRIPTION
		receive signaling pattern, and the lower byte represents the transmit signaling pattern. Bits 0 to 3 represent transmit A, B, C, and D bits. Bits 8 to 11 represent receive A, B, C, and D bits. OFF=0 and ON=1.
DTG_SEIZESIG	0x0C0F (default)	used to set the receive signaling pattern that defines a line seizure and the transmit signaling pattern to use for a response. Bits 0 to 3 represent transmit A, B, C, and D bits. Bits 8 to 11 represent receive A, B, C, and D bits. OFF=0 and ON=1.

■ Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical board device handle is specified.
   - The parameter specified is invalid.
2. This function will not fail if time slot devices are open on the Digital Network Interface logical board device.
3. The value of the parameter returned by this function is an integer. The **valuep** pointer is the address of an integer, but should be cast as a void pointer when passed in the parameter field.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
```

```
   int devh;                          /* Board device handle */
   int valuep;                        /* Parameter value */
   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Get current clock parameter value
    */
   if ( dt_getparm( devh, DTG_SETCLK, ( void * )&valuep ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

/*
    * Report current clock setting
    */
   if ( valuep & DTC_LOOP) {
      printf( "Clock is set to loop timing \n");
   }

.
   .
   .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_BADGLOB	invalid **param** value
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range

EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

See also

- **dt_setparm( )**

Name:	int dt_getxmitslot(devh,sc_tsinfop)	
Inputs:	int devh	• D/240SC-T1 or D/300SC-E1 Digital network interface device time slot
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 if error	
Includes:	srllib.h dtilib.h	
Category:	SCbus routing	
Mode:	Synchronous	

■ Description

The **dt_getxmitslot( )** function returns the SCbus time slot connected to the transmit of a digital network interface device time slot on a D/240SC-T1 or D/300SC-E1 board.

Parameter	Description
devh:	Specifies the valid digital network interface time slot device handle returned by a call to **dt_open( )**.
sc_tsinfop:	Specifies the pointer to the data structure SC_TSINFO.

NOTE: The SCbus convenience funtion **nr_scroute( )** includes **dt_getxmitslot( )** functionality; see the *Voice Programmer's Guide for Windows NT*.

The sc_numts member of the SC_TSINFO structure must be initialized with the number of SCbus time slots requested ("1" for a digital network interface device time slot). The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to a valid array. Upon return from the function, the array will contain the number (between 0 and 1023) of the SCbus time slot on which the digital network interface device time slot transmits. The SC_TSINFO structure is declared as follows:

```
typedef struct {
   unsigned long   sc_numts;
```

```
    long            *sc_tsarrayp;
} SC_TSINFO;
```

A D/240SC-T1 or D/300SC-E1 digital network interface device time slot can transmit on only one SCbus time slot.

Cautions

This function will fail under the following conditions:

- An invalid time slot device handle is specified.
- A PEB time slot is requested.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main( )
{
   int devh;                /* Time slot device handle */
   SC_TSINFO sc_tsinfo;     /* Time slot information structure */
   long scts;               /* SCbus time slot */

   /* Open board 1 time slot 1 for Digital network interface device */
   if ((devh = dt_open("dtiB1T1", 0)) == -1) {
      printf("Cannot open time slot dtiB1T1.  errno = %d", errno);
      exit(1);
   }

   /* Fill in the SCbus time slot information */
   sc_tsinfo.sc_numts = 1;
   sc_tsinfo.sc_tsarrayp = &scts;

   /* Get SCbus time slot connected to transmit of time slot (digital
   channel) 1 on board 1 */
   if (dt_getxmitslot(devh, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(devh));
      exit(1);
   }

      printf("%s is transmitting on SCbus time slot %d", ATDV_NAMEP(devh),
      scts);
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to

obtain a descriptive error message. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Board missing or defective
EDT_BADCMDERR	Invalid command parameter to driver
EDT_FWERR	Firmware returned an error
EDT_INVTS	Invalid time slot device handle
EDT_INVMSG	Invalid message
EDT_SH_BADLCLTS	Invalid local time slot number
EDT_SH_BADINDX	Invalid Switch Handler library index number
EDT_SH_BADMODE	Invalid Switch Handler bus configuration
EDT_SH_BADTYPE	Invalid local time slot type
EDT_SH_LCLDSCNCT	Local time slot is already disconnected from SCbus
EDT_SH_LIBBSY	Switch Handler library busy
EDT_SH_LIBNOTINIT	Switch Handler library is uninitialized
EDT_SH_MISSING	Switch Handler is not present
EDT_SH_NOCLK	Switch Handler clock fallback failed
EDT_SYSTEM	Windows NT system error
EDT_TMOERR	Timed out waiting for reply from firmware

■ See also

In the *Voice Software Reference for Windows NT:*

- **ag_listen( )**

Name:	dt_libinit (flags)	
Inputs:	unsigned short flags	• Specifies the programming model
Returns:	0 if success -1 if failure	
Includes:	srllib.h dtilib.h msilib.h cclib.h	

■ Description

The **dt_libinit ( )** function initializes the Network Library DLL and resolves all entry points in the LIBDTIMT.DLL.

This function has the following parameter:

Parameter	Description
flags	This flag has two possible values: DLGC_MT - Specify if using a multi-threaded or window callback model. DLGC_ST - Specify if using the single threaded model.

■ Cautions

The **sr_libinit( )** function must be called prior to using the **dt_libinit( )** function.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dxxxlib.h>
#include <dtilib.h>

int InitDevices()

{
   DWORD dwfilever, dwprodver;
   //
   // Initialize all the DLLs required. This will cause the DLLs to be
   // loaded and entry points to be resolved. Entry points not resolved
   // are set up to point to a default not implemented function in the
```

```
   // 'C' library. If the DLL is not found all functions are resolved
   // to not implemented.
   //

   if (sr_libinit(DLGC_MT) == -1) {
      // Must be already loaded, only reasoon if sr_libinit() was
      // already called
   }

   //
   // Call technology specific dt_libinit() functions to load Network DLL
   //
   if (dt_libinit(DLGC_MT) == -1) {
      // Must be already loaded, only reasoon if dx_libinit() was
      // already called
   }

   //
   // Network library initialised so all other DTI/ISDN/MSI functions may be called
   // as normal. Display the version number of the DLL
   //
   dt_GetDllVersion(&dwfilever, &dwprodver);
   printf("File Version for network DLL is %d.%02d\n",
                              HIWORD(dwfilever), LOWORD(dwfilever));
   printf("Product Version for network DLL is %d.%02d\n",
                              HIWORD(dwprodver), LOWORD(dwprodver));

   //
   // Now open all the network devices
   //
}
```

■ Errors

The **dt_libinit( )** function fails if the library has already been initialized. For example, if you try to make a second call to **dt_libinit( )**, it fails.

■ See Also

- **dx_libinit()**
- **fx_libinit()**
- **vr_libinit()**

Name:	int dt_listen(devh,sc_tsinfop)	
Inputs:	int devh	• D/240SC-T1 or D/300SC-E1 Digital network interface device time slot
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 if error	
Includes:	srllib.h dtilib.h	
Category:	SCbus routing	
Mode:	Synchronous	

■ Description

The **dt_listen( )** function connects the receive (listen) of a digital network interface device time slot on a D/240SC-T1 or D/300SC-E1 board to an SCbus time slot.

Parameter	Description
devh:	Specifies the valid digital network interface time slot device handle returned by a call to **dt_open( )**.
sc_tsinfop:	Specifies the pointer to the data structure SC_TSINFO.

NOTE: The SCbus convenience funtion **nr_scroute( )** includes **dt_getxmitslot( )** functionality; see the *Voice Programmer's Guide for Windows NT*.

The sc_numts member of the SC_TSINFO structure must be initialized with the number "1". The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to an array that contains a valid SCbus time slot number. Upon return from the function, the receive of the digital network interface device time slot will be connected to this SCbus time slot.

The SC_TSINFO data is obtained by calling the **dt_getxmitslot()** function (or an equivalent) prior to calling the **dt_listen()** function. The SC_TSINFO data structure contains two fields. The first field specifies the number "1" (a digital network interface time slot can connect to only one SCbus time slot). The second

field points to the array that lists the SCbus time slot number (between 0 and 1023) of the voice device or other technology device to be connected. The SC_TSINFO structure is declared as follows:

```
typedef struct {
   unsigned long   sc_numts;
   long            *sc_tsarrayp;
} SC_TSINFO;
```

Although multiple D/240SC-T1 or D/300SC-E1 time slots may listen to the same SCbus transmit time slot, the receive of each D/240SC-T1 or D/300SC-E1 time slot can connect to only one SCbus time slot.

■ Cautions

This function will fail under the following conditions:

- An invalid time slot device handle is specified.
- An invalid SCbus time slot number is specified.
- A PEB time slot is requested.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main( )
{
   int voxh;                /* Voice channel device handle */
   int dtih;                /* Digital channel (time slot) device handle */
   SC_TSINFO  sc_tsinfo;    /* Time slot information structure */
   long scts;               /* SCbus time slot */

   /* Open board 1 channel 1 device */
   if ((voxh = dx_open("dxxxB1C1", 0)) == -1) {
      printf("Cannot open channel dxxxB1C1.  errno = %d", errno);
      exit(1);
   }

   /* Fill in the SCbus time slot information */
   sc_tsinfo.sc_numts = 1;
   sc_tsinfo.sc_tsarrayp = &scts;

   /* Get SCbus time slot connected to transmit of channel 1 on board 1 */
   if (dx_getxmitslot(voxh, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(voxh));
      exit(1);
   }

   /* Open board 1 time slot 1 on Digital network interface device */
```

```
   if ((dtih = dt_open("dtiB1T1", 0)) == -1) {
      printf("Cannot open time slot dtiB1T1.  errno = %d", errno);
      exit(1);
   }

   /* Connect the receive of digital channel (time slot) 1 on board 1 to
    SCbus transmit time slot of voice channel 1*/
   if (dt_listen(dtih, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(dtih));
      exit(1);
   }
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Board missing or defective
EDT_BADCMDERR	Invalid command parameter to driver
EDT_FWERR	Firmware returned an error
EDT_INVTS	Invalid time slot device handle
EDT_INVMSG	Invalid message
EDT_SH_BADLCLTS	Invalid local time slot number
EDT_SH_BADEXTTS	External time slot unsupported at current clock rate
EDT_SH_BADINDX	Invalid Switch Handler library index number
EDT_SH_BADMODE	Invalid Switch Handler bus configuration
EDT_SH_BADTYPE	Invalid local time slot type
EDT_SH_LCLTSCNCT	Local time slot is already connected to SCbus
EDT_SH_LIBBSY	Switch Handler library busy
EDT_SH_LIBNOTINIT	Switch Handler library is uninitialized
EDT_SH_MISSING	Switch Handler is not present
EDT_SH_NOCLK	Switch Handler clock fallback failed
EDT_SYSTEM	Windows NT system error
EDT_TMOERR	Timed out waiting for reply from firmware

See also

- **dt_unlisten( )**

In the *Voice Software Reference for Windows NT:*

- **ag_getxmitslot( )**
- **dx_getxmitslot( )**

Name:	int dt_mtfcn(devh,fncid,tmo)	
Inputs:	int devh	• Dialogic E-1 logical time slot device handle
	unsigned char fncid	• ID of the function to be performed
	unsigned int tmo	• Timeout value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous / asynchronous	

■ Description

The **dt_mtfcn( )** function initiates or stops the multitasking (asynchronous) function specified by the function ID (fncid) parameter for a DTI/212 time slot. This function is used to allow the application to wait for a line seizure on an E-1 line. The purpose of this function is to give the DTI/212 the ability to respond to a line seizure faster to ensure compatibility with faster switches. The firmware responds to the seizure with a seizure acknowledge.

The application previously needed to receive a signaling event indicating seizure and then use the signaling functions to acknowledge.

NOTE: The specified DTI/212 time slot must be in signaling insertion mode.

Parameter	Description
devh:	Specifies the valid DTI/212 logical time slot device handle returned by a call to **dt_open( )**.
fncid:	The fncid parameter specifies the function to initiate or the multitasking function to abort. This parameter can take either of the following values: • DEMT_WTCALL - transmit FREE line signaling defined by DTG_RXTXIDLE and respond to line seizure defined by DTG_SEIZESIG. • DEMT_ABORT - abort the multitasking function.
tmo:	Specifies the maximum amount of time in seconds that the function will block while awaiting a response from the DTI/212.

This function is dependent upon the following global parameters:

DTG_RXTXIDLE - used to set the receive and transmit idle patterns that must be present prior to waiting for a seizure. The upper byte represents the receive IDLE signaling pattern and the lower byte is the transmit FREE signaling pattern. Bits 0 to 3 represent transmit A, B, C, and D bits. Bits 8 to 11 represent receive A, B, C, and D bits. (OFF = 0 and ON = 1.)

DTG_SEIZESIG - used to set the receive SEIZE signaling pattern that defines a line seizure and the transmit BUSY signaling pattern to use for a response. Bits 0 to 3 represent transmit A, B, C, and D bits. Bits 8 to 11 represent receive A, B, C, and D bits. (OFF = 0 and ON = 1.)

This function should be called when valid signaling states are present on the line. Signaling states are configurable through the download parameter file (see the *System Release Software Installation Reference for Windows NT* for details). The following line signaling states are valid:

Receive	Transmit
IDLE	FREE
SEIZE	FREE
IDLE	BUSY

The function will return the error EDT_INVSIGST if using the DEMT_WTCALL **fncid** and invalid transmit or receive signaling states are on the line. The function will return EDT_NOWTCALL if using the DEMT_ABORT **fncid** and DEMT_WTCALL is not in progress.

It should be noted that DEMT_WTCALL automatically transmits the FREE pattern. To be safe, it's best to have the time slot transmitting BUSY to the network, then call the function with DEMT_WTCALL to transmit FREE and respond to a line seizure.

■ Asynchronous Mode

To operate this function in asynchronous (non-blocking) mode, specify 0 for **tmo**. Setting **tmo** to 0 allows the application to continue processing while awaiting a completion event from the device. If event handling is set up properly for your application, DTEV_MTFCNCPT is returned by the SRL **sr_getevttype( )** function when the multitasking function is successfully completed. See *Appendix A* for information on event handling.

■ Synchronous Mode

To run this function in synchronous (blocking) mode, set **tmo** to the desired length of time, in seconds, to await a return. If a response is not returned within **tmo** seconds, an error is returned. A suggested **tmo** setting for this function is -1, so the function will wait indefinitely for an incoming call.

■ Cautions

1. This function will fail under the following conditions:
   - The time slot is busy.
   - The specified DTI/212 time slot is not in signaling insertion mode (EDT_SIGINS).

- The specified DTI/212 logical time slot device handle is invalid.
- The specified time slot is not in the correct signaling state (EDT_INVSIGST)
- The function is used on a DTI/211 board or a D/240SC-T1 board.

2. To use this function in asynchronous mode, you must use the SRL **sr_enbhdlr( )** function to enable trapping of events and create an event handler to process the completion event returned by the device. The event can be detected by using the SRL event management functions. See *Appendix A* for more information on Digital Network Interface event management.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


#define IDLEFREE 0x0F07
#define SEIZEBUSY 0x0303
#define BUSY 0x03


main()
{
   int devh;                /* Board and time slot device handle */


   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( "dtiB1", 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Set the receive and transmit idle patterns before
    * waiting for a seizure
    */
   if ( dt_setparm ( devh, DTG_RXTXIDLE,  IDLEFREE ) == -1 ) {
      printf( "Error message = %s \n", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }


/*
    * Set the receive signaling pattern to watch
    * for while waiting for a call
    * and the transmit signaling pattern to use for a response
    */
   if ( dt_setparm ( devh, DTG_SEIZESIG,  SEIZEBUSY ) == -1 ) {
      printf( "Error message = %s \n", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
```

```
   /*
    * Close board 1 device
    */
   if ( dt_close( devh ) == -1 ) {
      printf( "Cannot close board dtiB1.  errno = %d", errno );
   }

   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( "dtiB1T1", 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Set signaling bits to a known state
    */
   if ( dt_settssigsim( devh, BUSY ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Execute the function in async mode, transmit FREE
    * and respond to seizure
    */

if ( dt_mtfcn ( devh, DEMT_WTCALL, 0 ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

   /*
    * Wait for DTEV_MTFCNCPT event
    */
               .
               .
               .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message (see *Appendix A* for more information). The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVSIGST	invalid signaling state
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOIDLEERR	time slot not in idle/closed state
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_NOWTCALL	not waiting for a call
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIGINS	signaling insertion not enabled
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

See also

- **dt_setparm( )**

Name:	int dt_open(name,oflags)	
Inputs:	char *name	• Digital Network Interface logical board or time slot device name
	int oflags	• open attribute flags; reserved for future use
Returns:	device handle if successful -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Resource Management	
Mode:	synchronous	

■ Description

The **dt_open( )** function opens a Digital Network Interface device and returns a unique Dialogic handle to identify the device. A device can be opened more than once by any number of processes. All subsequent references to an opened device must use the returned device handle.

NOTE: The device handle returned by this function is **Dialogic defined**. It is not a standard Windows NT file descriptor. Any attempts to use Windows NT operating system commands such as **read( )**, **write( )**, or **ioctl( )** will produce unexpected results.

Parameter	Description
name:	Points to an ASCIIZ string that contains the name of a valid Digital Network Interface logical board or time slot device.
oflags:	Reserved for future use. Set this parameter to 0.

NOTE: If a parent process opens a device and enables events, there is no guarantee that the child process will receive a particular event. It is recommended that you open devices in a parent process and enable events in a child process.

All Digital Network Interface logical boards and time slot devices can be opened with this function. Opening a Digital Network Interface device does not alter the state of the device. Opening or closing a Digital Network Interface device does not affect other processes using the device but a command can only be issued while the device is idle.

To avoid conflict between the DTI/ driver and the generic driver, follow the guidelines below when defining devices in the configuration files:

The name of the DTI/211 or DTI/212 device defined in /usr/dialogic/config/dticfg must be in the form dtiB*x* or dtiB*x*T*y* where:

x is the DTI/212 logical board device number (e.g. 1, 2, 3, ...)

y is the time slot number, beginning with 1 (e.g. 1, 2, ... 24 for T-1; 1, 2, ... 30 for E-1)

The name of the D/240SC-T1 or D/300SC-E1 device defined in /usr/dialogic/cfg/.voxcfg may be in the form dtiB*x*, dtiB*x*, dtiB*x*T*y*, or dtiB*x*T*y* where:

x is the D/240SC-T1 logical board device number (e.g. 1, 2, 3, ...)

y is the time slot number, beginning with 1 (e.g. 1, 2, ... 24)

The logical board device number of the D/240SC-T1 or D/300SC-E1 device must not be the same as the logical board device number of the DTI/211 or DTI/212 device. The devices are named dtiB*x* and dtiB*x*T*y* by default, but may be named dtiB*x* or dtiB*x*T*y* to allow backwards compatibility for previously designed applications.

Cautions

1. This function will fail under the following conditions:
   - The device name is not valid.
   - The device is already open.
   - The system has insufficient memory to complete the open.
2. For T-1 systems, time slot number must be in the range of 1 to 24.
3. For E-1 systems, time slot number must be in the range of 1 to 30.
4. Dialogic devices should **never** be opened using the Windows NT **open( )**.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                              /* Board device handle */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

               .
               .
               .
}
```

Errors

The **dt_open( )** function does not return errors in the standard Digital Network Interface return code format because it is a Windows NT system error. If an error occurs during the **dt_open( )** call, a -1 will be returned and the specific error message will be returned in the **errno** global variable. If a call to **dt_open( )** is successful, the return value is a valid handle for the open device.

See also

- **dt_close( )**

Name:	int dt_rundiag(devh,tmo,diagbufp)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned int tmo	• Timeout value
	char *diagbufp	• Pointer to 1 byte buffer for diagnostic code
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Diagnostic	
Mode:	synchronous/asynchronous	

■ Description

The **dt_rundiag( )** function runs diagnostics on the Network firmware. The function can operate in synchronous (blocking) or asynchronous (non-blocking) mode.

Parameter	Description
devh:	Specifies the valid Digital Network Interface board device handle returned by a call to **dt_open( )**.
tmo:	When operating the function in synchronous mode, specifies the length of time in seconds the function will block while waiting for a response from the device.
diagbufp:	Pointer to a one-byte data buffer to which the diagnostic code will be returned when the function is operating in synchronous mode.

Please note the following guidelines when using this function:

- This function can be issued at any time, but it is recommended that all time slots be idle and closed.
- This function is destructive to calls in progress.
- The board will be restored to its previous state; that is, the state the board was in before the function was called.

Parameter	Description

- The function should take about 5 seconds to complete.

■ Synchronous Mode

To operate the function in synchronous (blocking) mode, specify in **tmo** the length of time in seconds that the function will block. This causes the application to await a return from the function before performing any other processing. A suggested setting for **tmo** is 5.

■ Asynchronous Mode

To operate the function in asynchronous (non-blocking) mode, set **tmo** to 0. This allows the application to continue processing while awaiting a completion event from the device.

If event handling is set up properly for your application, DTEV_RETDIAG is returned by the SRL **sr_getevttype( )** function when the diagnostics are successfully completed.

To use this function in asynchronous mode, you must use the SRL **sr_enbhdlr( )** function to enable trapping of the event and create an event handler to process the completion event returned by the device. See *Appendix A* for more information on Digital Network Interface event management.

NOTE: To run this function in asynchronous operation, you must pass a NULL pointer to **diagbufp**.

Diagnostic Return Codes

The diagnostic codes listed below provide results of the diagnostics run on the Digital Network Interface firmware. In synchronous mode, the diagnostic codes are returned to the one-byte buffer pointed to by **diagbufp**. In asynchronous mode, the codes are returned by the SRL **sr_getevtdatap( )** function.

D2DE_BRDCFG	• Invalid board configuration data
D2DE_INVEE	• Invalid EEPROM data (not valid for D/240SC-T1)
D2DE_LIUFAIL	• Read/write to LIU failed
D2DE_MEMTST	• Memory test failed
D2DE_NOERR	• No errors
D2DE_ROMCHK	• Bad ROM checksum (not valid for D/240SC-T1)
D2DE_XCVRFAIL	• Read XCVR register failed

■ Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical board device handle is specified.
   - There is a firmware/hardware problem on the device.
2. Make sure all time slots are closed and idle. This function is destructive to calls in progress.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


main()
{
   int devh;                    /* Board device handle */
   int retval;                  /* Return value from function call */
   char diagbufp;               /* Diagnostic buffer */
   /*
    * Open board 1 device
    */


   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Run diagnostics on the board with a 5 second timeout.
    */
   if ( ( retval = dt_rundiag( devh, 5, &diagbufp ) ) == -1 ) {
      printf("Error activating diag tests: error message = %S\n",
                 ATDV_ERRMSGP( devh ) );
```

```
   }
   if ( diagbufp ! = DTDE_NOERR )
      printf( "Diagnostic buffer value = %d\n", diagbufp );
      exit( 1 );
   }
            .
            .
            .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out.
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZEERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	indicates Windows NT system error. Look at global variable **errno** for actual error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **dt_tstcom( )**
- **dt_tstdat( )**

Name:	int dt_setalrm(devh, mode)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned int mode	• alarm handling mode
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Alarm	
Mode:	synchronous	

■ Description

The **dt_setalrm( )** function sets the Digital Network Interface device to one of three alarm handling modes. The alarm handling mode determines how the Digital Network Interface device and the application interact to perform T-1 or E-1 alarm handling. For more information on alarm handling, see *Chapter 2. Digital Network Interface Telephony* .

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
mode:	Specifies one of three alarm handling modes: • DTA_NONE - no firmware controlled alarm handling. All alarm handling must be controlled by the application. • DTA_TERM (default) - terminate alarm handling mode. Alarms in terminate configuration are handled automatically by device firmware. In terminate alarm handling mode, a red alarm will cause the automatic transmission of a yellow alarm. • DTA_DROP (except DTI/212 or D/300SC-E1) - drop-and-insert alarm handling mode. Alarm handling duties are shared by application and DTI/211 or D/240SC-T1 board firmware. In this mode, alarm transmission responsibilities are left to the application.

Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical board device handle is specified.
   - The specified mode is invalid.
2. The DTA_DROP parameter is not supported by the DTI/212 device. Using this parameter with a DTI/212 device will produce an error. For DTI/212 devices, use the DTA_NONE or DTA_TERM alarm handling mode.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                            /* Board device handle */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Set alarm mode to terminate
    */
   if ( dt_setalrm( devh, DTA_TERM ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

               .
               .
               .
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to

obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_BADVAL	invalid mode passed in parameter
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable errno for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file dtilib.h.

