Dialogic Audio Conferencing Software Reference

for Windows NT

05-0512-001

Publication Date: July, 1997

Part Number: 05-0512-001

Dialogic Corporation
1515 Route 10
Parsippany NJ 07054

Technical Support
Phone: 973-993-1443
Fax: 973-993-8387
BBS: 973-993-0864
Email: CustEng@dialogic.com

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

Table Of Contents

1. Introduction 1
1.1. Organization of this Guide 1
1.2. Dialogic Audio Conferencing Features 1
1.3. Software Features 2
1.3.1. Active Talkers 3
1.3.2. Volume Control 3
1.3.3. Resource Allocation 4
1.4. Compatibility 4
1.5. Functional Description 4
Products Covered by this Guide 9
Product Terminology 9
How to Use this Guide 10
2. DCB/SC Library Function Overview 13
2.1. Library Function Categories 13
2.1.1. Auxiliary Functions 13
2.1.2. Conference Management Functions 14
2.1.3. Configuration Functions 14
2.1.4. Device Management Functions 14
2.2. Error Handling 15
2.3. Include Files 18
3. Function Reference 19
3.1. Documentation Conventions 19
dcb_addtoconf( ) - adds one conferee to an existing conference 20
dcb_close( ) - closes the DCB/SC device 26
dcb_delconf( ) - deletes a previously established conference 28
dcb_dsprescount( ) - returns the available conferencing resource count 32
dcb_evtstatus( ) - gets or sets the status of a process-wide event 41
dcb_getatibits( ) - returns the active talker indicator bits 46
dcb_getbrdparm( ) - retrieves a DCB/SC board parameter value 50
dcb_getcde( ) - retrieves the attributes of a conferee 53
dcb_getcnflist( ) - retrieves a conferee list 58
dcb_getdigitmsk( ) - returns the digit mask 63
dcb_gettalkers( ) - retrieves information about the conferees actively talking 68
dcb_monconf( ) - adds a monitor to a conference 72
dcb_open( ) - opens a device 77

dcb_remfromconf( ) - removes a conferee from a conference 80
dcb_setbrdparm( ) - sets DCB/SC board device parameters................................ 85
dcb_setcde( ) - changes the attributes of a conferee 88
dcb_setdigitmsk( ) - enables specific digit detection 93
dcb_unmonconf( ) - removes a monitor from a conference 99

4. Application Guidelines 103

4.1. Writing a Simple Application 103
4.1.1. General Guidelines 103
4.1.2. Initialization 105
4.1.3. Terminating.. 106
4.1.4. Compiling and Linking 106
4.1.5. Aborting 106

Appendix A - Standard Runtime Library: DCB/SC Entries and Returns .. 109

Event Management Functions.. 110
Standard Attribute Functions 111

Appendix B - Related Publications 113

Dialogic References.. 113

Glossary.. 115

Index 119

List Of Tables

Table 1. Error Types Defined in dtilib.h 16
Table 2. Error Types Defined in msilib.h 17
Table 3. Valid Attribute Combinations... 21
Table 4. Valid Attribute Combinations... 36
Table 5. Possible Returns for Channel Attribute 54
Table 6. Valid Attribute Combinations... 89
Table 7. Guide to Appendix A.. 109
Table 8. DCB/SC Inputs for Event Management Functions............................. 110
Table 9. DCB/SC Returns from Event Management Functions........................ 111
Table 10. Standard Attribute Functions.. 111

List of Figures

Figure 1. Overview of the DCB/320SC Board 6
Figure 2. Overview of the DCB/640SC Board 7
Figure 3. Overview of the DCB/960SC Board 8

1. Introduction

The Dialogic Audio Conferencing Package supports the Dialogic DCB/SC product family. The DCB/SC series includes the following boards:

DCB/320SC	32 conferencing resources
DCB/640SC	64 conferencing resources
DCB/960SC	96 conferencing resources

Each product in the DCB/SC series contains one, two, or three DSP(s). Each Digital Signal Processor (DSP) supports up to 32 conferees, for a maximum of 96 (3 sets of 32) conferencing resources.

1.1. Organization of this Guide

This Software Reference is organized as follows:

Chapter 1 presents an overview of the Dialogic Dialogic Audio Conferencing technology and provides a description of the DCB/SC board.

Chapter 2 provides an overview of the DCB/SC library functions. A detailed description and a programming example for each function is found in *Chapter 3*.

Chapter 4 provides brief guidelines to use for developing applications using the library functions.

1.2. Dialogic Audio Conferencing Features

Dialogic Audio Conferencing provides the following features and benefits to your application:

- SCbus access to build high density systems — up to 1024 time slots.
- Conferencing of up to 32 conferees per conference, connected across the SCbus.

- Monitoring feature enabling many participants to monitor a single conference.
- Coach/pupil feature allowing two selected conferees to establish a private subconference with respect to the overall conference.
- DTMF digit detection for any conferee, allowing the application to determine when or if any party has generated a DTMF digit.
- On-board digit detection for each party in the conference.
- Tone clamping to reduce the amount of DTMF tones heard in a conference.

  NOTE: DTMF tones may be heard by conferees if the application encourages the user to repeatedly press DTMF tones, for example, press 9 to raise volume.
- Volume control for any conferee by simply issuing a pre-programmed DTMF digit.
- Automatic gain control (AGC) for all conferees.
- Active talker indication to determine which conferees in any given conference are currently talking.

1.3. Software Features

Each DCB/SC board has one or more DSPs. Conferences are associated with DSPs and may not span DSPs. Each DSP has 32 resources. Every time a conferee is added to a conference, a resource is used. When the conferee is removed from a conference and/or a conference is deleted, all associated resources are freed.

An application has the flexibility to associate conferences with any desired DSP. This allows for load balancing and better control over resources. Applications associate conferences with DSPs using a DSP device handle. DSP device handles are obtained using the **dcb_open( )** function.

Examples of valid DSP names used in **dcb_open( )** are shown below:

dcbBxD1, dcbBxD2, dcbBxD3

where x indicates the board number associated with the DCB/SC board.

1.3.1. Active Talkers

Active talkers are those conferees providing “non-silence” energy. The DCB/SC board can provide indication of the active talkers in a specified conference.

To retrieve active talkers, turn on the active talker feature by setting MSG_ACTID to ACTID_ON using the **dcb_setbrdparm( )** function. Then, retrieve active talkers using the **dcb_gettalkers( )** function. The frequency of active talker retrieval is determined by the application, although internally the active talkers are updated every 100 ms.

The internal implementation of the active talker feature consists of three parts:

1. The DSPs on the DCB/SC boards are programmed to furnish non-silence information for each of the 32 resources every 100 ms. This information is in bit form and is referred to as *active talker indicator (ATI) bits*. If the bit associated with a resource is on, it indicates non-silence energy in that resource during the time interval. The ATI bit pattern for a specific instant is retrieved using the **dcb_getatibits( )** function.
2. The resource assignment table contains resource assignments of a DSP to conferees in different conferences.
3. A combination of parts one and two is used to provide the output from the **dcb_gettalkers( )** function.

1.3.2. Volume Control

A conferee in a conference may wish to change the volume level of the received signal. This is accomplished using the volume control feature.

The MSG_VOLDIG parameter in **dcb_setbrdparm( )** allows the application to define the digits that cause the volume level to be adjusted up, down or back to the default value. When a conferee other than a monitor or receive-only party presses a digit programmed to change volume, the received signal is adjusted. The volume control digits do not cause events to be sent to the application, even if digit detection is turned on using the **dcb_setdigitmsk( )** function. Volume control digits are used only for controlling volume.

1.3.3. Resource Allocation

As stated earlier, each DSP has 32 resources. The available resource count is retrieved using the **dcb_dsprescount( )** function. Calling any of the following functions will cause the available resource count to change:

Function	Result
dcb_estconf( )	Uses the total number of conferees established in the new conference.
dcb_addtoconf( )	Uses one resource every time a conferee is successfully added to a conference.
dcb_remfromconf( )	Frees one resource.
dcb_delconf( )	Frees all resources in use by the conference, including the monitor.
dcb_monconf( )	Uses one resource.
dcb_unmonconf( )	Frees one resource.

NOTE: The channel selector value of the conferee does not affect the resource usage.

1.4. Compatibility

The Dialogic Audio Conferencing products (DCB/SC boards) interface with other Dialogic digital front-end products, such as the D/240SC-T1 and D/300SC-E1 boards. The DCB/SC boards do not interface with analog front-end boards, such as the D/160SC-LS, LSI/81C, LSI/161SC, MSI/SC boards.

1.5. Functional Description

The DCB/xxxSC series of Dialogic audio conferencing products can provide from 32 to 96 conferencing resources in increments of 32. The number of resources on any board is determined by the number of DSPs resident on the board. The baseboard with one DSP, DCB/320SC, is used for a 32 conferencing resource solution. The baseboard with two DSPs, DCB/640SC, is used for a 64 conferencing resource solution. The baseboard and daughterboard with an additional DSP, DCB/960SC, is used for a 96 conferencing resource solution.

Each DCB/SC product is built of one or more conference subassemblies. A subassembly consists of an SC2000 chip and a Motorola 56002 DSP (with attendant high speed RAM). Both of these components are under the control of the single Intel 80C286 control processor, and serviced by a single High-level Data Link Controller (HDLC) HDLC controller and dual-ported shared RAM.

Each conference subassembly can fully process up to 32 digitally derived conference parties. The subassembly is capable of maintaining up to 16 simultaneous conferences. The maximum conference size is 32 parties, hence all 32 parties of a given subassembly could reside in the same conference. Conferences can be created, deleted, or modified up to the conferee limits. Each subassembly implements the DCB/SC conferencing features described in *Section 1.2. Dialogic Audio Conferencing Features*.

The on-board control processors control access to the SCbus. This message bus is a separate SCbus channel carrying messages and control information among devices, such as out-of-band signaling and collision detection and resolution information. Out-of-band signaling enables faster response time to the messages and control information carried on the dedicated message channel. Bus contention and resolution capability uses a set of fixed and rotating guidelines to detect when two devices are trying to access the same SCbus slot and resolves this situation so each resource has equal access.

The on-board control processor(s) control all operations of the DCB/SC board via a local bus and interprets and executes commands from the host PC. The processor(s) handle real-time events, manages data flow to the host PC to provide faster system response time, reduces PC host processing demands, and frees the DSPs to perform signal processing. Communications between a processor and the host PC is via the dual-port shared RAM that acts as an input/output buffer. This RAM interfaces to the host PC via the ISA bus. All operations are interrupt driven to meet the demands of real-time systems. When the system is initialized, DCB/SC firmware to control all board operations is downloaded from the host PC to the on-board code/data RAM and DSP RAM.