■ See also

- **dt_xmitalrm( )**

Name:	dt_setevtmsk(devh,event,bitmask,action)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board or Digital Network Interface logical time slot device handle
	int event	• event to be enabled/disabled
	unsigned short bitmask	• bitmask for events
	int action	• set, add, or subtract bitmask
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Parameter Setting	
Mode:	synchronous	

■ Description

The **dt_setevtmsk( )** function enables and disables notification for events that occur on a Digital Network Interface logical board or time slot device. This function allows the application to set and alter a bitmask of transition events. The bitmask determines which transitions will cause an event to be generated.

The event can be retrieved by using the event management functions included in the Standard Runtime Library (refer to *Appendix A* for more information on the SRL). The current bitmask can be examined by using the **dt_getevtmsk( )** function.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board or Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
event:	Specifies the type of event to be enabled or disabled on the device specified by **devh**: • DTG_T1ERREVT - T-1 error events (DTI/211 and D/240SC-T1 logical board device handles only). Several T-1 error events can be monitored. Specific T-1 error events are enabled or disabled by setting the

Parameter	Description
	bitmask parameter. • DTG_E1ERREVT - E-1 error events (DTI/212 or D/300SC-E1 logical board device handles only). Several E-1 error events can be monitored. Specific E-1 error events are enabled or disabled by setting the **bitmask** parameter. • DTG_SIGEVT - Signaling bit transition events (time slot device handles only). Specific signaling events are enabled or disabled by setting the **bitmask** parameter. • DTG_PDIGEVT - pulse digit events (D/240SC-T1 or D/300SC-E1 time slot device handles only). **NOTE:** For D/240SC-T1 and D/300SC-E1 products, you must enable both the ON and OFF transitions on a speciifed bit to get events on that bit. For example, AON and AOFF must be enabled to detect events on the A bit.
bitmask:	Specifies the event to be enabled or disabled by setting the bitmask for that event. Multiple transition events may be enabled or disabled with one function call if the bitmask values are logically ORed together. The **bitmask** values for each **event** parameter are described in *Table 5*, found in the **dt_getevtmsk( )** function description.
action:	Specifies how the signaling bit transition event mask is changed. Events can be added to or subtracted from those specified in **bitmask**, or events can replace the existing ones. The possible values for the **action** parameter are: • DTA_SETMSK - enable notification of events specified in **bitmask** and disable notification of previously set events. • DTA_ADDMSK - enable notification of events specified in **bitmask** in addition to previously set

Parameter	Description
	events. (Not valid for DTG_PDIGEVT.) • DTA_SUBMSK - disable notification of events specified in **bitmask**.

For example, to enable event notification:

1. Specify the events to enable in the **bitmask** field.
2. Specify the DTA_SETMSK bitmask in the **action** field.

This enables notification of the events specified in the **bitmask** parameter and disables notification of previously set events.

To enable an additional event:

1. Specify the events in **bitmask**.
2. Specify DTA_ADDMSK in the **action** field.

This adds the notification of events specified in **bitmask** without disabling the currently enabled events.

To disable events, use the following procedure:

1. Specify the events in **bitmask**.
2. Specify DTA_SUBMSK in the **action** field.

This disables the event in **bitmask** without disabling any other events.

To disable all currently enabled events:

1. Specify 0 in **bitmask**.
2. Specify DTA_SETMSK in the **action** field.

■ Event Notification and Handling

NOTE: Event handling operations vary with the mode type (i.e., callback, polled, synchronous, etc.) used by your application. For more information on application development models, refer to the *Standard Runtime Library*

Programmer's Guide for Windows NT (part of the *Voice Software Reference for Windows NT*).

To trap and handle a specified Digital Network Interface event, follow these steps in the order listed:

- Call **sr_enbhdlr( )**. This function specifies the event and the application defined event handler that is called when this event occurs.
- Call **dt_setevtmsk( )**. This specifies the list of events for which the application should be notified.

NOTE: When the DTG_T1ERREVT, DTG_E1ERREVT, or DTG_SIGEVT event is generated, call the **sr_getevtdatap( )** function in the event handler to get a pointer to the event value. The pointer should be cast to an unsigned short pointer and the event retrieved as an unsigned short value.

Refer to *Appendix A* for more information on SRL data structures and functions.

■ Cautions

1. This function will fail under the following conditions:
   - An invalid time slot or an invalid Digital Network Interface logical board device handle is specified.
   - The **event** specified is invalid.
   - The **action** specified is invalid.
2. For the application to process an event, the SRL **sr_enbhdlr( )** Event Management function should be called prior to calling the **dt_setevtmsk( )** function.
3. When a wink event occurs, the signaling bits associated with the wink will be reported to the application. Therefore, your application's signaling event handlers must make sure that *any* transition of the selected wink signaling bit is not part of a wink event.

■ Example

```
#include <windows.h>
#include <srllib.h>
```

```
#include <dtilib.h>
#include <errno.h>

main()
{

   int devh;                                /* Time slot device handle */
   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open device dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Enable an event handler to catch AON and AOFF events
    */
              .
              .
              .
   /*
    * Enable AON and AOFF signaling transition events
    */
   if ( dt_setevtmsk(devh, DTG_SIGEVT, DTMM_AON | DTMM_AOFF, DTA_SETMSK )
      ts

== -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

              .
              .
              .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board

	device handle
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

In this guide:

- **dt_getevtmsk( )**

In *Appendix A*:

- **sr_enbhdlr( )**
- **sr_dishdlr( )**

Name:	int dt_setidle(devh,state)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned int state	• idle state of time slot
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Audio	
Mode:	synchronous	

■ Description

The **dt_setidle( )** function enables or disables transmission of silence to the network for the audio portion of the specified time slot. Transmitting silence is referred to as “idling“ or “inserting idle” on a time slot.

When two Digital Network Interface boards are arranged in drop-and-insert configuration, this function can be used to disable pass-through operation. Transmitting idle overrides voice data being passed between Dialogic network devices on the selected time slot.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
state:	Specifies whether to enable or disable the transmission of silence. The possible values are: • DTIS_DISABLE - disable idling on the time slot • DTIS_ENABLE - enable idling on the time slot

The default idle value transmitted is 7FH (T-1 only) or 54H (E-1 only). We recommend you initialize the device idle value to a known state before idling a time slot. The device idle value is set using the **dt_setparm( )** function with the parameter DTG_IDLTYP. The values of this parameter can be set as follows:

- IDLE_7F - sets idle value to 7FH (T-1 only)
- IDLE_FF - sets idle value to FFH (T-1 only)
- IDLE_54 - sets idle value to 54H (E-1 only)
- IDLE_D5 - sets idle value to D5H (E-1 only)

Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical time slot device handle is specified.
   - The state specified is invalid.
2. If the signaling mode is set to transparent for a time slot on a T-1 system (one using a DTI/211 or D/240SC-T1 board), the time slot is idled, and the idle pattern is FF, then the signaling data for that time slot will be overwritten with ones. Before idling a T-1 time slot, the time slot should be set to signaling insertion mode.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                              /* Time slot device handle */
   /*
    * Open time slot 1 on board 1
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }
   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Disable silence transmission
    */
   if ( dt_setidle( devh, DTIS_DISABLE ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
```

```
   /*
    * Go offhook
    */
   if ( dt_settssig( devh, DTB_ABIT | DTB_BBIT, DTA_SETMSK ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
               .
               .
               .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_IDLEST( )**

- **dt_setsigmod( )**

Name:	int dt_setparm(devh,param,valuep)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned long param	• device parameter defined name
	void* valuep	• pointer to device parameter value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Parameter Setting	
Mode:	synchronous	

■ Description

The **dt_setparm( )** function changes the value of a device parameter.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
param:	Specifies the parameter value to alter.
valuep:	Specifies the address of the integer containing the value to be assigned to the parameter.

All time slots on the selected Digital Network Interface device must be closed when this function is called.

Table 6, found in the **dt_getparm( )** function description, lists each parameter name, its default value, and a brief description.

■ Cautions

1. This function will fail under the following conditions:

- An invalid Digital Network Interface logical board device handle is specified.
- One or more time slots on the Digital Network Interface device are open.
- The parameter specified is invalid.
- The Digital Network Interface is in test mode (remote loopback switch set to ON) and DTG_SETBDMD is passed in the **param** field.

2. Changing a Digital Network Interface device parameter affects all the time slots on the logical board. All the time slots on a logical board must be closed when device parameters are altered.

3. All values of the parameter have to be integers, but since this routine expects a void pointer to **valuep**, the address must be cast as a void*.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


main()
{
   int devh;                              /* Board device handle */
   int valuep;                            /* Parameter value */
   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Set current clock parameter value
    */
   valuep = DTC_EXT;
   if ( dt_setparm( devh, DTG_SETCLK, ( void * )&valuep ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
               .
               .
               .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_BADGLOB	invalid **param**
EDT_BADVAL	invalid parameter value passed in **valuep** pointer
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOCLK	no clock source present
EDT_NOIDLEERR	time slot not in idle/closed state
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware
EDT_TSTMOD	in test mode; cannot set Digital Network Interface mode

Error defines can be found in the file *dtilib.h*.

■ See also

- **dt_getparm( )**

Name:	int dt_setsigmod(devh,mode)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned int mode	• transmit mode
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous	

■ Description

The **dt_setsigmod( )** function sets the type of signaling that will be performed on the transmitted time slot.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
mode:	Specifies the transmit mode. Possible values are: • DTM_TRANSP - set to transparent signaling • DTM_SIGINS - set to signaling insertion

■ Transparent Signaling

When a time slot is set to transparent, transmit signaling for the selected Digital Network Interface time slot originates at a compatible resource device; for example a D/12x. The Digital Network Interface has no control over signaling information from the resource device in transparent signaling mode.

NOTES: 1. To initiate a wink from a Voice device channel using the Voice library function **dx_wink( )**, the Digital Network Interface time slot attached to that channel MUST be in transparent signaling mode.

2. The DTI/212 board does not support transparent signaling when used in a drop-and-insert configuration.

■ Signaling Insertion

When a time slot is set to signaling insertion, transmit signaling for the selected time slot is inserted by the Digital Network Interface. The Digital Network Interface can insert signaling information over the transmit signaling already on that time slot.

■ Cautions

This function will fail under the following conditions:

- An invalid Digital Network Interface logical time slot device handle is specified.
- The mode specified is invalid.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;                                /* Time slot device handle */

   /*
    * Open time slot 1 on board 1
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }

   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

            .
            .
```

```
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

See also

- **ATDT_BDMODE( )**
- **ATDT_BDSIGBIT( )**
- **ATDT_TSMODE( )**
- **ATDT_TSSGBIT( )**
- **dt_settssig( )**

Name:	int dt_settssig(devh,bitmask,action)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned short bitmask	• signaling bits to change
	int action	• set, add, or subtract bitmask
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous	

■ Description

The **dt_settssig( )** function sets or clears the transmit for the time slot requested.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
bitmask:	Specifies which signaling bits to change. All signaling bits may be changed with one function call if the bitmask values are logically ORed together as in the example. The possible values for the **bitmask** parameter are: • DTB_ABIT - A signaling bit • DTB_BBIT - B signaling bit • DTB_CBIT - C signaling bit (E-1 only) • DTB_DBIT - D signaling bit (E-1 only)
action:	Specifies whether the signaling bits in the mask should be set or cleared, (i.e. set to one or set to zero). The possible values are: • DTA_SETMSK - set bits specified in **bitmask** and clear all other bits. (Not valid for a DTI/101 device.) • DTA_ADDMSK - set bits specified in **bitmask**. This will not affect other bits that are currently set. • DTA_SUBMSK - clear bits in specified **bitmask**. This will not affect other bits that are currently set.

Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical time slot device handle is specified.
   - The action specified is invalid.
2. On a DTI/212, do not set all signaling bits (A, B, C, and D) to 0 (0000). A setting of four zeros in the signaling bits is used to provide multiframe synchronization by identifying frame 0.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   /*
    * Open time slot 1 on board 1
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }
   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Go offhook
    */
   if ( dt_settssig( devh, DTB_ABIT | DTB_BBIT, DTA_SETMSK ) == -1 ) {
       printf( "Error message = %s.", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
            .
            .
            .
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to

obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDMODE( )**
- **ATDT_BDSIGBIT( )**
- **ATDT_TSMODE( )**
- **ATDT_TSSGBIT( )**
- **dt_setsigmod( )**

Name:	int dt_settssigsim(devh,bitmask)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned short bitmask	• signaling bits to simultaneously clear and set
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous	

■ Description

The **dt_settssigsim( )** allows simultaneous setting or clearing of the transmit signaling bits on a Digital Network Interface time slot. The bitmask parameter specifies which signaling bits to change. To simultaneously set and clear the transmit signaling bits, the chosen values can be logically ORed together.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
bitmask:	Specifies which signaling bits to change. All signaling bits may be changed with one function call if the bitmask values are logically ORed together as in the example. The possible values for the **bitmask** parameter are: • DTB_AON - A signaling bit on • DTB_AOFF - A signaling bit off • DTB_BON - B signaling bit on • DTB_BOFF - B signaling bit off • DTB_CON - C signaling bit on (E-1 only) • DTB_COFF - C signaling bit off (E-1 only) • DTB_DON - D signaling bit on (E-1 only) • DTB_DOFF - D signaling bit off (E-1 only)

Parameter	Description
	All signaling bits may be changed with one function call if the bitmask values are ORed together.

■ Cautions

1. This function will fail if an invalid Digital Network Interface logical time slot device handle is specified.
2. Do not set all signaling bits (A, B, C, and D) to 0 (0000) on a DTI/212 time slot. A setting of four zeros in the signaling bits is used to provide multiframe synchronization by identifying frame 0.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   /*
    * Open time slot 1 on board 1
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }
   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Set A & C time slot bits while clearing the B bit simultaneously
    * the D bit is left untouched
    *  /
   bitmask = DTB_AON | DTB_CON | DTB_BOFF;

if ( dt_settssigsim( devh, bitmask) == -1) |{
      printf( "Error message = %s.", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }            .
              .
              .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **ATDT_BDMODE( )**
- **ATDT_BDSIGBIT( )**
- **ATDT_TSMODE( )**
- **ATDT_TSSGBIT( )**
- **dt_setsigmod( )**

Name:	int dt_tstcom(devh,tmo)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned int tmo	• timeout value
Returns:	Digital Network Interface return code -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Diagnostic	
Mode:	synchronous/asynchronous	

■ Description

The **dt_tstcom( )** function tests the ability of a Digital Network Interface device to communicate with the host PC. This function can operate in either synchronous (blocking) or asynchronous (non-blocking) mode.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
tmo:	Specifies the maximum amount of time in seconds that the function will block while waiting for a response from the Digital Network Interface. If a response is not returned within **tmo** seconds, an error is returned.

Please note the following guidelines when using this function:

- This function can be issued at any time, but it is recommended that all time slots be idle and closed.
- This function has no effect on calls in progress.
- This function has no effect on the state of the board.

Synchronous Mode

To run this function in synchronous (blocking) mode, set **tmo** to the length of time, in seconds, to await a return. If a response is not returned within **tmo** seconds, an error is returned. A suggested **tmo** setting for this function is 5.

Asynchronous Mode

To operate this function in asynchronous (non-blocking) mode, specify 0 for **tmo**. This allows the application to continue processing while awaiting a completion event. If event handling is set up properly for your application, DTEV_COMRSP will be returned by the **sr_getevttype( )** function included in the SRL when the test is successfully completed. See *Appendix A* for information on event handling.

Cautions

1. This function returns a failure under the following conditions:

   - The specified device fails to respond within **tmo** seconds, if operating in synchronous mode.
   - A time slot or invalid Digital Network Interface logical device handle is specified.
   - There is a hardware problem on the Digital Network Interface.
   - There is a configuration problem (for example, IRQ conflict).

   NOTE: Device configuration information is found in the appropriate hardware installation card (see *Appendix B*).

2. To use this function in asynchronous mode, you must use the SRL **sr_enbhdlr( )** function to enable trapping of events and create an event handler to process the completion event returned by the device.

   The event can be detected by using the new event management functions included in the new release of the Standard Runtime Library. See *Appendix A* for more information on Digital Network Interface event management.

Example

```
#include <windows.h>
#include <srllib.h>
```

```
#include <dtilib.h>
#include <errno.h>

main()
{

   int devh;                          /* Board device handle */
   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Test the board's ability to communicate with the system.  Give it 5
    * seconds to complete.
    */
   if ( dt_tstcom( devh, 5 ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

               .
               .
               .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter

EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

See also

- **dt_tstdat( )**
- **dt_rundiag( )**

Name:	int dt_tstdat(devh,tmo)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned int tmo	• timeout value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Diagnostic	
Mode:	synchronous/asynchronous	

■ Description

The **dt_tstdat( )** function performs a test that verifies the integrity of the Digital Network Interface I/O interface to the PC. The data test is performed by sending a series of bytes to the Digital Network Interface and checking the integrity of the bytes returned.

Please note the following guidelines when using this function:

- This function can be issued at any time, but it is recommended that all time slots be idle and closed.
- This function has no effect on calls in progress.
- This function has no effect on the state of the board.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
tmo:	Specifies the maximum amount of time in seconds that the function will block while awaiting a response from the Digital Network Interface.

■ Asynchronous Mode

To operate this function in asynchronous (non-blocking) mode, specify 0 for **tmo**. This allows the application to continue processing while awaiting a completion event. If event handling is set up properly for your application, DTEV_DATRSP will be returned by the **sr_getevttype( )** function included in the SRL when the test is successfully completed. See *Appendix A* for information on event handling.

■ Synchronous Mode

To run this function in synchronous (blocking) mode, set **tmo** to the length of time, in seconds, to await a return. If a response is not returned within **tmo** seconds, an error is returned. A suggested **tmo** setting for this function is 5.

■ Cautions

1. This function will return a failure if:
   - The test data is corrupted.
   - A time slot or invalid Digital Network Interface logical board device handle is specified.
2. To use this function in asynchronous mode, you must use the SRL **sr_enbhdlr( )** function to enable trapping of events and create an event handler to process the completion event returned by the device. The event can be detected by using the SRL event management functions. See *Appendix A* for more information on Digital Network Interface event management.

■ Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Board device handle */

   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }

   /*
    * Perform a data integrity test between the board and PC.  Give it 5
    * seconds to complete.
    */
   if ( dt_tstdat( devh, 5 ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }

               .
               .
               .
}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver

EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*

See also

- **dt_tstcom( )**
- **dt_rundiag( )**

Name:	dt_unlisten(devh)	
Inputs:	int devh	• D/240SC-T1 or D/300SC-E1 Digital network interface device time slot
Returns:	0 on success -1 on error	
Includes:	srllib.h dtilib.h	
Category:	SCbus routing	
Mode:	Synchronous	

■ Description

The **dt_unlisten( )** function disconnects the receive of a digital network interface device time slot on a D/240SC-T1 or D/300SC-E1 board from the SCbus time slot.

Calling the **dt_listen( )** function to connect to a different SCbus time slot will automatically break an existing connection. Thus, when changing connections, you need not call the **dt_unlisten( )** function.

NOTE: The SCbus convenience funtion **nr_scunroute( )** includes **dt_getxmitslot( )** functionality; see the *Voice Programmer's Guide for Windows NT*.

Parameter	Description
devh:	Specifies the valid digital network interface time slot device handle returned by a call to **dt_open( )**.

■ Cautions

This function will fail under the following conditions:

- An invalid time slot device handle is specified.
- If called to disconnect a PEB time slot.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>


main( )
{
   int devh;             /* Digital channel (time slot) device handle */

   /* Open board 1 time slot 1 device */
   if ((devh = dt_open("dtiB1T1", 0)) == -1) {
      printf("Cannot open time slot dtiB1T1.  errno = %d", errno);
      exit(1);
   }

   /*
    * Disconnect receive of board
    * 1, time slot 1 from all
    * SCbus time slots
    */
   if (dt_unlisten(devh) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(devh));
      exit(1);
   }
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Board missing or defective
EDT_BADCMDERR	Invalid command parameter to driver
EDT_FWERR	Firmware returned an error
EDT_INVTS	Invalid time slot device handle
EDT_INVMSG	Invalid message
EDT_SH_BADLCLTS	Invalid local time slot number
EDT_SH_BADEXTTS	External time slot unsupported at current clock rate
EDT_SH_BADINDX	Invalid Switch Handler library index number
EDT_SH_BADMODE	Invalid Switch Handler bus configuration

Equate	Returned When
EDT_SH_BADTYPE	Invalid local time slot type
EDT_SH_LCLDSCNCT	Local time slot is already disconnected from SCbus
EDT_SH_LIBBSY	Switch Handler library busy
EDT_SH_LIBNOTINIT	Switch Handler library is uninitialized
EDT_SH_MISSING	Switch Handler is not present
EDT_SH_NOCLK	Switch Handler clock fallback failed
EDT_SYSTEM	Windows NT system error
EDT_TMOERR	Timed out waiting for reply from firmware

See also

- **dt_listen( )**

Name:	int dt_xmitalrm(devh, alrmtype, state)	
Inputs:	int devh	• Dialogic Digital Network Interface logical board device handle
	unsigned char alrmtype	• T-1 or E-1 alarm type
	unsigned int state	• enable or disable sending the alarm
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Alarm	
Mode:	synchronous	

■ Description

The **dt_xmitalrm( )** function starts and stops transmission of an alarm to a network span. For a detailed description of T-1 and E-1 alarm handling, refer to *Chapter 2. Digital Network Interface Telephony* .

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical board device handle returned by a call to **dt_open( )**.
alrmtype:	Specifies the T-1 or E-1 alarm type to be transmitted: • YELLOW - T-1 only • BLUE - T-1 only • DEA_REMOTE - E-1 only • DEA_UNFRAMED1 (unframed all 1s) - E-1 only • DEA_SIGNALALL1 (signaling all 1s) - E-1 only • DEA_DISTANTMF (distant multiframe alarm) - E-1 only
state:	Specifies whether to enable or disable transmission of the specified alarm: • DTIS_DISABLE - disable transmission of alarm • DTIS_ENABLE - enable transmission of alarm

Cautions

1. This function will fail under the following conditions:
   - The specified Digital Network Interface device is invalid.
   - The specified **alrmtype** parameter is invalid.
   - The specified **state** parameter is invalid.
2. Transmission of alarms requires that the proper alarm mode is set by the **dt_setalrm( )** function.
3. The alarm type transmitted must correspond to the type of network circuit you are using (either T-1 or E-1).

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Board device handle */
   /*
    * Open board 1 device
    */
   if ( ( devh = dt_open( “dtiB1”, 0 ) ) == -1 ) {
      printf( "Cannot open board dtiB1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Transmit a BLUE alarm
    */
   if ( dt_xmitalrm( devh, BLUE, DTIS_ENABLE ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
            .
            .
            .
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVBD	invalid Digital Network Interface logical board device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

■ See also

- **dt_setalrm( )**

Name:	int dt_xmittone(devh,state)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned int state	• enable/disable test tone
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Audio	
Mode:	synchronous	

Description

The **dt_xmittone( )** function enables or disables transmission of a test tone to the PEB on the specified DTI/211 or D/240SC-T1 time slot. The digital milliwatt test tone can be used to test for proper connections between DTI/211 or D/240SC-T1 boards and the PEB module.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
state:	Specifies tone on or off. The possible values are: • DTIS_DISABLE - disable tone generation • DTIS_ENABLE - enable tone generation

Cautions

1. This function will fail under the following conditions:
   - An invalid Digital Network Interface logical time slot device handle is specified.
   - The state specified is invalid.
2. This function is not supported by the DTI/212 or D/300SC-E1 device. Using the **dt_xmittone( )** function will produce an error.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   /*
    * Open board 1 time slot 1 device
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open time slot dtiB1T1.  errno = %d", errno );
      exit( 1 );
   }
   /*
    * Start transmitting a test tone on this time slot
    */
   if ( dt_xmittone( devh, DTIS_ENABLE ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
               .
               .
               .
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. See *Appendix A* for more information on SRL functions. The error codes returned by **ATDV_LASTERR( )** are:

Equate	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_RANGEERR	bad/overlapping physical memory range

EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware

Error defines can be found in the file *dtilib.h*.