The board locator technology (BLT) circuit operates in conjunction with a rotary switch eliminating the need to set any jumpers or DIP switches.

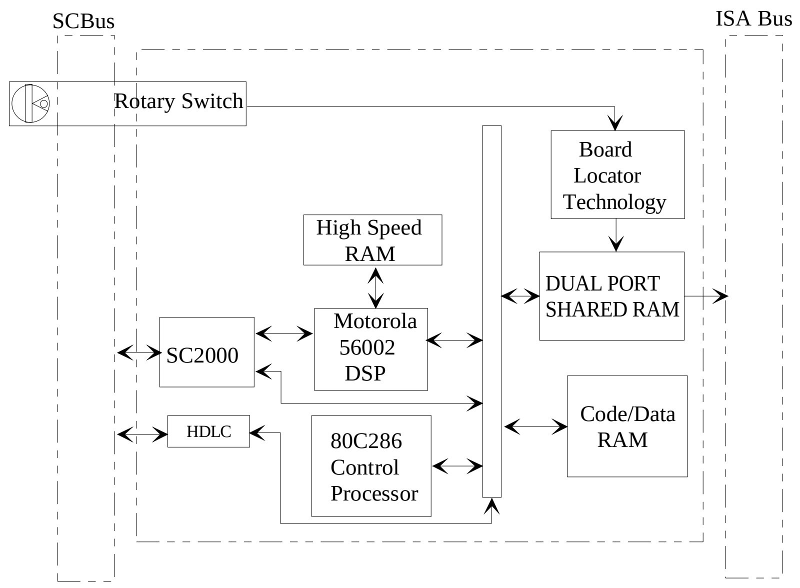

Figure 1. Overview of the DCB/320SC Board

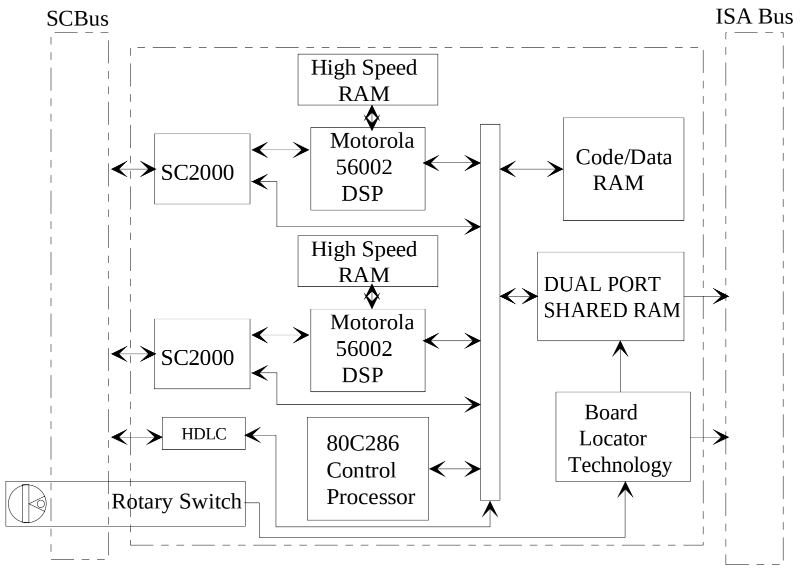

Figure 2. Overview of the DCB/640SC Board

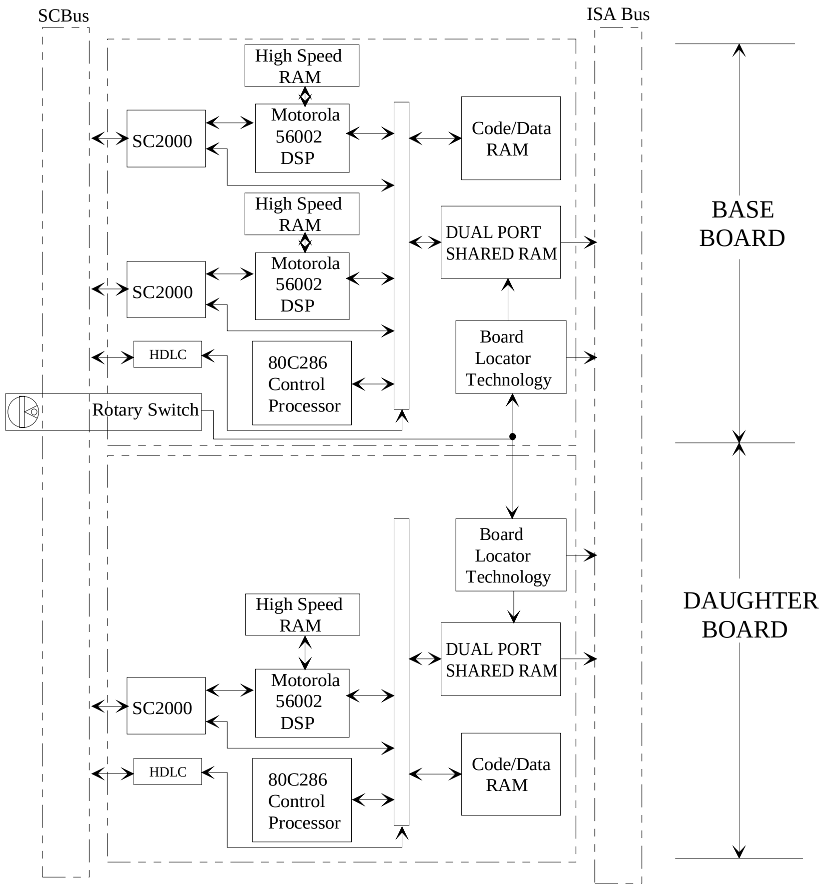

Figure 3. Overview of the DCB/960SC Board

Products Covered by this Guide

The Dialogic Audio Conferencing product line consists of three models listed in the table below.

Model	Description
DCB/320SC	Baseboard-only product with 1 DSP for 32 audio conferencing resources.
DCB/640SC	Baseboard-only product with 2 DSPs for 64 audio conferencing resources.
DCB/960SC	Baseboard product with 2 DSPs and 1 daughterboard with 1 DSP for 96 audio conferencing resources.

Product Terminology

The following product naming conventions are used throughout this guide:

D/41ESC refers to the Dialogic 4-channel voice board with on-board analog loop start interface.

D/160SC-LS refers to the Dialogic 16-channel voice board with on-board 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 on-board T-1 digital interface.

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

D/300SC refers to the Dialogic voice board for use with a network interface board.

DCB/SC refers to the Dialogic single slot, DSP-based conferencing solution.

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/241SC, D/300SC-E1, D/300SC, and D/301SC.

MSI/SC refers to the Dialogic modular station interface board for the SCbus.

MSI/SC-R refers to the Dialogic station interface product with ringing capability.

SCbus is the TDM (Time Division Multiplexed) bus voice, telephone network interface, and other technology resource boards together.

SpanCard - same as **DIALOG/HD**.

VFX/40ESC is a Dialogic SCbus voice and fax resource board with on-board analog loop-start interfaces. The VFX/40ESC board consists of a D/41ESC baseboard and a FAX/40E daughterboard that provides 4-channels of enhanced voice and fax services in a single slot. Throughout this document, all references to the D/41ESC board apply to the D/41ESC baseboard component of the VFX/40ESC board.

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

How to Use this Guide

This guide is written for users who have purchased a Dialogic DCB/SC board and the Dialogic Audio Conferencing Package software for installation on a PC operating in a Windows NT environment.

The following steps explain the order in which the board and Dialogic software for Windows NT should be installed, checked, and programmed:

1. Prepare the board for installation using the appropriate hardware quick installation card (see *Appendix B*).
2. Install the board in your PC following the procedure in the hardware quick installation card (see *Appendix B*).

3. Install the necessary software following the procedure described in the *Dialogic Audio Conferencing Package Release Notes for Windows NT*.
4. Refer to this *Dialogic Audio Conferencing 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*).

2. DCB/SC Library Function Overview

The Dialogic DCB/SC Windows NT library provides support for the Dialogic Dialogic Audio Conferencing product line. This chapter provides an overview of the library functions. A detailed reference for these functions is located in Chapter 3.

2.1. Library Function Categories

The DCB/SC library functions provide the necessary building blocks to create digital conferencing applications. These functions can be divided into the following categories:

Auxiliary	• adds flexibility while using active talker feature
Conference Management	• controls conferencing
Configuration	• sets or retrieves device parameters
Device Management	• opens and closes devices

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

2.1.1. Auxiliary Functions

dcb_evtstatus( )	• gets or sets the status of a process-wide event
dcb_getatibits( )	• reads active talker bits
dcb_gettalkers( )	• gets active talkers in a conference

The **dcb_getatibits( )** function provides flexibility while using the active talker feature. **dcb_evtstatus( )** is used to retrieve or set events related to the process and not a specific device. **dcb_gettalkers( )** retrieves active talker information.

2.1.2. Conference Management Functions

dcb_addtoconf( )	• adds one conferee to an existing conference
dcb_delconf( )	• deletes a conference
dcb_estconf( )	• establishes a conference
dcb_getcde( )	• retrieves the attributes of a conferee
dcb_getcnflist( )	• gets list of conferees in a conference
dcb_monconf( )	• adds a monitor to a conference
dcb_remfromconf( )	• removes a conferee from a conference
dcb_setcde( )	• sets the attributes of a conferee
dcb_unmonconf( )	• removes the monitor from a conference

These functions are used to manage all conference activities.

2.1.3. Configuration Functions

dcb_dsprescount( )	• retrieves the free resource count
dcb_getbrdparm( )	• gets DCB/SC board device parameters
dcb_getdigitmsk( )	• reads digit event message mask
dcb_setbrdparm( )	• changes DCB/SC board device parameters
dcb_setdigitmsk( )	• sets digit event message mask

These functions set the DCB/SC board or DSP device parameters, check the status of the DCB/SC board or DSP device parameter settings, and retrieve or set specific information about the DCB/SC DSPs or conferences.

2.1.4. Device Management Functions

dcb_open( )	• opens a DCB/SC device
dcb_close( )	• closes a DCB/SC device

These functions are used to open and close DCB/SC devices. Before using any other DCB/SC library function on a specific device, the device must first be opened. Each time a device is opened using **dcb_open( )**, the function returns a new unique device handle.