Name:	int dt_xmitwink(devh, tmo)	
Inputs:	int devh	• Dialogic Digital Network Interface logical time slot device handle
	unsigned int tmo	• timeout value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h	
Category:	Time Slot Signaling	
Mode:	synchronous/asynchronous	

■ Description

The **dt_xmitwink( )** function transmits wink signaling to the T-1 or E-1 network span on any of the available signaling bits. The bit to be used and the polarity or beginning state of the wink are configurable through the download parameter file (see the *System Release Software Installation Reference for Windows NT* for details). A wink starts by transmitting signaling state 0, then transmits signaling state 1, and returns to signaling state 0. The signaling bit selected must be in the proper state (state 0) when the **dt_xmitwink( )** function is called. Also, the time slot must be in signaling insertion mode to transmit a wink.

Board parameters may be set through **dt_setparm( )** to control prewink delay and transmit wink duration for all time slots simultaneously.

NOTE: Separate board parameters are provided for setting minimum and maximum receive wink duration. These have no effect on wink transmission.

Parameter	Description
devh:	Specifies the valid Digital Network Interface logical time slot device handle returned by a call to **dt_open( )**.
tmo:	Specifies the maximum amount of time in seconds that the function will block while awaiting a response from the Digital Network Interface.

■ Asynchronous Mode

To operate this function in asynchronous (non-blocking) mode, specify 0 for **tmo**. This allows the application to continue processing while awaiting a completion event from the device. If event handling is set up properly for your application, DTEV_WINKCPLT is returned by the SRL **sr_getevttype( )** function when the wink is successfully completed. See *Appendix A* for information on event handling.

■ Synchronous Mode

To run this function in synchronous (blocking) mode, set **tmo** to the length of time, in seconds, to await a return. If a response is not returned within **tmo** seconds, an error is returned. A suggested **tmo** setting for this function is 2.

■ Cautions

1. This function will fail under the following conditions:
   - The specified Digital Network Interface logical time slot device handle is invalid.
   - The specified time slot is not in the correct signaling state (must begin in state 0).
   - Signaling insertion is not enabled for the specified time slot device.
   - A T-1 system (DTI/211 or D/240SC-T1 board) is configured for wink transmission using the C or D bit.
   - An application attempts to change signaling mode or signaling bits while wink transmission is in progress.
2. To use this function in asynchronous mode, you must use the SRL **sr_enbhdlr( )** function to enable trapping of events and create an event handler to process the completion event returned by the device. The

event can be detected by using the SRL event management functions. See *Appendix A* for more information on Digital Network Interface event management.

Example

```
#include <windows.h>
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

main()
{
   int devh;         /* Time slot device handle */
   /*
    * Open time slot 1 on board 1
    */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Failed to open device dtiB1T1.  errno = %d\n", errno );
      exit( 1 );
   }
   /*
    * Set signaling bits to a known state
    */
   if ( dt_settsig ( devh, DTB_ABIT | DTB_BBIT, DTA_SUBMSK ) == -1 ) {
      printf( "Error message = %s \n", ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Set signaling mode to signaling insertion
    */
   if ( dt_setsigmod( devh, DTM_SIGINS ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Disable silence transmission
    */
   if ( dt_setidle( devh, DTIS_DISABLE) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Go offhook assuming that wink set to negative polarity on A bit
    */
   if ( dt_settssig( devh, DTB_ABIT, DTA_SETMSK ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
   /*
    * Transmit wink with 2 second timeout.  Note that this is the blocking
    * (synchronous) mode
    */

if ( dt_xmitwink( devh, 2 ) == -1 ) {
      printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
      exit( 1 );
   }
            .
```

```
                .
                .
}
```

Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message (see *Appendix A* for more information). The error codes returned by **ATDV_LASTERR( )** are:

Equates	Returned When
EDT_BADBRDERR	Digital Network Interface missing or defective
EDT_BADCMDERR	invalid or undefined command to driver
EDT_DATTO	data reception timed out
EDT_FWERR	firmware returned an error
EDT_INVTS	invalid Digital Network Interface logical time slot device handle
EDT_NOMEMERR	cannot map or allocate memory in driver
EDT_PARAMERR	invalid parameter
EDT_PDOFFHK	wink bit not in correct initial state
EDT_RANGEERR	bad/overlapping physical memory range
EDT_SIGINS	signaling insertion not enabled
EDT_SIZERR	message too big or too small
EDT_SKIPRPLYERR	a required reply was skipped
EDT_SYSTEM	Windows NT system error. Check the global variable **errno** for more information about the error.
EDT_TMOERR	timed out waiting for reply from firmware
EDT_WKACT	already transmitting wink
EDT_WKSIG	cannot disable insertion when transmitting wink

Error defines can be found in the file *dtilib.h*.

See also

- **dt_setparm( )**

- **dt_setsigmod( )**
- **dt_settssig( )**

5. Digital Network Interface Application Guidelines

Digital Network Application Overview

This chapter offers advice and suggestions to guide programmers in designing and coding a Dialogic Digital Network Interface application for Windows NT.

NOTE: In the context of this guide, "Digital Network Interface" is used to refer to the DTI/211 board, the DTI/212 board, the D/240SC-T1 board or the D/300SC-E1 board unless otherwise noted.

5.1. Writing a Simple Digital Network Interface Application

This chapter is not meant to be a comprehensive guide to developing or debugging Digital Network Interface applications. Instead, the following sections provide Digital Network Interface general and task-specific programming guidelines:

- General Guidelines
- Initialization
- Processing
- Terminating
- Compiling and Linking
- Aborting

5.1.1. General Guidelines

The following general guidelines for writing Dialogic applications are explained in this section.

- Use symbolic defines
- Include header files
- Check return codes

Use Symbolic Defines

Dialogic does not guarantee that the numerical values of defines will remain the same as new versions of a software package are released. In general, do not use a numerical value in your application when an equivalent symbolic define is available.

Include Header Files

Various header files must be included in your application to test for error conditions, to use other library functions from the System Release Development Package, or to perform event management and standard attribute functions. An example is shown below. See *Section 3.3.* Include Files, for details.

```
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>
```

NOTE: To avoid redundancy in the remaining programming examples in this chapter, **#include** statements will not be shown.

Check Return Codes

Most Digital Network Interface Windows NT library functions return a value of -1 if they fail (extended attribute functions return AT_FAILURE or AT_FAILUREP if they fail). Any call to a Digital Network Interface library function should therefore check for a return value indicating an error. This can be done using a format similar to the following:

```
/* call to Dialogic DTI/xxx Library function */

if (dt_xxx(arguments) == -1) {
          /* error handling routine */
}
/* successful function call -
          continue processing ...  */
```

Using this technique ensures that all errors resulting from a Digital Network Interface device Windows NT library call will be trapped and handled properly by the application. In many cases, you can check for a return value of other than zero (0), as shown in the example below. However, this should only be used where a nonzero value is returned when the function fails. See *Section 3.2.* Digital

Network Interface Error Handling, or *Chapter 4. Digital Network* Interface Function Reference, for function specific details.

```
        .
           .
           .
   /* error handling routine */
   void do_error( devh, funcname )
      int devh;
      char *funcname;
   {
      int errorval = ATDV_LASTERR( devh );
      printf( "Error while calling function %s on device %s.  \n", funcname,
         ATDV_NAMEP( devh ) );
      if ( errorval == EDT_SYSTEM ) {
         printf( "errno = %d\n", errno );
         perror("");
      } else {
         printf( "Error value = %d\n Error message = %s\n",
            errorval,ATDV_ERRMSGP( devh ) );
      }
      return;
   }

   main( )
   {
            .
            .
            .

   /* call to Dialogic DTI/xxx library function */
      if (dt_setevtmsk( devh, DTG_SIGEVT, 0, DTA_SETMSK ) ) != 0) {
         do_error( devh, "dt_setevtmsk()" );
      }

   /* successful function call -
         continue processing ...  */
               .
               .
               .

}
```

NOTES: 1. Calls to **dt_open( )** return either -1 or a nonzero device handle. Therefore, when issuing the **dt_open( )** function, check for a return of -1. The specific error can be found in the global variable **errno**, contained in *errno.h*. Calls to **ATDT_BDSGBIT( )** return the pointer **AT_FAILUREP** when the function fails.

2. To avoid redundancy in the remaining programming examples in this chapter, the **do_error( )** function will not be shown.

The *dtilib.h* header file lists Windows NT library symbolic defines.

5.1.2. Initialization

Before a Digital Network Interface application can perform any processing or access devices, it should initialize the Digital Network Interface hardware to reflect the physical configuration of your system and set other parameters needed to support the application. Tasks that are performed as a part of initialization generally include:

- Set hardware configuration
- Set alarm handling parameters and masks
- Initialize time slots

These involve the following Digital Network Interface device Windows NT functions:

- **dt_setalrm( )**
- **dt_setevtmsk( )**
- **dt_setidle( )**
- **dt_setparm( )**
- **dt_setsigmod( )**
- **dt_settssig( )**

NOTE: Preferably, parameters set by **dt_setparm( )** are those that must be changed "on the fly" or that cannot be set through the download parameter file (see the *System Release Software Installation Reference for Windows NT*).

Set Hardware Configuration

Use **dt_setparm( )** to set hardware configuration, test mode, clock source, and network telephony parameters. Specific settings include:

- cable type connecting the Digital Network Interface device to the network
- loopback test mode
- clock source (see **dt_setparm( )** in *Chapter 4. Digital Network Interface Function Reference* for an example)
- wink detection and transmission duration

NOTE: If your application uses the **dt_xmitwink( )** function for receipt of Automatic Number Identification (ANI) or Direct Number Identification Service (DNIS) digits, you must make sure that Digital Network Interface wink duration conforms to the proper protocol requirements. Consult your carrier for details.

For specific parameter or mask values to use for configuring your hardware, see the relevant function description(s) in *Chapter 4. Digital Network* Interface Function Reference.

Set Alarm Handling Parameters and Masks

Use **dt_setalrm( )** to set the alarm handling mode for each Digital Network Interface device. Recommended settings are shown in *Table 7*. See **dt_setalrm** in *Chapter 4. Digital Network* Interface Function Reference for an example of setting the alarm handling mode.

Table 7. Recommended dt_setalrm( ) Settings

Telephony Standard	Configuration	Alarm Handling Mode
T-1	terminate drop and insert	DTA_TERM DTA_DROP
E-1	terminate drop and insert	DTA_TERM DTA_TERM

Use **dt_setevtmsk( )** to set the alarm handling masks for each Digital Network Interface device. At a minimum, your application must set masks to detect the T-1 or E-1 alarm conditions listed below.

NOTE: Unless your application is running in poll mode, your application must issue the SRL **sr_enbhdlr( )** function to enable trapping of the event return before setting alarm handling masks with **dt_setevtmsk( )**. You must enable event handlers when running in callback or interrupt mode. See Section ***Error! Reference source not found.***, for more details.

- T-1 alarm masks:
    - DTEC_RBL (receive blue alarm)

    - DTEC_RED (receive red alarm)
    - DTEC_RLOS (receive loss of sync)
    - DTEC_RYEL (receive yellow alarm)
- E-1 alarms:
    - DEEC_FSERR (frame sync error)
    - DEEC_MFSERR (multiframe sync error)
    - DEEC_RDMA (receive distant multiframe alarm)
    - DEEC_RLOS (receive loss of sync)
    - DEEC_RRA (receive remote alarm)
    - DEEC_RSA1 (receive signaling all 1s alarm)
    - DEEC_RUA1 (receive unframed all 1s alarm)

Initialize Time Slots

Before making or receiving any calls, an application should initialize all time slots to a known state. Initialization consists of:

- clearing/setting all signaling event masks
- setting time slots to the idle state
- setting the proper signaling mode
- idling the time slots

Setting event masks to a known state helps ensure that the application receives only those events it "expects" and can handle appropriately. Use **dt_setevtmsk( )** to set the signaling event masks to the desired state.

Setting all time slots to the idle state at the start of your application helps ensure that off-hook/on-hook transitions will be processed correctly. Use **dt_settssig( )** to set the state of a time slot to idle.

To generate system signaling from the Digital Network Interface board, it must be in the **signaling insertion mode**. In this mode, signaling from a resource board, such as a D/12x, will be overwritten by the Digital Network Interface board. To throughput the signaling data generated by the D/12x resource, the Digital Network Interface must be set to **transparent signaling mode**. In transparent signaling, signaling data from a PEB resource is routed straight through the Digital Network Interface board to the network without modification.