2.2. Error Handling

All the DCB/SC library functions return a value that indicates the success or failure of the function call. DCB/SC library functions return one of the following values:

0	function success
-1	function error

If a function fails, the error can be retrieved using the Standard Runtime Library (SRL) function **ATDV_LASTERR( )**.

NOTES:

1. The function **dcb_open( )** is an exception to the above error-handling rules. A **dcb_open( )** function call returns a device handle if the function call is successful. A device handle is a non-zero value. If **dcb_open( )** fails, the return code is -1 and the specific error is found in **errno** defined in *errno.h*.
2. If the error returned by **ATDV_LASTERR( )** is **EDT_SYSTEM**, a Windows NT system error has occurred, and you need to check the global variable **errno** defined in *errno.h*.

The causes and values of error codes that may be returned to the application by the DCB/SC board are a subset of errors used for the Dialogic Digital Network Interface products or the Dialogic Modular Station Interface (MSI/SC) products. The following tables list possible error codes.

Table 1. Error Types Defined in dtilib.h

Error Returned	Description
EDT_SH_LIBBSY	Switching Handler Library is busy.
EDT_SH_BADINDX	Invalid Switching Handler index number.
EDT_SH_LIBNOTINIT	Switching Handler Library has not been initialized.
EDT_SH_NOCLK	Switching Handler Clock fallback failed.
EDT_SH_MISSING	Switching Handler is not present.
EDT_SH_BADMODE	Invalid bus mode.
EDT_SH_BADLCLTS	Invalid local time slot number.
EDT_SH_BADTYPE	Invalid local time slot type.
EDT_ADDRS	Incorrect address.
EDT_BADBRDERR	Board is missing or defective.
EDT_BADCMDERR	Invalid or undefined command to driver.
EDT_BADCNT	Incorrect count of bytes requested.
EDT_BADDEV	Bad device error.
EDT_BADGLOB	Incorrect global parameter number.
EDT_BADPORT	First byte appeared on reserved port.
EDT_BADVAL	Invalid parameter value passed in value pointer.
EDT_CHKSUM	Incorrect checksum.
EDT_DATTO	Data reception timed out.
EDT_DTTSTMOD	In test mode; cannot set board mode.
EDT_FWERR	Firmware returned an error.
EDT_INVBD	Invalid board.
EDT_INVMSG	Invalid message.
EDT_INVTS	Invalid time slot.
EDT_MBFMT	Wrong number of bytes for multiple byte request.
EDT_MBIMM	Received an immediate termination.
EDT_MBINV	First byte appeared on data port.
EDT_MBOVR	Message was too long.
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_NOCLK	No clock source present.
EDT_NOIDLEERR	Time slot is not in idle/closed state.
EDT_NOMEMERR	Cannot map or allocate memory in driver.

Error Returned	Description
EDT_NOTDNLD	Not downloaded.
EDT_PARAMERR	Invalid parameter.
EDT_RANGEERR	Bad/overlapping physical memory range.
EDT_SIGINS	Insertion signaling 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_SYSTEM	Windows NT system error - check the global variable **errno** for more information .
EDT_TMOERR	Timed out waiting for reply from firmware.
EDT_TSASN	Time slot already assigned.

Table 2. Error Types Defined in msilib.h

Error Returned	Description
E_MS1PTY	Cannot remove party from one party conference.
E_MSBADCHPARM	Invalid channel parameter number.
E_MSBADVAL	Invalid parameter value.
E_MSCHASNCNF	Channel is assigned to conference.
E_MSCNFFUL	Conference system is full.
E_MSCNFLMT	Exceeds conference limit.
E_MSGLOBREAD	Cannot read parameter globally.
E_MSINVCB	Invalid control block ID.
E_MSINVCATTR	Invalid conference attribute.
E_MSINVCNF	Invalid conference number.
E_MSINVDSP	Invalid DSP specified.
E_MSINVMT	Invalid multitasking function.
E_MSINVPATTR	Invalid party attribute.
E_MSINVPTYNUM	Invalid party number.
E_MSINVPTYCNT	Invalid number of parties specified.
E_MSINVPTYTYPE	Invalid conference member type.
E_MSINVVAL	Bad global parameter value.
E_MSINVTS	Invalid time slot number specified.
E_MSMONEXT	Monitor already exists for this conference.
E_MSNOCNF	No conferencing available on device.
E_MSNODSPTS	All time slots going to the DSP are busy.

Error Returned	Description
E_MSNOFEMCH	No DCB/SC daughterboard to support this channel.
E_MSNOMON	No monitor exists for this conference.
E_MSNONCNFCH	Channel is not assigned to specified conference.
E_MSNOTS	No time slot assigned to channel.
E_MSPTYASN	Party already assigned.
E_MSTSASNCNF	Time slot already assigned to a conference.

2.3. Include Files

Function prototypes and equates are defined in *dtilib.h*, *msilib.h*, and *dcblib.h*. Applications that use the DCB/SC Windows NT library functions for DCB/SC support must include the following statements:

```
#include <windows.h>
#include “srllib.h”
#include “dtilib.h”
#include “msilib.h”
#include “dcblib.h”
```

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

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

Code that uses voice resources must include the following statements in the order shown:

```
#include <windows.h>
#include <errno.h>
#include “srllib.h”
#include “dxxxlib.h”
#include “dtilib.h”
#include “msilib.h”
#include “dcblib.h”
```

3. Function Reference

This chapter contains an alphabetical listing of the Dialogic DCB/SC library functions used to interface with the DCB/SC board. For information about Standard Attribute functions, refer to *Appendix A*.

3.1. Documentation Conventions

Each function provides the following information:

Function Header	Located at the beginning of each function. The header lists the function name, function syntax, input parameters, outputs or returns, include files, function category, and mode (asynchronous and/or synchronous). The function syntax and inputs are shown using standard C language syntax.
Description	Provides a detailed description of the function operation, including parameter descriptions.
Cautions	Provides warnings and reminders.
Example	Provides one or more C language coding examples showing how the function can be used.
Errors	Provides error code information for each function.
See Also	Provides a list of related functions.

Name:	int dcb_addtoconf(devh,confid,cdt)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	MS_CDT *cdt	• pointer to conference descriptor element
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_addtoconf( )** function adds one conferee to an existing conference. The conference identifier specifies the conference to which the conferee will be added. The **cdt** parameter describes the added conferee.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier number.
cdt:	Pointer to the conference descriptor element.

NOTES: **1.** Only one conferee can be added at a time using this function.

2. Invoking this function uses one conferencing resource each time a conferee is successfully added to a conference.

A conference descriptor element has the form of the MS_CDT structure. The format is as follows:

```
typedef struct {
          int chan_num;      /* time slot number */
          int chan_sel;      /* meaning of time slot number */
          int chan_attr;     /* attribute description */
} MS_CDT;
```

The chan_num denotes the SCbus transmit time slot number of the device to be included in the conference. The chan_sel defines the meaning of chan_num. At present, chan_sel must be set to the following value:

• MSPN_TS	SCbus time slot

The chan_attr is a bitmask describing the conferee's properties within the conference. Valid choices are:

• MSPA_NULL	No special attributes for conferee.
• MSPA_RO	Conferee participates in conference in receive-only mode.
• MSPA_TARIFF	Conferee receives periodic tone for duration of call.
• MSPA_COACH	Conferee is a coach. Coach heard by pupil only.
• MSPA_PUPIL	Conferee is a pupil. Pupil hears everyone including the coach.

Table 3. Valid Attribute Combinations

Pupil	**Coach**	**Periodic Tone**	**Receive-only mode**
			X
		X	
		X	X
	X		
X			
X		X	

NOTES: **1.** Only one coach and one pupil are allowed in a conference at any time.

2. The default MSPA_NULL must be used if channel attributes are not set.

3. Invalid attribute combinations may lead to unexpected results.

This function returns the SCbus listen time slot number for the MSPN_TS party. The number is placed in the chan_lts field of the MS_CDT structure for the MSPN_TS party. The chan_lts field is defined as follows:

```
#define   chan_lts     chan_attr
```

The chan_lts value must be used by the application to listen to the conferenced signal.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- Too many parties are specified for a single conference.
- The conference ID is invalid.
- DSP resources are not available.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{
   int  dspdevh;                       /* DCB/SC DSP device descriptor   */
   int  tsdevh1, tsdevh2, tsdevh3;     /* Time slot device descriptor */
   MS_CDT  cdt[NUM_PARTIES];           /* Conference descriptor table */
   int  confid;                        /* Conference ID */
   SC_TSINFO tsinfo;                   /* Time slot information structure */
   long  scts;                         /* SCbus time slot  */

   /* Open DCB/SC board 1, DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf( "Cannot open dcbB1D1: errno = %d\n", errno);
      exit(1);
   }

   /* Assume DCB/SC is connected to a DTI via SCbus. */

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1: errno = %d", errno);
      exit(1);
   }

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

   /* Get the SCbus transmit time slot of tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts ;       /* SCbus transmit time slot */
   cdt[0].chan_sel  = MSPN_TS ;         /* returned from dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_NULL;        /* Conferee has no special attributes */

   /* Open DTI board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2: errno = %d\n", errno);
      exit(1);
   }

   /* Get the SCbus transmit time slot of tsdevh2 */
   if  (dt_getxmitslot(tsdevh2, &tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* SCbus time slot to be conferenced */
   cdt[1].chan_num  = (int)scts;                /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;                  /* from dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_NULL;                /* Conferee has no special attributes */

   /* Establish a 2 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Open DTI board 1, time slot 3 */
   if ((tsdevh3 = dt_open("dtiB1T3",0)) == -1) {
      printf("Cannot open dtiB1T3: errno = %d", errno);
      exit(1);
   }

   /* Do a listen for tsdevh1 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for tsdevh2 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if  (dt_listen(tsdevh2,&tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
```

```
}

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

/* Get the SCbus transmit time slot of tsdevh3 */
if  (dt_getxmitslot(tsdevh3, &tsinfo) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh3));
   exit(1);
}

/* Add another conferee to conference */
cdt[0].chan_num  = (int)scts;        /* scts is the time slot */
cdt[0].chan_sel  = MSPN_TS;  /* returned from getxmitslot*/
cdt[0].chan_attr = MSPA_COACH;

if (dcb_addtoconf(dspdevh, confid, &cdt) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
   exit(1);
}
else
   printf("Party added to conference\n");

/* Do a listen for tsdevh3 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

if (dt_listen(tsdevh3,&tsinfo) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh3));
   exit(1);
}

/* Continue processing */

/* Unlisten the time slots */
if (dt_unlisten(tsdevh1) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

if (dt_unlisten(tsdevh2) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

if (dt_unlisten(tsdevh3) == -1) {
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh3));
   exit(1);
}

/* Delete the conference */
if (dcb_delconf(dspdevh, confid) == -1) {
   printf("Cannot delete conference %d. Error Message = %s", confid,
          ATDV_ERRMSGP(dspdevh));
   exit(1);
}

if (dt_close(tsdevh1) == -1) {
   printf("Error closing tsdevh1\n");
   exit(1);
}
```

```
   if (dt_close(tsdevh2) == -1) {
      printf("Error closing tsdevh2\n");
      exit(1);
   }

if  (dt_close(tsdevh3) == -1) {
      printf("Error closing tsdevh3\n");
      exit(1);
   }

   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D1. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_delconf( )**
- **dcb_estconf( )**
- **dcb_remfromconf( )**

Name:	int dcb_close(devh)	
Inputs:	int devh	• DCB/SC device handle
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h errno.h	
Category:	Device Management	
Mode:	synchronous	

■ Description

The **dcb_close( )** function closes the DCB/SC device previously opened by **dcb_open( )**. The devices are either DCB/SC boards or DSPs on the board. The **dcb_close( )** function releases the handle.