NOTES: 1. Before idling a T-1 time slot, set the signaling mode to signaling insertion. If the Digital Network Interface board is set to transparent signaling in a T-1 system and idle is transmitted, the signaling bits could be overwritten by the idle pattern. Use **dt_setsigmod( )** to initialize Digital Network Interface time slots to transparent signaling mode (DTM_TRANSP) or signaling insertion mode (DTM_SIGINS), as required.

2. To transmit a wink to the network, the Digital Network Interface time slot on which the wink is to be transmitted must be set to signaling insertion.

Use **dt_setidle( )** to idle a time slot.

The programming example below represents a typical initialization routine for a single time slot on a single board in a T-1 environment.

```
int init( )
{
   int dti1;

   /* open time slot 1 on DTI/211 or D/240SC-T1 board 1 ("dti1") */
      .
      .
      .
   /* Set time slot "onhook" */
   if ( dt_onhook ( dti1 ) !=0 ) {
      do_error( dti1, "dt_onhook( ) " );
      exit( 1 );
   }
   /* Reset all signaling event masks */
   if ( dt_setevtmsk( dti1, DTG_SIGEVT, 0, DTA_SETMSK ) !=0 ) {
      do_error( dti1, "dt_setevtmsk()" );
      exit ( 1 );
   }
}
int dt_onhook ( devh)
   int devh;

{
   int retval;

   /*
    * Transmit AOFF and BOFF
    */
   if ( ( retval = dt_settssig( devh, DTB_ABIT | DTB_BBIT,
         DTA_SUBMSK ) ) != 0 ) {
      do_error( devh, "dt_settssig()");
      return ( retval );
   }
   /*
    * Set signaling mode to signaling insertion
    */
```

```
   if ( ( retval = dt_setsigmod( devh, DTM_SIGINS ) ) != 0 ) {
      do_error( devh, "dt_setsigmod()");
      return ( retval );
   }
   /*
    * Enable idle transmission
    */
   if ( ( retval = dt_setidle( devh, DTIS_ENABLE ) ) != 0 ) {
      do_error( devh, "dt_setidle()");
   }
   return ( retval );
}
```

The **dt_setevtmsk( )** function disables generation of signaling events (see *Appendix A* or *Chapter 4. Digital Network* Interface Function Reference for details).

The **dt_onhook( )** routine is a user-defined function that forces the selected time slot to the on-hook, idle state using three separate library functions.

The **dt_setsigmod( )** function sets the time slot to signaling insertion mode. (This enables the device to transmit idle on the time slot without overriding signaling.)

The **dt_settssig( )** function forces the time slot to the on-hook state.

NOTE: This example assumes that clearing both the A-bits and B-bits is equal to the on-hook state. Your carrier service may differ.

The **dt_setidle( )** function transmits an idle pattern to the network on the selected time slot.

NOTE: When two Digital Network Interface boards are arranged in drop-and-insert configuration, **dt_setidle( )** can be used to disable pass-through operation. Transmitting idle overrides voice data being passed between Dialogic network devices on the selected time slot(s).

5.1.3. Processing

The main processing tasks for a Digital Network Interface application involve:

- Opening Digital Network Interface board and time slot devices
- Establishing connections

Opening and Using Board and Time Slot Devices

Windows NT opens and closes devices in the same manner that it opens and closes files. Windows NT views Digital Network Interface board and time slot devices as special files. When you open a file under Windows NT, it returns a unique file descriptor for that file. For example:

```
int file_descriptor;
file_descriptor = open(filename,mode);
```

Any subsequent action you perform on that file is accomplished by identifying the file using **file_descriptor**. No action at all can be performed on the file until it is first opened. Dialogic devices work in a similar fashion. You must first open a Dialogic device before you can perform an operation with it. When you open a device, the value returned is a unique handle for that process:

```
int device_handle;
device_handle = dt_open(device_name,mode);
```

NOTE: A Dialogic device handle is NOT the same handle returned by an **open( )** system call.

The Dialogic Digital Network Interface Windows NT device driver treats time slot and Digital Network Interface logical board devices similarly. Each is referred to by using a **device handle**. Any time you want to use the device, you must identify the device with its handle. A time slot device is an individual T-1 or E-1 time slot; for example, 1 of the 30 time slots on a DTI/212. A DTI/212 is one Digital Network Interface logical board device containing 30 time slot devices.

NOTE: Time slot devices can be opened without opening the board device containing that time slot. (It is unnecessary to open a board device unless you are setting or getting a board-level device parameter or alarm handling.)

To avoid conflict between the DTI driver and the generic driver, follow the guidelines below when defining devices in the configuration file:

Valid device names for DTI devices are found in the /dev directory. For the DTI/211 and DTI/212 boards, the device name format is dtiB*x* or dtiB*x*T*y*, where:

- *x* represents the Digital Network Interface logical board number

- *y* represents the time slot number, ranging from 1 to 24 (T-1) or 1 to 30 (E-1)

Valid device names for the D/240SC-T1 and D/300SC-E1 are built from the board name specified in the configuration file. The name of the D/240SC-T1 or D/300SC-E1 device may be in the form dtiB*x*, dtiB*x*, dtiB*x*T*y*, or dtiB*x*T*y* where:

- *x* represents the D/240SC-T1 or D/300SC-E1 logical board number
- *y* represents the time slot number, ranging from 1 to 24 (T-1) or 1 to 30 (E-1)

NOTE: The logical board device number of the D/240SC-T1 or D/300SC-E1 device must not be the same as the logical board number of the DTI/211 device or the DTI/212 device.

The following example shows how time slot 1 can be opened on two different D/240SC-T1 boards. For details on opening and closing Dialogic devices, refer to **dt_open( )** in *Chapter 4. Digital Network* Interface Function Reference.

```
int dti1;
int dti2;

/* Open device dtiB1T1 */
if ( ( dti1 = dt_open( "dtiB1T1", 0 ) ) == -1 ) {
   printf( "Cannot open DTI device dtiB1T1\n" );
   perror( " " );
   exit ( 1 );
}

/* Open device dtiB2T1 */
if ( ( dti2 = dt_open( "dtiB2T1", 0 ) ) == -1 ) {
   printf( "Cannot open DTI device dtiB2T1\n" );
   perror( " " );
   exit ( 1 );
}
```

NOTE: To avoid redundancy in the remaining programming examples in this chapter, the dt_**open( )** function will not be shown. The remaining examples are based on the device name conventions used in the examples above and assume that the relevant Digital Network Interface devices have previously been opened.

Establishing Connections

The examples below show how an incoming call can be established.

```
#include <srllib.h>
#include <dtilib.h>
#include <errno.h>

int devh;                    /* Time slot device handle */
int retval;                          /* Function return value */
int AON_received = 0;        /* AON_received flag */
int AON_handler( )
{
   int event = sr_getevttype( );
   int *datap = (int *)sr_getevtdatap( );
   short indx;
   if (event != DTEV_SIG) {
      printf("Unknown event %d received.  Data = %d\n",event,*datap);
      return 0;
   }
   for (indx = 0; indx < 4; indx++) {
      /*
       * Check if bit in change mask (upper nibble - lower byte) is
       * set or if this is a WINK (upper nibble - upper byte) event
       */
      if (!(*datap & (0x1010 << indx))) {
         continue;
      }
      switch (*datap & (0x1111 << indx)) {
      case DTMM_AON:
         AON_received = 1;
         break;
         .
         .
         .
      default:
         printf(Signal Event Error: Data = %d\n,*datap);
      }
   }
   return 0;
}
int wait_ring()
{
/*
 * This routine waits for an event from AON_handler to signal
 * an incoming call
 */

   int devh;      /* Time slot device handle */

/*
 * Open board 1 time slot 1 device (dti1)
 */
   if ( ( devh = dt_open( “dtiB1T1”, 0 ) ) == -1 ) {
      printf( "Cannot open device dtiB1T1.  errno = %d", errno );
      return ( -1 );
   }
   /*
    * Enable event handler to catch AON events
    */
   if ( ( retval = sr_ehbhdlr( devh, DTEV_SIG, AON_handler ) ) == -1 ) {
      printf( "Unable to set AON handler for device %s",
         ATDV_NAMEP ( devh ) );
      return( retval );
   }
```

```
   /*
    * Enable AON signaling transition events
    */
   if ( ( retval = dt_setevtmsk( devh, DTG_SIGEVT, DTMM_AON,
         DTA_SETMSK ) ) == -1 ) {
      printf ( "Error message = %s.", ATDV_ERRMSGP( devh ) );
      return ( retval );
   }
   /*
    * Now wait for an incoming call
    */
   while( AON_received == 0 ) {
      sleep( -1 );   /* Sleep until we receive an incoming call */
   }
   /* We have received an incoming call.  See next segment.  */
      .
      .
      .
}
```

The **AON_handler( )** routine is an asynchronous event handler that flags transitions of signaling bit "A" to the ON state. When the system detects an A-ON condition, **AON_handler( )** sets the AON_received flag to 1. The **AON_handler( )** function uses the SRL **sr_enbhdlr( )** function and related event management functions to determine when a signaling transition occurs. For details, see *Appendix A*.

NOTES: 1. Asynchronous signal handling is one of several ways to manage event notification and is shown for ease of explanation only. For more information on application development models, refer to the *Standard Runtime Library Programmer's Guide for Windows NT* (part of the *Voice Software Reference for Windows NT*).

2. This example assumes that setting the A-bit to ON is equal to the off-hook state. Your carrier service may differ.

The **wait_ring( )** routine is a user-defined function that performs the following tasks:

- Opens a time slot device
- Enables trapping of the desired signaling condition for the selected time slot device
- Puts the application to sleep until detection of the appropriate signaling condition.

The **dt_open( )** function opens time slot 1 on Digital Network Interface board 1 and assigns the returned device handle to variable **devh**.

The SRL **sr_enbhdlr( )** function enables processing by the **AON_handler** function of any signaling events detected on the device represented by **devh** (for details see *Appendix A*).

The **dt_setevtmsk( )** function enables detection of signaling bit A-ON transitions on device **devh**. Using E&M signaling protocol, a transition of the A-bit from OFF to ON signifies a request for service or ring event. When enabling event notification, the **dt_setevtmsk( )** function should be invoked only after the applicable handler has been enabled; otherwise, events could be missed. In the previous example, the **AON_handler( )** function was used.

The **while** statement puts the routine to sleep until the **AON_handler** routine detects a ring event. When a ring event is detected, processing resumes with the following segment.

```
/*
 * Continued from previous example
 */
      .
      .
int dt_offhook ( devh)
int devh;
{
   int retval;
   /*
    * Transmit AON and BON
    */
   if ( ( retval = dt_settssig( devh, DTB_ABIT | DTB_BBIT,
         DTA_ADDMSK ) ) != 0 ) {
      do_error( devh, "dt_settssig()" );
      return ( retval );
   }
   /*
    * Set signaling mode to signaling insertion
    */
   if ( ( retval = dt_setsigmod( devh, DTM_SIGINS ) ) != 0 ) {
      do_error( devh, "dt_setsigmod()";
      return ( retval );
   }
   /*
    * Disable idle transmission
    */
   if ( ( retval = dt_setidle( devh, DTIS_DISABLE ) ) != 0 ) {
      do_error( devh, "dt_setidle()" );
   }
   return ( retval );
}
```

The **dt_offhook( )** routine is a user-defined function that forces the selected time slot to the off-hook state and disables the transmission of idle using three separate library functions.

NOTE: The **dt_offhook( )** function is similar to the **dt_onhook( )** function explained above, under *Initialize Time Slots*, in the **init( )** example.

The **dt_setsigmod( )** function sets the time slot to signaling insertion mode.

NOTES:
1. Setting signaling to insertion mode is necessary if your application will be generating signaling from the Digital Network Interface board. To generate signaling from a Voice or other resource channel, set the signaling mode to transparent.
2. The DTI/212 board does not support transparent signaling mode in drop and insert. In a DTI/212-based drop-and-insert configuration, the signaling mode must be set to insertion and the signaling must be generated from the DTI/212 board.

The **dt_settssig( )** function forces the time slot to the off-hook state.

NOTE: This example assumes that setting the A-bits and B-bits is equal to the off-hook state. Your carrier service may differ.

The **dt_setidle( )** function disables the transmission of the idle pattern to the network on the selected time slot.

5.1.4. Terminating

When your process completes, devices should be shut down in an orderly fashion. Tasks that are performed to terminate an application generally include:

- Disable events
- Reset time slots
- Close devices

The example that follows is based in part on the processes illustrated in the previous examples. When your application is done processing a call, the following example should be executed.

NOTE: The following example assumes that relevant devices have been previously opened and variable names have been declared.

```
/* Disable all signaling events for this time slot */
if ( dt_setevtmsk( dti1, DTG_SIGEVT, 0, DTA_SETMSK) != 0 ) {
   do_error( dti1, "dt_setevtmsk()" );   /* Error function */
}


   /*
    * Disable event handler for AON events
    */
   if ( ( retval = sr_dishdlr( devh, DTEV_SIG, AON_handler ) ) == -1 ) {
      printf( "Unable to disable AON handler for device %s",
         ATDV_NAMEP ( devh ) );
      return( retval );
   }


/*
 * close time slot 1 on Digital Network Interface board 1 ("dti1") and Digital Network
Interface board2 ("dti2")
 */
if ( dt_close( dti1 ) != 0 ) {
   do_error( dti1, "dt_close()" );
}
if ( dt_close( dti2 ) != 0 ) {
   do_error( dti2, "dt_close()" );
}
```

The **dt_setevtmsk( )** function disables all currently enabled event notification masks. The routine that follows uses SRL functions (not illustrated) to disable all signal handlers (for SRL details, see *Appendix A*).

NOTES: **1.** The **dt_setevtmsk( )** and any SRL functions must be called in the order shown in the example.

2. SRL Event Management functions (such as **sr_dishdlr( )**, which disables an event handler) must be called prior to closing the device that is sending the handler event notifications (see *Appendix A* for SRL details).

The **dt_onhook( )** routine is a user-defined function that forces the selected time slot back to the on-hook, idle state using three separate library functions.

NOTE: The **dt_onhook( )** function is identical to the one explained above, under *Initialize Time Slots*, in the **init( )** example segment.

The **dt_setsigmod( )** function resets the time slot device to signaling insertion mode.

The **dt_settssig( )** function sets the time slot device to the on-hook state, ready for another call.

The **dt_setidle( )** function transmits idle on the selected time slot. When two Digital Network Interface boards are arranged in drop-and-insert configuration, **dt_setidle( )** can be used to disable pass-through operation. Transmitting idle overrides voice data being passed between Dialogic network devices on the selected time slot(s).

The **dt_close( )** function closes the time slot device.

5.1.5. Compiling and Linking

To compile and link your application, follow the syntax instructions for your version of the Windows NT C Development Package.

NOTE: If your application includes a Digital Network Interface and you are using the single-threaded asynchronous programming model, you must link the application with *libdti.lib*. If your application includes a Digital Network Interface and you are using the multi-threaded synchronous programming model, you must link the application with *libdtint.lib*. See the *Standard Runtime Library Programmer's Guide for Windows NT* for a full discussion of programming models.

Appendix A

Dialogic Standard Runtime Library

Digital Network Interface Entries and Returns

The Dialogic Standard Runtime Library (SRL) is a device independent library containing Event Management functions, Standard Attribute functions, and the DV_TPT Termination Parameter table. Dialogic SRL functions and data structures are described in detail in the *Standard Runtime Library Programmer's Guide for Windows NT* (part of the *Voice Software Reference for Windows NT*). This appendix lists all Dialogic SRL entries and returns applicable to the Digital Network Interface. *Table 8* provides a guide to the contents of this appendix.

NOTES: 1. This appendix documents the Dialogic Standard Runtime Library (SRL 4.1), included in the System Release Development Package for Windows NT. SRL 4.1 is fully compatible with the earlier releases of the SRL, therefore, existing applications designed to work with earlier versions of the voice software for Windows NT will work with the current voice software and SRL 4.1. However, Dialogic encourages you to upgrade your applications to the voice software included in the System Release Development Package to take advantage of new functionality. See the Dialogic Application Note entitled *Upgrading Applications for Voice Driver for Windows NT (version 4.1)* for instructions on upgrading applications.

2. In the context of this guide, "Digital Network Interface" is used to refer to the DTI/211 board, the DTI/212 board, the D/240SC-T1 board and the D/300SC-E1 board unless otherwise noted.

Table 8. Guide to Appendix A

Digital Network Interface

Table SRL Components	Data	Number
Event Management functions	Digital Network Interface inputs for Event Management functions.	*Table 9*
	Digital Network Interface returns from Event Management functions.	*Table 10*
Standard Attribute functions	Digital Network Interface values returned by the Standard Attribute functions.	*Table 11*
DV_TPT Table	Termination conditions and related data, required to set the DV_TPT for a Digital Network Interface device.	*Table 12*

NOTE: The header file for this library is *srllib.h*. It must be "included" in application code prior to including *dtilib.h*. For example:

```
#include <srllib.h>
#include <dtilib.h>
```

Event Management Functions

The enable processing of unsolicited and asynchronous termination events returned by Dialogic library functions. For the Digital Network Interface, these functions include:

dt_rundiag( )
dt_setevtmsk( )
dt_tstcom( )
dt_tstdat( )
dt_xmitwink( )

Each of the Event Management functions applicable to the Digital Network Interface are listed in the following tables. *Table 9* shows Digital Network

Interface-specific inputs and *Table 10* shows valid Digital Network Interface returns.

Table 9. Digital Network Interface Inputs for Event Management Functions

Event Management Function	Digital Network Interface-specific Input	Valid Input Value
sr_enbhdlr( ) *Enable event handler*	evt_type	DTEV_T1ERRC - T-1 alarm condition detected. DTEV_E1ERRC - E-1 alarm condition detected DTEV_SIG - Signaling transition event detected. DTEV_COMRSP - Successful communications test. DTEV_DATRSP - Response to data test. DTEV_RETDIAG - Diagnostic complete (DTI/211 devices only). DTEV_WINKCPLT - Wink transmission complete. DTEV_RCVPDG - Receive pulse digits. DTEV_PDDONE - Pulse dial complete events. DTEV_ERREVT - Error condition event. DTEV_MTFCNCPT - Multitasking function complete.

Event Management Function	Digital Network Interface-specific Input	Valid Input Value
sr_dishdlr( ) *Disable event handler*	evt_type	Same as above.
sr_getevtdev( ) *Get Dialogic device handle*	device	Digital Network Interface device handle.
sr_getevttype( ) *Get event type*	event type	DTEV_T1ERRC
		DTEV_E1ERRC
		DTEV_SIG
		DTEV_COMRSP
		DTEV_DATRSP
		DTEV_RETDIAG
		DTEV_WINKCPLT
		DTEV_RCVPDG
		DTEV_PDDONE
		DTEV_ERREVT
		DTEV_MTFCNCPT
sr_getevtlen( ) *Get event data length*	event length	Number of bytes in the data returned.
sr_getevtdatap( ) *Get pointer to event data*	event data	Pointer to event specific data.

Table 10. Digital Network Interface Returns Event Management Functions

Event Management Function	Digital Network Interface-specific Return	Returned Value
sr_getevtdev( ) *Get Dialogic device handle*	device	Digital Network Interface device handle.
sr_getevttype( )	event type	DTEV_T1ERRC

Event Management Function	Digital Network Interface-specific Return	Returned Value
Get event type		DTEV_E1ERRC
		DTEV_SIG
		DTEV_COMRSP
		DTEV_DATRSP
		DTEV_RETDIAG
		DTEV_WINKCPLT
		DTEV_RCVPDG
		DTEV_PDDONE
		DTEV_ERREVT
		DTEV_MTFCNCPT
sr_getevtlen( ) *Get event data length*	event length	Digital Network Interface event length information
sr_getevtdatap( ) *Get pointer to event data*	event data	Digital Network Interface event data pointer information

Standard Attribute Functions

The Standard Attribute functions return general Dialogic device information, such as the device name or the last error that occurred on the device. The Standard Attribute functions and the Digital Network Interface-specific information they return are listed in *Table 11*.

Table 11. Standard Attribute Functions

Standard Attribute Function	Information Returned for Digital Network Interface
ATDV_ERRMSGP( )	Pointer to string describing the error that occurred during the last function call on the Digital Network Interface. (See the error listing section and

Standard Attribute Function	Information Returned for Digital Network Interface
	function reference section of the appropriate software reference.)
ATDV_IOPORT( )	Valid port address for the Digital Network Interface.
ATDV_IRQNUM( )	Valid IRQ number range.
ATDV_LASTERR( )	The error that occurred during the last function call on the Digital Network Interface. (See the error listing section and function reference section of the appropriate software reference.)
ATDV_NAMEP( )	Pointer to device name (dtiBbXx).
ATDV_SUBDEVS( )	Number of subdevices (time slots, channels, etc.). List Digital Network Interface-specific returns. Refer to the *Standard Runtime Library Programmer's Guide for Windows NT* (part of the *Voice Software Reference for Windows NT*) for information on subdevices.

PT Structure

The DV_TPT termination parameter table sets termination conditions for a range of Dialogic products. The valid values for the DV_TPT structure in relation to the Digital Network Interface board are contained in this section.

The DV_TPT structure has the following format:

```
typedef struct dv_tpt  (
   unsigned short tp_type;        /* Flags describing this entry */
   unsigned short tp_termno;      /* Termination Parameter number */
   unsigned short tp_length;      /* Length of terminator */
   unsigned short tp_flags;       /* Parameter attribute flag */
   unsigned short tp_data;        /* Optional additional data */
   unsigned short rfu;            /* Reserved                 */
   DV_TPT         *tp_nextp;      /* Pointer to next termination
                                   * parameter if IO_LINK set
                                   */
}DV_TPT;
```

Table 12 shows the Digital Network Interface equates for this structure.

Table 12. DV_TPT Structure

Field	Value	Description
tp_type	IO_LINK	Structure is part of a linked list. The structure is linked through the Dialogic Standard Runtime Library.
	IO_CONT (default)	The next structure will be contiguous in memory.
	IO_EOT	This structure is the final entry in the DV_TPT table.
rfu	0	Reserved for future use.
tp_nextp	0	Pointer to the next termination parameter.

Refer to the Dialogic *Standard Runtime Library Programmer's Guide for Windows NT* for further information on the termination parameter table structure.

Digital Network Interface Programmer's Guide for Windows NT

190-CD

Appendix B

Related Publications

This section lists publications you should refer to for additional information on Dialogic products or T-1 and/or E-1 telephony.

Dialogic Digital Network Interface References

- For information about the Voice And Diagnostic Libraries and about library data structures, see the *Voice Programmer's Guide for Windows NT* in the *Voice Software Reference for Windows NT*.
- For information about the Standard Runtime Library, see the *Standard Runtime Library Programmer's Guide for Windows NT* in the *Voice Software Reference for Windows NT*.
- For information about installing 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*.

Other Dialogic Publications

- *Porting Windows NT Applications to the SCbus*, Dialogic Application Note
- *Why Is T-1 Important and How It Can Be Used*, Dialogic Application Note
- *Use of Dialogic T-1 for Telemarketing Applications*, Dialogic Application Note
- *Use of Dialogic T-1 in Operator Services Applications*, Dialogic Application Note
- *Use of Dialogic T-1 in Telephone Company Networks*, Dialogic Application Note
- *Use of Dialogic T-1 Equipment in CPE Gateways*, Dialogic Application Note

- *Integrating Analog Devices Into Dialogic-Based T-1 Voice Processing Systems*, Dialogic Application Note
- *Designing Operator-Assisted Voice Processing Systems*, Dialogic Application Note
- *Use of Dialogic Components in Automatic Number Identification Systems*, Dialogic Application Note
- *Ordering Service and Installing Equipment for T-1 Applications*, Dialogic Application Note

T-1/E-1 Technology

- Bellamy, John. *Digital Telephony*. 2nd ed. New York: John Wiley & Sons, 1991.
- Fike, John L., and George Friend. *Understanding Telephone Electronics*. Indiana: Howard W. Sams & Company, 1988.
- Flanagan, William A. *The Guide To T-1 Networking*. 4th ed. New York: Telecom Library Inc., 1990.

Glossary

A-LAW: A pulse-code modulation (PCM) algorithm used in digitizing telephone audio signals in E-1 areas.

ANI: Automatic Number Identification. A feature of certain telecommunications networking protocols or processes that allows the caller's phone number to be detected and displayed by the called party.

asynchronous function: On Windows NT platforms, 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 completion event. See *synchronous function*.