Parameter	Description
devh:	The valid DCB/SC device handle returned by a call to **dcb_open( )**.

■ Cautions

1. This function fails if the device handle is invalid.
2. The **dcb_close( )** function affects only the link between the calling process or thread and the device. Other processes or threads are unaffected by **dcb_close( )**.
3. If event notification is active for the device to be closed, call the SRL **sr_dishdlr( )** function prior to calling **dcb_close( )**.
4. A call to **dcb_close( )** does not affect the configuration of the DCB/SC.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

main()
{
   int dspdevh;             /* DSP device descriptor variable */

   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open DSP dcbB1D2 : errno = %d", errno);
      exit(1);
   }

   /*  Continue Processing  */

   /* Done processing - close device */
   if (dcb_close(dspdevh) == -1) {
      printf("Cannot close DSP dcbB1D2 : errno = %d", errno);
      exit(1);
   }
}
```

■ Errors

The **dcb_close( )** function does not return errors in the standard return code format. If an error occurred during the **dcb_close( )** call, a -1 will be returned, and the specific error number will be returned in the **errno** global variable.

■ See Also

- **dcb_open( )**

Name:	int dcb_delconf(devh, confid)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_delconf( )** function deletes a previously established conference. The conference ID specifies the conference to be deleted.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The DCB/SC conference identifier.

NOTES: **1.** Calling this function frees all resources in use by the conference.

2. Dialogic recommends that the appropriate **xx_unlisten( )** function be called for each conferee before **dcb_delconf( )** is called.

■ Cautions

This function fails when:

- The specified device handle is invalid.
- The conference ID is invalid.

■ Example

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

```
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{
   int  dspdevh;                       /* DSP device descriptor variable */
   int  tsdevh1, tsdevh2;              /* Time slot device descriptors */
   MS_CDT  cdt[NUM_PARTIES];           /* Conference descriptor table */
   int  confid;                        /* Conference ID */
   SC_TSINFO tsinfo;                   /* Time slot information structure */
   long scts;                          /* SCbus time slot */


   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open dcbB1D2 : errno = %d", errno);
      exit(1);
   }

   /* Assume DCB/SC connected to a DTI via SCbus. */

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1: errno = %d", errno);
      exit(1);
   }


   /* Get SCbus transmit time slot of tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts ;             /* SCbus time slot returned */
   cdt[0].chan_sel  = MSPN_TS;                /* from dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_NULL;              /* Conferee has no special attributes */

   /* Open DTI board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2 : errno = %d", errno);
      exit(1);
   }

   /* Get SCbus transmit time slot of tsdevh2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /*  Set up CDT structure */
   cdt[1].chan_num  = (int)scts;       /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;         /* from dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_TARIFF;     /* Conferee receives periodic tariff tone */

   /* Establish a 2 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for time slot tsdevh1 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for the time slot tsdevh2 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Continue Processing */

   /* Unlisten the time slots */
   if (dt_unlisten(tsdevh1) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if (dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
            ATDV_ERRMSGP(dspdevh));
      exit(1);
   }
   else
      printf("Conference deleted\n");

   /* Done processing - close all open devices */
   if(dcb_close(dspdevh) == -1) {
      printf("Cannot close dcbB1D2 : errno = %d", errno);
      exit(1);
   }

   if(dt_close(tsdevh2) == -1) {
      printf("Cannot close dtiB1T2 : errno = %d", errno);
      exit(1);
   }

   if(dt_close(tsdevh1) == -1) {
      printf("Cannot close dtiB1T1 : errno = %d", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_addtoconf( )**
- **dcb_estconf( )**
- **dcb_remfromconf( )**

Name:	int dcb_dsprescount(devh,valuep)	
Inputs:	int devh	• DCB/SC DSP device handle
	int * valuep	• integer pointer to where the free DSP resource count is returned
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Configuration	
Mode:	synchronous	

■ Description

The **dcb_dsprescount( )** function returns the available conferencing resource count for a specified DSP. Each DSP has 32 resources managed by the application.

Parameter	Description
devh:	The DCB/SC DSP device handle.
valuep:	Integer pointer to where the free DSP resource count is returned.

Calling any of the following functions will cause the available resource count to change:

Function	Condition
dcb_estconf( )	Uses the total number of parties in the new conference.
dcb_addtoconf( )	One resource will be used every time a conferee is added to a conference.
dcb_remfromconf( )	Frees one resource.
dcb_delconf( )	Frees all resources in use by the conference, including the monitor.
dcb_monconf( )	Uses one resource.
dcb_unmonconf( )	Frees one resource.

■ Cautions

This function fails when the device handle specified is invalid.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

main()
{
   int  dspdevh;    /* DCB/SC DSP device descriptor */
   int  count;      /* DSP resource count */

   /* Open DCB/SC board 1, DSP 2 */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open dcbB1D2: errno = %d\n", errno);
      exit(1);
   }

   /* Get unused conference-resource count for dspdevh */
   if (dcb_dsprescount(dspdevh, &count) == -1) {
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }
   else
      printf("Free DSP Resource count = %d\n", count);

   if (dcb_close(dspdevh) == -1) {
      printf("Cannot close dcbB1D2 : %s/n”, ATDV.ERRMSGP (dspdevh)),
        exit(1);
    }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_addtoconf( )**
- **dcb_estconf( )**
- **dcb_delconf( )**
- **dcb_monconf( )**
- **dcb_remfromconf( )**
- **dcb_unmonconf( )**

Name:	int dcb_estconf(devh,cdt,numpty,confattr,confid)	
Inputs:	int devh	• DCB/SC DSP device handle
	MS_CDT *cdt	• pointer to conference descriptor table
	int numpty	• number of parties in conference
	int confattr	• conference attributes
	int *confid	• pointer to the conference identifier
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_estconf( )** function establishes a conference of up to four parties. A conference is associated with a DSP and all DSP resources used by the conference are on that DSP. When **dcb_estconf( )** returns successfully, **confid** will contain the conference identification number for use in all further modifications to that conference.

Parameter	Description
devh:	The DCB/SC DSP device handle.
cdt:	Pointer to the conference descriptor table.
numpty:	Number of parties in the conference.
confattr:	The conference attributes.
confid:	Pointer to the conference ID number.

The conference descriptor table is an array of the MS_CDT structure. The MS_CDT structure has the following format:

```
typedef struct {
          int chan_num;      /* SCbus time slot number */
          int chan_sel;      /* meaning of time slot number */
          int chan_attr;     /* attribute description */
} MS_CDT;
```

The chan_num field denotes the SCbus transmit time slot number of the device to be included in the conference. The chan_sel defines the meaning of the chan_num. At present, chan_sel must be set to the following value:

• MSPN_TS	SCbus time slot number

chan_attr is a bitmask describing the conferee's properties within the conference. Valid choices are:

• MSPA_NULL	No special attributes for conferee. The conferee hears everyone except the coach.
• MSPA_RO	Conferee participates in conference in receive-only mode.
• MSPA_TARIFF	Conferee receives periodic tone for duration of call.
• MSPA_COACH	Conferee is a coach. Coach is heard by pupil only.
• MSPA_PUPIL	Conferee is a pupil. Pupil hears everyone including coach.

Table 4. Valid Attribute Combinations

Pupil	**Coach**	**Periodic Tone**	**Receive-only mode**
			X
		X	
		X	X
	X		
X			
X		X	

NOTES: **1.** Only one coach and one pupil are allowed in a conference at any time. Specifying more than one of either will cause unexpected results.

2. The default MSPA_NULL must be used if channel attributes are not specified.

3. Invalid attribute combinations may lead to unexpected results.

The conference attribute parameter, conf_attr, is a bitmask describing the properties of the conference. The properties are in effect for all parties in the conference.

• MSCA_ND	All parties in conference are notified by a tone if another conferee is added or removed from a conference.
• MSCA_NN	If MSCA_ND is set, do not notify conferees if a conferee joins the conference in receive-only mode or as a monitor.
• MSCA_NULL	No special attributes.

For MSCA_NN to have an effect, it should be ORed together with MSCA_ND.

The default MSCA_NULL must be used if the conference attribute is not specified.

For each member of a conference, the number of the SCbus time slot to listen to is returned in the chan_lts field of MS_CDT. The chan_attr field in the CDT structure is redefined as follows:

```
#define    chan_lts                  chan_attr
```

NOTES: **1.** Calling this function causes **numpty** resources to be used when the conference is successfully established.

2. This function may be used to establish a conference of up to 4 parties. You must use **dcb_addtoconf( )** or **dcb_monconf( )** to increase the size of the conference.