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

AT-class: 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.

BCD: Binary coded decimal. A numbering system often used in data processing where each decimal digit is represented by a four-bit binary value.

BIOS: Basic input-output system. The set of permanently stored system service programs needed to manage the PC and consisting of drivers and other software to control peripheral units.

B8ZS: Binary 8-zero Substitution. Basic bipolar coding algorithm for digital telephony. At the transmitting end, a string of 8 zeros is deliberately replaced with a pulse that produces a bipolar violation. At the receiving end, bipolar violations are replaced with a string of 8 zeros. See *HDB3*.

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.

CAS: Channel Associated Signaling. The signaling protocol used with the CEPT E-1 telephony standard. In CAS, one of the 32 channels, time slot 16, is dedicated to signaling for all of the 30 voice channels. Unlike T-1 systems, which use robbed-bit signaling, telephony systems using CAS are considered examples of out-of-band signaling. See *in-band signaling*, *robbed-bit signaling*.

CEPT: Conference of European Postal and Telecommunications administrations. Defines how bits of a PCM carrier system in E-1 areas will be used and in what sequence. CEPT format consists of 30 voice channels, one signaling channel, and one framing (synchronization) channel. See *E-1*.

CCITT: International Telephone and Telegraph Consultative Committee, a part of the ICU (International Telecommunications Union) responsible for formulating telephony and other standards, such as E-1.

CO: Central Office. The telephone company facility where subscriber lines are linked, through switches, to other subscriber lines (including local and long distance lines).

CRC: Cyclic Redundancy Check. A basic error checking mechanism for digital transmissions in which a CRC character, indicating the number of bits in a block of data, is included in the transmission. The receiving end calculates the number of bits in the block independently and compares the result to the received CRC character. CRC4 is a specific algorithm used to implement error checking.

crossover cable: A cable used to interconnect two Dialogic network boards, often to join two T-1 or E-1 lines. The cable is split and folded so that the lines carrying network receive data on one side of the crossover connector mate with network transmit lines on the other side of the crossover.

D/xxx: A general term used to refer to any Voice board made by Dialogic.

D/120: A model of 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 model of 12-channel Voice board from Dialogic with all the features of the D/120 plus patented call-analysis algorithms for outbound applications and multifrequency (MF) tone capability.

D/121A: A model of 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/12x: A general term used to refer to any 12-channel Voice board made by Dialogic.

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

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

D/81: A model of 8-channel Voice board from Dialogic which can interface to eight analog telephone lines in conjunction with Dialogic LSI products or to digital E-1 spans in conjunction with Dialogic DTI products.

data structure: C programming term for a data element consisting of fields, where each field may have a different type definition and length. The elements of a data structure usually share a common purpose or functionality, rather than being similar in size, type, etc.

device: A computer peripheral or component that is controlled through a software device driver. A Dialogic Digital Network Interface board is considered a physical board containing one or more logical *board devices*, and each time slot on the board is a *time slot 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 12 device channels on a D/12x. Compare *time slot*.

DNIS: Dialed Number Identification Service. An 800 service feature that allows a business to determine the geographical area from which a call

originated by the digits dialed (a different phone number is made available to callers in each region).

drop-and-insert: A Dialogic system configuration in which two network boards are interconnected by a PEB crossover cable and continuously pass all time slots through to each other. A time slot from one network can be "dropped" to a resource module (such as a D/12x) for processing. In return, the resource module can "insert" signaling and audio into the bit stream received from the other side of the PEB crossover connector. (A resource module can insert only to the network board on the same side of the PEB crossover.) This bit stream is applied through the network module for outbound transmission to the attached network span.

DTI/101: A first generation model of Dialogic digital telephony interface boards designed for use with the T-1 telephony standard used in North American market.

Digital Network Interface: A general term used to refer to any of Dialogic second generation digital telephony interface boards.

DTI/211: A second generation model of Dialogic digital telephony interface device designed for use with the T-1 telephony standard.

DTI/212: A second generation model of Dialogic digital telephony interface board designed for use with the E-1 telephony standard.

E-1: Another name given to the CEPT digital telephony format devised by the CCITT. See *CEPT*.

E&M protocol: A signaling protocol that defines the sending and receiving of signals. E&M protocol is the most common protocol on T-1 trunks.

8-bit expansion slot: These slots connect additional circuit boards (expansion boards) into the PC bus. The slot contains contacts for the 62 lines of the standard PC bus.

EPROM: Erasable Programmable Read-Only Memory.

event: An unsolicited or asynchronous communication from a hardware device to an operating system, application, or driver. Events are generally attention getting messages, allowing a process to decide when and where to redirect its resources.

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

Extended Attribute functions: Class of functions that take one input parameter (a valid Dialogic device handle) and return device-specific information. For instance, a Digital Network Interface Windows NT Extended Attribute function returns information specific to the Digital Network Interface class of devices. Extended Attribute function names are case-sensitive and must be in capital letters. See *Standard Attribute functions*.

firmware: A set of program instructions that reside (usually in EPROM) on an expansion board.

HDB3: Digital line-coding algorithm used by the DTI/212. HDB3 (high density bipolar) is a variation on the basic bipolar coding scheme BNZS (Binary *N* zero substitution), in which a string of *N* zeros is deliberately replaced with a pulse that produces a bipolar violation. At the receiving end, bipolar violations are replaced with a string of *N* zeros. In HDB3, strings of four zeros are replaced with a bipolar violation. The bipolar violation always occurs in the fourth bit position.

in-band signaling: **1.** In an analog telephony circuit, in-band refers to signaling that occupies the same transmission path and frequency band used to transmit voice tones. **2.** In digital telephony, "in-band" has come to mean signaling that is transmitted within an 8-bit voice sample or time slot, as in T-1 "robbed-bit" signaling. **3.** On the Dialogic PCM Expansion Bus (PEB), signaling is considered "in-band" only if it occupies the same transmission path and frequency band used to transmit voice data. See *CAS, robbed-bit signaling*.

IRQ: Interrupt request. 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.

Mu-LAW: (1) A pulse code modulation (PCM) algorithm used in digitizing telephone audio signals in T-1 areas.ss (2) The PCM coding and compounding standard used in Japan and North America.

MSI: Modular Station Interface. A PEB-based Dialogic expansion board that interfaces PEB time slots to analog station devices by way of modular daughterboards.

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

PC AT: An IBM Personal Computer or compatible having the characteristics described under AT-class.

PC-bus: The common communication channel in a PC. The channel uses an 8-bit data path architecture. The bus contains 62 lines for data and power transmission, addressing, and interrupt request handling.

PEB: PCM Expansion Bus. The common communication medium for passing signaling, audio, and control information between Dialogic D/12x, Digital Network Interface, and other PEB-compatible expansion boards. Non-Dialogic products using the appropriate encoding method and clock rate may interface with Dialogic products by using this bus.

peripheral: Any equipment, apart from the central processing unit, that provides a system with outside communication or additional facilities.

PSTN: Public Switched Telephone Network.

robbed-bit signaling: The type of signaling protocol implemented in areas using the T-1 telephony standard. In robbed-bit signaling, signaling information is carried in-band, within the 8-bit voice samples. These bits are later stripped away, or "robbed," to produce the signaling information for each of the 24 time slots. See *CAS, in-band signaling*.

route: Assign a resource to a time slot.

separate signaling: A Dialogic-unique signaling protocol for data crossing the PEB that makes use of different physical lines than are used for voice data. Boards that support this protocol can write signaling data to the PEB independent of the network signaling protocol. See CAS, in-band signaling, out-of-band signaling, robbed-bit signaling.

16-bit AT expansion slot or 16-bit AT bus slot: These slots connect additional circuit boards (expansion boards) into the bus of AT machines. One of the main features of the AT bus is 16-bit memory data transfer. AT expansion slots are really two slots placed end to end

on the motherboard. The larger slot contains contacts for the 62 lines of the standard PC bus. The smaller slot contains contacts for the 36 lines added to the standard PC bus to make up the AT bus.

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.

signaling insertion: Mode in which the Digital Network Interface (or any network board) overwrites signaling data from PEB resource modules in order to perform signaling to the network (see *transparent signaling*).

SpringBoard: A Dialogic expansion board using digital signal processing to emulate the functions of other products. The SpringBoard is a development platform for Dialogic products.

SRL: Standard Runtime Library. A Dialogic software resource containing Event Management functions, Standard Attribute functions, and data structures used by all Dialogic devices, but which return data unique to the device. Version 2.00 of the SRL is included with version 3.00 and later of the Voice Development Package for Windows NT. Version 1.00 of the SRL was packaged with the DTI/1xx Development Package for Windows NT and if needed, can still be ordered separately.

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 *Extended Attribute functions*.

synchronous function: On Windows NT platforms, a function that blocks program execution until a value is returned by the device. Also called a blocking function. 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. Each individual digitized communication is called a time slot. In T-1 areas, 24 time slots are multiplexed onto a single twisted-wire pair. In E-1 areas, 32 time slots are multiplexed together. Compare *device channel*.

time slot assignment: The ability to route the digital information contained in a time slot to a specific device channel. See *device channel*.

transparent signaling: Mode in which the Digital Network Interface (or any network board) accepts signaling data from a PEB resource module transparently, or without modification. In effect, the resource module performs signaling to the network (see *signaling insertion*).

wink: In T-1 or E-1 systems, a signaling bit transition from on to off, or off to on, and back again to the original state. In T-1 systems, the wink signal can be transmitted on either the A or B signaling bit. In E-1 systems, the wink signal can be transmitted on either the A, B, C, or D signaling bit. Using either system, the choice of signaling bit and wink polarity (on-off-on or off-on-off hook) is configurable through Digital Network Interface board download parameters.

Voice board: Any of Dialogic D/xxx family of 4-, 8-, and 12-channel voice-store-and-forward boards.

Index

A

Dialogic Sales Offices

North American Sales
1-800-755-4444
fax: 201-631-9631

Corporate Headquarters
1515 Route 10
Parsippany, NJ 07054-4596
201-993-3000
fax: 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 North Creek Parkway #120
Bothell, WA 98011

GammaLink Division
1314 Chesapeake Terrace
Sunnyvale, CA 94089

Computer-Telephone Division
100 Unicorn Park Drive
Woburn, MA 01801

Spectron Microsystems Division
315 Bollay Drive
Santa Barbara, CA 93117
805-968-5100
fax: 805-968-9770

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

Latin America and the Caribbean
Dialogic Latin America and Caribbean
Roque Saenz Pena
730 Tercer Piso
Oficina 34 y 37
1035 - Buenos Aires
Argentina
541-328-1531 or -9943
fax: 541-328-5425

European Headquarters (serving Western Europe, Middle East and Africa)
Dialogic Telecom Europe N.V.-S.A.
Airway Park
Lozenberg 23 (3rd floor)
B-1932 Sint-Stevens-Woluwe
Belgium
32-2-712-4311
fax: 32-2-712-4300

Germany, Switzerland & Austria
Dialogic Telecom Deutschland GmbH
Ridlerstrasse, 11
D-80339 Munich
Germany
49-89-50-20-09-14
fax: 49-89-50-24-540

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

Italy
Dialogic Telecom Italy
Strade Pavese, 1/3
I-20089 Rozzano
Milan Italy
39-2-57554302
fax: 39-2-57554310

United Kingdom, Ireland and Scandinavian Countries
Dialogic Telecom U.K. Ltd.
Dialogic House

Dairy Walk
Hartley Wintney
Hampshire
RG27 8XX
United Kingdom
44-1252-844000
fax: 44-1252-844525

Japan and Korea
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

East Asia, Southeast Asia, West Asia and Australia
Dialogic SEA Pte. Ltd.
150 Beach Road
#17-08 Gateway West Bldg.
Singapore 0718
65-298-8208
fax: 65-298-1820

New Zealand
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

Dialogic On-Line Information Retrieval System
1-800-755-5599
or 201-993-1063

Dialogic Telecom Europe (DTE) On-Line Information Retrieval System
32-2-712-4322

GammaLink Fax on Demand
408-734-9906

Computer Telephony BBS (ctBBS)
201-993-0864

Dialogic Telecom Europe (DTE) BBS
32-2-725-7846

CTI@ Dialogic WWW Site
http://www.dialogic.com

Dialogic Sales Internet
sales@dialogic.com

NOTES

NOTES

NOTES