■ Cautions

This function fails when:

- An invalid device handle is specified.
- DSP resources are not available.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define   NUM_PARTIES   2

main()
{
   int  dspdevh;                        /* DCB/SC DSP Device handle variable */
   int  tsdevh1, tsdevh2;               /* Time slot device handles */
   MS_CDT  cdt[NUM_PARTIES];            /* Conference descriptor table */
   SC_TSINFO tsinfo;                    /* Time slot information structure */
   int  confid;                         /* Conference ID */
   long scts;                           /* SCbus time slot */

   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2", 0) == -1) {
      printf("Cannot open dcbB1D2. errno = %d", errno);
      exit(1);
   }

   /* Open network board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1", 0)) == -1) {
      printf( "Cannot open dtiB1T1: errno=%d", errno);
      exit(1);
   }

   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts  ;    /* scts is the time slot... */
   cdt[0].chan_sel  = MSPN_TS    ;    /* ...returned from getxmitslot() */
   cdt[0].chan_attr = MSPA_TARIFF;

   /* Open board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2", 0)) == -1) {
      printf( "Cannot open dtiB1T2: errno=%d", errno);
      exit(1);
```

```
}


if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* SCbus time slot to be conferenced */
cdt[1].chan_num  = (int)scts ;     /* scts is the time slot... */
cdt[1].chan_sel  = MSPN_TS   ;     /* ...returned from getxmitslot() */
cdt[1].chan_attr = MSPA_PUPIL;     /* Conferee may be coached later */

/* Establish conference */
if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* Do a listen() for the tsdevh1 to its conference signal */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

if (dt_listen(tsdevh1,&tsinfo) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Do a listen() for the tsdevh2 to its conference signal */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

if (dt_listen(tsdevh2,&tsinfo) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/*
 * Continue processing
 */

/* Unlisten the time slots */
if (dt_unlisten(tsdevh1) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

if (dt_unlisten(tsdevh2) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

if (dt_close(tsdevh1) == -1){
   printf("Error closing tsdevh1\n");
   exit(1);
}
 if (dt_close(tsdevh2) == -1){
    printf("Error closing tsdevh2\n");
    exit(1);
}

/* Delete the conference */
if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
           ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Done Processing - Close device */
   if(dcb_close(dspdevh) == -1) {
      printf("Cannot close DSP dcbB1D2. errno = %d", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

■ See Also

- **dcb_addtoconf( )**
- **dcb_delconf( )**
- **dcb_monconf( )**
- **dcb_remfromconf( )**
- **dcb_unmonconf( )**

Name:	int dcb_evtstatus(event, action, status)	
Inputs:	unsigned char event	• event ID
	unsigned char action	• action to be performed
	unsigned char *status	• pointer to status of event-generation
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Auxiliary	
Mode:	synchronous	

■ Description

The **dcb_evtstatus( )** function gets or sets the status of a process-wide event. Certain features of the Dialogic Audio Conferencing software are board-level features in that they are enabled or disabled on a per board basis. Process-wide events are enabled or disabled once for all devices used by that process.

Parameter	Description
event:	The specified process-wide event.
action:	Specifies whether the event status is to be set or retrieved.
status:	If the event status is being set, ON or OFF is passed to the function in this parameter. If the event status is being retrieved, this parameter will contain ON or OFF when the function returns.

event must be set to MSG_RESTBL. MSG_RESTBL controls the resource table update event generation. The resource assignment table is the mapping of resources to conferees. Anytime there is a change in this mapping and event generation is enabled, a DCBEV_CTU event is generated. Associated with this event is the resource table. An application may be in the resource table implementing active talkers. When this event notification is enabled, and the application makes a change to the assignment of conferencing resources on a

DCB/SC board, a DCBEV_CTU event will be generated. The updated resource table will be returned as the event data. Refer to the code example for details.

action may be set to SET_EVENT or GET_EVENT.

The **status** parameter may contain ON or OFF.

To enable an event handler for a specified event, follow these steps:

1. Call **sr_enbhdlr( )**. This function specifies the event and the application defined event handler that is called from a signal handler.
2. Call **dcb_evtstatus( )**. This function sets the digit message mask.

NOTE: The request for an event to be posted to an event handler must be specified using both the **sr_enbhdlr( )** and **dcb_evtstatus( )** functions.

■ Cautions

dcb_evtstatus( ) is a process-wide function and does not have a Dialogic device-handle as one of its parameters. Any event set ON or OFF is set for all devices used by the process, not for any particular device.

■ Example

```
#include <windows.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"


#define   MAX_PTY     32
#define   TABLE_SIZE 192

DCB_CT res_table[MAX_PTY];    /* DCB_CT structure array */

void handler()
{
   int   size = sr_getevtlen();
   char *datap = (char *)sr_getevtdatap();
   int   event_type = (int)sr_getevttype();

   printf("Event occurred on %s : Data size = %d : Data is at 0x%x\n",
           ATDV_NAMEP(sr_getevtdev()), size, datap);

   if (event_type == DCBEV_CTU) {
      memcpy(res_table, datap, TABLE_SIZE);
   }
   else {
      for (i=0; i<size; i++){
         printf("%d\n", *(datap++));
      }
   }
}

main()
{
   int  bddevh, dspdevh;       /* DCB/SC board and DSP device descriptors */
   unsigned long atibits;      /* Active talker bits     */
   int  mode = SR_POLLMODE;    /* SRL function-call mode */
   unsigned int status;        /* DCB/SC Feature status */
   unsigned int i, count = 1000; /* Loop counters */

   /* Open DCB/SC board device */
   if ((bddevh = dcb_open("dcbB1",0)) == -1) {
      printf("Cannot open dcbB1. errno = %d", errno);
      exit(1);
   }

   /* Set SRL function call mode */
   if (sr_setparm(SRL_DEVICE, SR_MODEID, (void *)&mode) == -1){
       printf("Error setting sr_setparm()\n");
       exit(1);
   }

   /* Enable SRL event handler */
   if (sr_enbhdlr(EV_ANYDEV, EV_ANYEVT, (void *)handler) == -1) {
       printf("Error setting sr_enbhdlr()\n");
       exit(1);
   }

   /* Set Active Talker On */
   status = ACTID_ON;

   if (dcb_setbrdparm(bddevh, MSG_ACTID, (void *)&status) == -1) {
      printf("Error setting board parameter - %s\n", ATDV_ERRMSGP(bddevh));
      exit(1);
   }

   /* Done with board-level calls : close device */
   if (dcb_close(bddevh) == -1) {
      printf("Cannot close dcbB1. errno = %d\n", errno);
      exit(1);
   }


   /* Set Resource Assignment Table Update event ON */
   status = ON;

   if (dcb_evtstatus(MSG_RESTBL, SET_EVENT, &status) == -1) {
      printf("Error enabling system-wide event\n");
      exit(1);
   }

   /* Open board 1, DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf("Cannot open dcbB1D1. errno = %d", errno);
      exit(1);
   }
   /* Establish a conference and continue processing */

   /* Wait in a 1000-count loop to get the active talkers */

   while (count--) {
      if (dcb_getatibits(dspdevh, &atibits) == -1){
         printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
         exit(1);
      }
      printf("ATIBITS = %d\n", atibits);

      for (i=0; i<32; i++){
         if (atibits & (1<<i)){
            printf("confid = %d, TimeSlot = %d, Selector = %d\n",
                    res_table[i].confid, res_table[i].chan_num,
                    res_table[i].chan_sel);
         }
      } /* End of for() loop */
   } /* End of while() loop */

   /* Set Resource Table Update events OFF */
   status = OFF;

   if (dcb_evtstatus(MSG_RESTBL, SET_EVENT, &status) == -1) {
      printf("Error enabling system-wide event\n");
      exit(1);
   }

   /* Disable event handler */
   if (sr_dishdlr(EV_ANYDEV, DCBEV_CTU, (void *)handler) == -1) {
       printf("Error in sr_dishdlr()\n");
       exit(1);
   }

   /* Done processing - close DSP device */

   if (dcb_close(dspdevh) == -1) {
      printf("Cannot close dcbB1D1. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

■ See Also

- **dcb_gettalkers( )**

Name:	int dcb_getatibits(devh,atibits)	
Inputs:	int devh	• DCB/SC DSP device handle
	unsigned long *atibits	• pointer to active talker indicator bits
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Auxiliary	
Mode:	synchronous	

■ Description

The **dcb_getatibits( )** function returns the active talker indicator bits set at the time this function is called. This function is provided to give more control to the application and must be used with the resource assignment table update event to implement active talker retrieval.

Parameter	Description
devh:	The valid DSP device handle returned by a call to **dcb_open( )**.
atibits:	Pointer to location where active talker indicator bits will be stored.

NOTE: The snapshot of information provided by **dcb_getatibits** is accurate for a split second. This information may not be accurate by the time the application processes it.

atibits is 4 bytes long. Each bit corresponds to one DSP resource and is updated every 100 ms. Bits are turned on or off by the DSP based on the speaker energy (volume) level.

Bit 0 of **atibits** corresponds to resource 0 on the specified DSP. **atibits** should be used with the resource assignment table. The table is copied from the event data returned once the current DCBEV_CTU event is detected by the application.

DCBEV_CTU is enabled using the **dcb-evtstatus( )** function. Bit *n* should be paired with the *n*th DCB_CT element of the resource table.

■ Cautions

This function fails when the device handle is invalid.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"


#define   MAX_PTY     32
#define   TABLE_SIZE 192

DCB_CT res_table[MAX_PTY];    /* DCB_CT structure array */

void handler()
{
   int   size = sr_getevtlen();
   char *datap = (char *)sr_getevtdatap();
   int   event_type = (int)sr_getevttype();

   printf("Event occurred on %s : Data size = %d : Data is at 0x%x\n",
           ATDV_NAMEP(sr_getevtdev()), size, datap);

   if (event_type == DCBEV_CTU) {
      memcpy(res_table, datap, TABLE_SIZE);
   }
   else {
      for (i=0; i<size; i++){
         printf("%d\n", *(datap++));
      }
   }
}

main()
{
   int  bddevh, dspdevh;       /* DCB/SC board and DSP device descriptors */
   unsigned long atibits;      /* Active talker indicator bits     */
   int  mode = SR_POLLMODE;    /* SRL function-call mode */
   unsigned int status;        /* DCB/SC Feature status */
   unsigned int i, count = 1000; /* Loop counters */

   /* Open DCB/SC board device */
   if ((bddevh = dcb_open("dcbB1",0)) == -1) {
      printf("Cannot open dcbB1. errno = %d", errno);
      exit(1);
   }
   /* Set SRL function call mode */
   if (sr_setparm(SRL_DEVICE, SR_MODEID, (void *)&mode) == -1){
       printf("Error setting sr_setparm()\n");
       exit(1);
   }

   /* Enable SRL event handler */
   if (sr_enbhdlr(EV_ANYDEV, EV_ANYEVT, (void *)handler) == -1) {
       printf("Error setting sr_enbhdlr()\n");
       exit(1);
   }

  /* Set Active Talker On */
   status = ACTID_ON;

   if (dcb_setbrdparm(bddevh, MSG_ACTID, (void *)&status) == -1) {
      printf("Error setting board parameter - %s\n", ATDV_ERRMSGP(bddevh));
      exit(1);
   }

   /* Done with board-level calls : close device */
   if (dcb_close(bddevh) == -1) {
      printf("Cannot close dcbB1. errno = %d\n", errno);
      exit(1);
   }


   /* Set Resource Table Update events ON */
   status = ON;

   if (dcb_evtstatus(MSG_RESTBL, SET_EVENT, &status) == -1) {
      printf("Error enabling system-wide event\n");
      exit(1);
   }

   /* Open board 1, DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf("Cannot open dcbB1D1. errno = %d", errno);
      exit(1);
   }

   /* Establish a conference and continue processing */

   /* Wait in a 1000-count loop to get the active talkers */

   while (count--) {
      if (dcb_getatibits(dspdevh, &atibits) == -1){
         printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
         exit(1);
      }
      printf("ATIBITS = %d\n", atibits);

      for (i=0; i<32; i++){
         if (atibits & (1<<i)){
            printf("confid = %d, TimeSlot = %d, Selector = %d\n",
                    res_table[i].confid, res_table[i].chan_num,
                    res_table[i].chan_sel);
         }
      } /* End of for() loop */
   } /* End of while() loop */

   /* Set Resource Table Update events OFF */
   status = OFF;
   if (dcb_evtstatus(MSG_RESTBL, SET_EVENT, &status) == -1) {
      printf("Error enabling system-wide event\n");
      exit(1);
   }

   /* Disable event handler */
   if (sr_dishdlr(EV_ANYDEV, DCBEV_CTU, (void *)handler) == -1) {
       printf("Error in sr_dishdlr()\n");
       exit(1);
   }

   /* Done processing - close DSP device */

   if (dcb_close(dspdevh) == -1) {
      printf("Cannot close dcbB1D1. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

■ See Also

- **dcb_evtstatus( )**
- **dcb_gettalkers( )**

Name:	int dcb_getbrdparm(devh,param,valuep)	
Inputs:	int devh	• DCB/SC board device handle
	unsigned char param	• device parameter defined name
	void * valuep	• pointer to returned parameter value
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Configuration	
Mode:	synchronous	

■ Description

The **dcb_getbrdparm( )** function retrieves a DCB/SC board parameter value. Each parameter has a symbolic name that is defined in *dcblib.h*. The parameters are explained in the **dcb_setbrdparm( )** function description.

Parameter	Description
devh:	The valid board device handle returned by a call to **dcb_open( )**.
param:	The parameter to be examined.
valuep:	Pointer to the integer or DCB_VOL structure for storage of the parameter value.

The valid values for **param** and **valuep** are shown below:

Parameter	Description
MSG_ACTID	Indicates Active Talker feature status. Possible values are ACTID_ON or ACTID_OFF. ACTID_OFF is the default.
MSG_VOLDIG	Defines the volume control status and volume up/down/reset digits.

The following structure contains the volume control status and digits:

```
typedef  struct DCB_VOL{
         unsigned char vol_control;
         unsigned char vol_up;
         unsigned char vol_reset;
         unsigned char vol_down;
}DCB_VOL;
```

The digits for volume control are not received by the application. The vol_control field indicates whether the volume control is turned on or off. The vol_up field indicates the digit used for increasing the volume level. vol_down indicates the digit used for decreasing the volume. vol_reset indicates the digit used to set the volume to its default level. By default, vol_control is OFF.

■ Cautions

1. The value of the parameter returned by this function is currently an integer or an DCB_VOL structure. **valuep** is the address of the value, but should be cast as a void pointer when passed in the value field.
2. This function fails when:
   - The device handle is invalid.
   - The parameter specified is invalid.
   - The DSP device handle is used.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

main()
{
   int  brddevh;      /* Board dev descriptor variables */
   int  actid_status; /* Active talker status (ON/OFF)  */
   DCB_VOL  volume;    /* Volume control structure */

   /* Open DCB board 1 */
   if ((brddevh = dcb_open("dcbB1",0)) == -1) {
      printf( "Cannot open dcbB1: errno = %d\n", errno);
      exit(1);
   }

   /* Retrieve Status (ON/OFF) of the Active Talker Feature */
   if (dcb_getbrdparm(brddevh, MSG_ACTID, (void *)&actid_status) == -1) {
      printf("Error getting board param:0x%x\n ", ATDV_LASTERR(brddevh));
      exit(1);
   }
   printf("Active talker identification is %s\n", (actid_status ? "ON" : "OFF"));

   /* Retrieve Information on Volume Control Feature */
   if (dcb_getbrdparm(brddevh, MSG_VOLDIG,(void *)&volume) == -1) {
      printf("Error getting volume control parameters : 0x%x\n ",
             ATDV_LASTERR(brddevh));
      exit(1);
   }
   printf("Volume Control is %s\n", (volume.vol_control ? "ON" : "OFF"));
   printf("The Up Digit is       %d\n", volume.vol_up);
   printf("The Reset Digit is    %d\n", volume.vol_reset);
   printf("And the Down Digit is %d\n", volume.vol_down);

   /* Continue processing */

   if (dcb_close(brddevh)== -1){
      printf("Cannot close dcbB1. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_setbrdparm( )**

Name:	int dcb_getcde(devh,confid,cdt)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	MS_CDT *cdt	• pointer to conference descriptor table element
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_getcde( )** function retrieves the attributes of a conferee in an existing conference.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.
cdt:	Pointer to the conference descriptor table element.

This function requires that the conferee's chan_num and chan_sel be specified. On successful completion, the conference party's attribute will be returned in the char_attr field of the MS_CDT structure.

The conference descriptor table is an array of the MS_CDT structure. The MS_CDT structure has the following format:

```
typedef struct {
          int chan_num;      /* time slot number */
          int chan_sel;      /* meaning of time slot number */
          int chan_attr;     /* attribute description */
} MS_CDT;
```

The chan_num denotes the SCbus transmit time slot number of the device. The chan_sel defines the meaning of chan_num. At present, chan_sel should be set to the following value:

• MSPN_TS	SCbus time slot number

Channel attribute is a bitmask describing the conferee's properties within the conference. Possible returns are listed in the table below. A combination of any of the attributes shown in the table may be returned.

Table 5. Possible Returns for Channel Attribute

Channel Attribute	**Description**
MSPA_NULL	No special attributes for conferee. The conferee hears everyone except the coach.
MSPA_RO	Conferee participates in the conference in receive-only mode.
MSPA_TARIFF	Conferee receives periodic tone for duration of call.
MSPA_COACH	Conferee is a coach. Coach is heard by pupil only.
MSPA_PUPIL	Conferee is a pupil. Pupil hears everyone including coach.

NOTE: This function must be invoked multiple times if the attributes of more than one conferee are desired.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- An invalid conference ID is specified.
- The queried conferee is not in the conference.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{
   int    dspdevh;            /* DCB/SC DSP device handle */
   int    tsdevh1, tsdevh2;   /* DTI time slot device handles */
   MS_CDT cdt[NUM_PARTIES];   /* Conference descriptor table */
   int    confid;             /* Conference ID */
   int    attrib;             /* Time slot attribute */
   long   scts1, scts2;       /* SCbus time slots */

   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open dcbB1D2: errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1: errno = %d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts1;

   /* Retrieve the SCbus transmit time slot for tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up the CDT structure */
   cdt[0].chan_num  = (int)scts1;        /* scts is the SCbus transmit time slot */
   cdt[0].chan_sel  = MSPN_TS;          /* returned from dt_getxmitslot(). */
   cdt[0].chan_attr = MSPA_TARIFF;      /* Conferee will receive periodic tariff tone */

   /* Open DTI board 1, tslot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2: errno = %d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts2;

   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }
```

```
/* SCbus time slot to be conferenced */
cdt[1].chan_num  = (int)scts2;      /* scts is the SCbus transmit time slot */
cdt[1].chan_sel  = MSPN_TS;        /* returned from dt_getxmitslot(). */
cdt[1].chan_attr = MSPA_PUPIL;     /* The conferee may be coached later */

/* Establish the two party conference */
if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* Do a listen to the conference signal for tsdevh1 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

if (dt_listen(tsdevh1,&tsinfo) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Do a listen to the conference signal for tsdevh2 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

if (dt_listen(tsdevh2,&tsinfo) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Now get the attribute of conferee on tsdevh2 */
cdt[0].chan_num = (int)scts2;
cdt[0].chan_sel = MSPN_TS;

if(dcb_getcde(dspdevh, confid, &cdt)==-1){
  printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
  exit(1);
}
printf ("%s has conferee attribute 0x%x\n", ATDV_NAMEP(tsdevh2), cdt[0].chan_attr);

/* Finished with conference, so remove listens */

if (dt_unlisten(tsdevh1) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

if (dt_unlisten(tsdevh2) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Delete the conference */
if(dcb_delconf(dspdevh, confid) == -1) {
   printf("Cannot delete conference %d. Error Message = %s", confid,
           ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* And close all open devices */
if (dt_close(tsdevh1) == -1){
   printf("Error (0x%x) closing tsdevh1\n", errno);
   exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error (0x%x) closing tsdevh2\n", errno);
      exit(1);
   }

   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D2 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

■ See Also

- **dcb_setcde( )**

Name:	int dcb_getcnflist(devh,confid,numpty,cdt)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	int *numpty	• pointer to the conferee count
	MS_CDT *cdt	• pointer to conference descriptor table
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Conference	
Mode:	synchronous	

■ Description

The **dcb_getcnflist( )** function retrieves a conferee list and the total number of parties within a conference. The list contains specific information about each conferee in that conference. The information includes each conferee's SCbus transmit time slot number, selector, and conferee attribute description.

NOTE: The list is not returned in any specified order.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.
numpty:	Pointer to the conferee count.
cdt:	Pointer to the conference descriptor table.

NOTE: The application is responsible for allocating an MS_CDT table with sufficient elements. Refer to the sample code.

When a conference is being monitored, one member of the conference list will be the monitor. chan_num for the monitor will equal 0x7FFF and chan_sel will be MSPN_TS.

The conference descriptor table is an array of the MS_CDT structure. The MS_CDT structure has the following format:

```
typedef struct {
        int chan_num;      /* time slot number */
        int chan_sel;      /* meaning of time slot number */
        int chan_attr;     /* attribute description */
} MS_CDT;
```

■ Cautions

1. This function fails when an invalid conference ID is specified.
2. It is the responsibility of the application to allocate enough memory for the conference descriptor table. There must be an MS_CDT element allocated for each conferee description returned by this function. For example, if a conference was started with four conferees, and three conferees were added later, the MS_CDT array must be able to hold seven entries.

   NOTE: Even though **dcb_monconf( )** does not use the CDT structure, the array must have an additional structure if the conference is being monitored.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{

   int  dspdevh;            /* DCB/SC DSP device handle variable */
   int  tsdevh1, tsdevh2;   /* DTI SCbus time slot device handles */
   int  partycnt;           /* Pointer to the number of conferenced parties */
   MS_CDT  cdt[32];         /* Conference descriptor table */
   SC_TSINFO tsinfo;
   int  confid;             /* Conference ID */
   long  scts;              /* Returned SCbus time slot */
   int  i;                  /* Loop index */

   /* Open board 1 DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf( "Cannot open dcbB1D1: error = %d", errno);
      exit(1);
   }
```

```
/* Assume the DCB/SC connected to a DTI via SCbus. */

/* Open board 1, tslot 1 */
if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
    printf( "Cannot open dtiB1T1: errno=%d", errno);
    exit(1);
}

/* Prepare the SCbus time slot information structure */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &scts;


/* Retrieve the SCbus transmit time slot for tsdevh1 */
if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Set up the CDT structure */
cdt[0].chan_num  = (int)scts;       /* scts is the SCbus time slot */
cdt[0].chan_sel  = MSPN_TS;         /* returned from dt_getxmitslot() */
cdt[0].chan_attr = MSPA_NULL;       /* Conferee has no special attributes */

/* Open board 1, tslot 2 */
if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
   printf( "Cannot open dtiB1T2: errno=%d", errno);
   exit(1);
}

/* Retrieve the SCbus transmit time slot for tsdevh2 */
if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
   printf("Error Message = %s" , ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* SCbus time slot to be conferenced */
cdt[1].chan_num  = (int)scts;       /* scts is the SCbus time slot */
cdt[1].chan_sel  = MSPN_TS;         /* returned from dt_getxmitslot() */
cdt[1].chan_attr = MSPA_PUPIL;      /* Conferee may be coached later */

/* Establish 2 party conference */
if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* Do a listen to the SCbus listen time slot for tsdevh1 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

if (dt_listen(tsdevh1,&tsinfo) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Do a listen to the SCbus listen time slot for tsdevh2 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

if (dt_listen(tsdevh2,&tsinfo) == -1){
   printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
```

```
      exit(1);
   }

   /* Get conferee list */
   if (dcb_getcnflist(dspdevh, confid, &partycnt, &cdt[0]) == -1) {
      printf("Error Message = %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Display conference information */
   printf("Number of parties in conference %d = %d\n", confid, partycnt);

   for (i=0; i<partycnt; i++){
       printf("%d : Chan_num = 0x%x", i+1, cdt[i].chan_num);
       printf("     Chan_sel = 0x%x",      cdt[i].chan_sel);
       printf("     Chan_att = 0x%x\n",    cdt[i].chan_attr);
   }

   /* Remove all listens */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d : Error Message = %s", confid,
              ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error (0x%x) closing tsdevh1\n", errno);
      exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error (0x%x) closing tsdevh2\n", errno);
      exit(1);
   }

   if (dcb_close(dspdevh)== -1){
      printf("Cannot close dcbB1D1 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_estconf( )**

Name:	int dcb_getdigitmsk(devh,confid,bitmaskp)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	unsigned int * bitmaskp	• pointer to digit bitmask
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Configuration	
Mode:	synchronous	

■ Description

The **dcb_getdigitmsk( )** function returns the digit mask for a specified conference. The values set in the mask corresponds to the digits which, when received, will cause a DCBEV_DIGIT event to be generated to the application.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.
bitmask:	Pointer to the digit bitmask.

NOTE: If MSG_VOLDIG is enabled to give transparent volume control to the conferees, the digits for volume increase, decrease, and reset will not cause digit events to be generated. As a result, the application will not know if the volume changes.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- An invalid conference ID is specified.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define   NUM_PARTIES   2

main()
{
   int  dspdevh;             /* DCB/SC DSP device handle */
   int  confid;              /* Conference Identifier */
   unsigned int bitmask;     /* bitmask variable */
   int  tsdevh1, tsdevh2;    /* DTI time slot device handles */
   MS_CDT cdt[NUM_PARTIES];   /* Conference descriptor table */
   SC_TSINFO tsinfo;
   long scts;                 /* SCbus time slot */

   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open dcbB1D2 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1: errno=%d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Retrieve the SCbus transmit time slot for tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up the CDT[0] structure */
   cdt[0].chan_num  = (int)scts;   /* scts is the SCbus time slot */
   cdt[0].chan_sel  = MSPN_TS;     /* returned from dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_TARIFF; /* Party receives periodic tariff tone */

   /* Open DTI board 1, tslot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2: errno=%d", errno);
      exit(1);
   }

   /* Retrieve the SCbus transmit time slot for tsdevh2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Set up the CDT[1] structure */
   cdt[1].chan_num  = (int)scts;           /* scts is the SCbus time slot */
   cdt[1].chan_sel  = MSPN_TS;             /* returned from dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_PUPIL;          /* Conferee may be coached later */

   /* Establish a two party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for the tsdevh1 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for the tsdevh2 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Enable digit detection for digits 1,3 and 5 only */

      if (dcb_setdigitmsk(dspdevh, confid, CBMM_ONE | CBMM_THREE | CBMM_FIVE,
          CBA_SETMSK)) == -1) {
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Get the bitmask value for the digit detection event */
  if (dcb_getdigitmsk(dspdevh, confid, &bitmask) == -1) {
     printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
     exit(1);
   }
    /*
    *   Display list of digits enabled for detection
    */
   if (bitmask & CBMM_ZERO)
     printf("Digit 0 is enabled\n");
   if (bitmask & CBMM_ONE)
      printf("Digit 1 is enabled\n");
   if (bitmask & CBMM_TWO)
      printf("Digit 2 is enabled\n");
   if (bitmask & CBMM_THREE)
      printf("Digit 3 is enabled\n");
   if (bitmask & CBMM_FOUR)
      printf("Digit 4 is enabled\n");
   if (bitmask & CBMM_FIVE)
      printf("Digit 5 is enabled\n");
   if (bitmask & CBMM_SIX)
      printf("Digit 6 is enabled\n");
   if (bitmask & CBMM_SEVEN)
      printf("Digit 7 is enabled\n");
   if (bitmask & CBMM_EIGHT)
      printf("Digit 8 is enabled\n");
   if (bitmask & CBMM_NINE)
```

```
      printf("Digit 9 is enabled\n");
   if (bitmask & CBMM_STAR)
      printf("Digit * is enabled\n");
   if (bitmask & CBMM_POUND)
      printf("Digit # is enabled\n");
   if (bitmask & CBMM_A)
      printf("Digit A is enabled\n");
   if (bitmask & CBMM_B)
      printf("Digit B is enabled\n");
   if (bitmask & CBMM_C)
      printf("Digit C is enabled\n");
   if (bitmask & CBMM_D)
      printf("Digit D is enabled\n");

   /* Unlisten the time slots */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
           ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Done Processing - Close all open devices */

   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }

   if(dcb_close(dspdevh) == -1) {
     printf("Cannot close dcbB1D2 : errno = %d\n", errno);
     exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_setdigitmsk( )**
- **dcb_setbrdparm( )**

Name:	int dcb_gettalkers(devh,confid,numpty,talkers)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	int * numpty	• pointer to number of active talkers
	MS_CDT * talkers	• pointer to array of talker descriptions
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Auxiliary	
Mode:	synchronous	

■ Description

The **dcb_gettalkers( )** function retrieves information about the conferees actively talking in the specified conference.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.
numpty:	Pointer to number of active talkers.
talkers:	Pointer to the array of talker descriptions.

NOTE: The list is not returned in any specified order.

The returned array of MS_CDT structures contains the active talker descriptions. The array has **numpty** number of elements. Each MS_CDT structure describes one active talker and has the following format:

```
typedef struct {
          int chan_num;      /* SCbus time slot number */
          int chan_sel;      /* meaning of time slot number */
          int chan_attr;     /* attribute description */
} MS_CDT;
```

chan_num contains the transmit time slot number of the actively talking conferee. chan_sel specifies that the conferee is an SCbus time slot.

NOTES: **1.** For active talker retrieval, chan_attr is not used.

2. Active talker information is associated with the DSP device handle. The information is invalid upon closing the device.

3. The application is responsible for allocating a table of MS_CDTs large enough to store all the active talkers.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- An invalid conference ID is specified.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define   NUM_PARTIES   2
#define   MAX_PTY       32

main()
{
   int  dspdevh;    /* DCB/SC DSP device handle */
   int  tsdevh1;    /* DTI time slot device handle */
   int  partycnt;   /* The no. of conferenced parties */
   int  confid;     /* Conference ID */
   SC_TSINFO     tsinfo;   /* Time slot information structure */
   MS_CDT  cdt[MAX_PTY];   /* Conference descriptor table */
   long  scts;     /* SCbus time slot */
   int   i;               /* Loop index */

   /* Open DCB/SC board 1, DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf("Cannot open dcbB1D1 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 device */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf("Cannot open dtiB1T1 : errno = %d", errno);
      exit(1);
```

```
}

/* Open DTI board 1, time slot 2 device */
if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
   printf("Cannot open dtiB1T2 : errno = %d", errno);
   exit(1);
}

/* Prepare time slot information structure */
tsinfo.sc_numts=1
tsinfo.sc_tsarrayp=&scts;

/* Get SCbus transmit time slot of tsdevh1 */
if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Set up CDT structure, for tsdevh1 */
cdt[0].chan_num = (int)scts;    /* SCbus time slot returned */
cdt[0].chan_sel = MSPN_TS;      /* ...by dt_getxmitslot() */
cdt[0].chan_attr = MSPA_NULL;   /* Conferee has no special attributes */

/* Get SCbus transmit time slot of tsdevh2 */
if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Set up CDT structure, for tsdevh2 */
cdt[1].chan_num = (int)scts;    /* SCbus time slot returned */
cdt[1].chan_sel = MSPN_TS;      /* by dt_getxmitslot() */
cdt[1].chan_attr = MSPA_PUPIL;  /* Conferee may be coached later */

/* Establish a 2 party conference */
if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1) {
   printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* Prepare time slot information structure */
tsinfo.sc_numts=1
tsinfo.sc_tsarrayp=cdt[0].chan_lts;

/* Listen to the time slot returned by dcb_estconf() */
if (dt_listen(tsdevh1, &tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Prepare time slot information structure */
tsinfo.sc_numts=1
tsinfo.sc_tsarrayp=cdt[1].chan_lts;

/* Listen to the time slot returned by dcb_estconf() */
if (dt_listen(tsdevh2, &tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Find out who is currently talking */
if ((dcb_gettalkers(dspdevh,confid,&partycnt,&cdt)) == -1) {
   printf ("Error Message = %s", ATDV_ERRMSGP(dspdevh));
```

```
      exit(1);
   }

   /* Print out the time slot numbers of currently active talkers */
   printf ("There are %d currently active talkers\n", partycnt);
   for (i=0; i<partycnt; i++){
      printf ("Time slot = %d , Chan_sel = 0x%x\n", cdt[i].chan_num, cdt[i].chan_sel);
   }

   /* Remove all time slot listens */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
              ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Done processing - close all open devices */
   if (dt_close(tsdevh1) == -1) {
      printf("Error closing %s : errno = %d\n", ATDV_NAMEP(tsdevh1), errno);
      exit(1);
   }

   /* Done processing - close device */
   if (dt_close(tsdevh2) == -1) {
      printf("Error closing %s : errno = %d\n", ATDV_NAMEP(tsdevh2), errno);
      exit(1);
   }

   /* Done processing - close device */
   if (dcb_close(dspdevh) == -1) {
      printf("Cannot close dcbB1D1 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

Name:	int dcb_monconf(devh,confid,lts)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	long *lts	• pointer to listen SCbus time slot
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_monconf( )** function adds a monitor to a conference. A monitor has no input in the conference.

This function places the monitored signal on the SCbus. Several parties can listen to the monitored signal simultaneously.

Parameter	Description
devh:	The DCB/SC board device handle.
confid:	The conference identifier.
lts:	Indicates the pointer to the returned listen SCbus time slot. The monitored signal is present on this time slot.

A monitor counts as one of the parties in the conference. If all the resources on the DSP are already in use, it is not possible to monitor the conference. When a conference is deleted, the conference monitor is also deleted.

NOTES:
1. There may only be one monitor in a conference.
2. Calling this function uses one conferencing resource.
3. It is the application's responsibility to listen to the time slot on which the monitored signal is transmitted.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- The conference is full.
- There are no more DSP conferencing resources available.
- The conference ID is invalid.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{
   int  dspdevh;              /* DCB/SC DSP device handle */
   int  tsdevh1, tsdevh2;     /* DTI time slot device handles */
   MS_CDT  cdt[NUM_PARTIES];  /* Conference descriptor table */
   int  confid;               /* Conference ID */
   long  lts, scts;           /* SCbus listen/transmit time slots */
   SC_TSINFO  tsinfo;         /* Time slot information structure */

   /* Open DCB/SC board 1, DSP 3 device */
   if ((dspdevh = dcb_open("dcbB1D3",0) == -1) {
      printf("Cannot open dcbB1D3 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, tslot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
          printf( "Cannot open dtiB1T1 : errno = %d", errno);
          exit(1);
   }

   /* Prepare the SCbus time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Get SCbus transmit time slot of DTI tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message = %s",ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts;   /* SCbus transmit time slot returned */
   cdt[0].chan_sel  = MSPN_TS;     /* ...from dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_NULL;   /* Conferee has no special attributes */
```

```
/* Open DTI board 1, tslot 2 */
if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
   printf("Cannot open dtiB1T2 : errno = %d", errno);
   exit(1);
}

/* Get transmit time slot of DTI tsdevh2 */
if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Set up CDT structure */
cdt[1].chan_num  = (int)scts ;              /* SCbus time slot returned */
cdt[1].chan_sel  = MSPN_TS;                 /* from dt_getxmitslot() */
cdt[1].chan_attr = MSPA_NULL;               /* Conferee has no special attributes */

/* Open DTI board 1, tslot 3 */
if ((tsdevh1 = dt_open("dtiB1T3",0)) == -1) {
     printf( "Cannot open dtiB1T3 : errno = %d", errno);
     exit(1);
}

/* Establish 2 party conference */
if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
   exit(1);
}

/* Do a listen for the tsdevh1 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

if (dt_listen(tsdevh1,&tsinfo) == -1){
   printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
   exit(1);
}

/* Do a listen for the tsdevh2 */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

if (dt_listen(tsdevh2,&tsinfo) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh2));
   exit(1);
}

/* Now monitor the conference on time slot lts */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarrayp = &lts;

if((dcb_monconf(dspdevh,confid,&lts)) == -1){
  printf("Error Message = %s",ATDV_ERRMSGP(dspdevh));
  exit(1);
}

/* Assume that a DTI time slot, tsdevh3, is a monitor */
if (dt_listen(tsdevh3,&tsinfo) == -1){
   printf("Error Message = %s",ATDV_ERRMSGP(tsdevh3));
   exit(1);
}
```

```
   /* Perform an unlisten() to end monitor listening */
   if (dt_unlisten(tsdevh3) == -1){
       printf("Error Message : %s", ATDV_ERRMSGP(tsdevh3));
       exit(1);
   }

   /* Now remove the monitor from the conference */
   if((dcb_unmonconf(dspdevh,confid)) == -1){
     printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
     exit(1);
   }

   /* 'Unlisten' the SCbus time slots */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }


   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d : Error Message = %s", confid,
      ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }

   if (dt_close(tsdevh3) == -1){
      printf("Error closing tsdevh3\n");
      exit(1);
   }

   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D3. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_unmonconf( )**

Name:	int dcb_open(name,rfu)	
Inputs:	char *name	• DCB/SC device name
	int rfu	• reserved parameter
Returns:	device handle on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
	errno.h	
Category:	Device Management	
Mode:	synchronous	

■ Description

The **dcb_open( )** function opens a device and returns a unique handle to identify the device. The device may be a DCB/SC board or a DSP on the board. All subsequent references to the opened device must be made using the device handle.

Parameter	Description
name:	Points to an ASCIIZ string that contains the name of a valid DCB/SC DSP device or board device.
rfu:	Reserved for future use. Set this parameter to 0.

Valid device names are listed below, where x specifies the board number.

Board	Valid Device Names	Description
DCB/320SC	dcbBxD1	DSP1 device handle
	dcbBx	Board device handle
DCB/640SC	dcbBx	Board device handle
	dcbBxD1	DSP1 device handle
	dcbBxD2	DSP2 device handle
DCB/960SC	dcbBx	Board device handle
	dcbBxD1	DSP1 device handle
	dcbBxD2	DSP2 device handle
	dcbBxD3	DSP3 device handle

NOTES: **1.** If a parent process opens a device and enables events, there is no guarantee that the child process will receive a particular event.

2. No action can be performed on a device until it is opened.

■ Cautions

This function fails when:

- The device name is invalid.
- The system has insufficient memory to complete the open.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

main()
{
   int bddevh;

   /* open board 1*/
   if ((bddevh = dcb_open("dcbB1", 0)) == -1) {
      printf("Cannot open device dcbB1. errno = %d\n", errno);
      exit(1);
   }
   else
      printf("Board %s is OPEN\n", ATDV_NAMEP(bddevh));

   /* Done processing - Close device */
   if (dcb_close(bddevh) == -1) {
      printf("Cannot close dcbB1 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

The **dcb_open( )** function does not return errors in the standard return code format. If an error occurred during the **dcb_open( )** call, a -1 will be returned, and the specific error number will be returned in the **errno** global variable. If a call to **dcb_open( )** is successful, the return value will be a handle for the opened device.

■ See Also

- **dcb_close( )**

Name:	int dcb_remfromconf(devh,confid,cdt)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference ID
	MS_CDT *cdt	• pointer to conference descriptor table element
Returns:	0 if success	
	-1 if failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_remfromconf( )** function removes a conferee from a conference. The conference ID is the value previously returned by the **dcb_estconf( )** function. In this case, the channel attributes of the MS_CDT structure are ignored. For a full description of the MS_CDT structure, see the **dcb_estconf( )** function.

NOTE: Calling this function frees one resource.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier number.
cdt:	Pointer to the conference descriptor table element.

NOTE: Dialogic recommends that the appropriate **xx_unlisten( )** function be called before removing the SCbus time slot member.

■ Cautions

1. An error will be returned if this function is used to attempt removal of the last remaining conferee from a conference. The **dcb_delconf( )** function must be used to end a conference.

2. This function also fails when:
    - The device handle passed is invalid.
    - The conference ID is invalid.
    - The conferee to be removed is not part of the specified conference.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define   NUM_PARTIES   3

main()
{
   int  dspdevh;                     /* Conference descriptor table */
   int  confid;                      /* Conference ID */
   int  tsdevh1, tsdevh2, tsdevh3;   /* DTI time slot device handles */
   long  scts;                       /* Transmit time slot */
   SC_TSINFO  tsinfo;                /* Time slot information structure */

   /* Open DCB/SC board 1, DSP 3 device */
   if ((dspdevh = dcb_open("dcbB1D3",0) == -1) {
      printf("Cannot open dcbB1D3 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
     printf( "Cannot open dtiB1T1 : errno = %d", errno);
     exit(1);
   }

   /* Prepare SCbus time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Get transmit time slot of DTI tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message = %s",ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts;   /* SCbus time slot returned */
   cdt[0].chan_sel  = MSPN_TS;     /* ...by dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_NULL;   /* Conferee has no special attributes */

   /* Open DTI board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf("Cannot open dtiB1T2 : errno = %d", errno);
      exit(1);
   }
   /* Get transmit time slot of DTI tsdevh2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message : %s",ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[1].chan_num  = (int)scts;     /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;       /* ...from dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_TARIFF;   /* Conferee receives periodic tariff tone */

   /* Open board 1, tslot 3 */
   if ((tsdevh3 = dt_open("dtiB1T3",0)) == -1) {
      printf( "Cannot open dtiB1T3: errno=%d", errno);
      exit(1);
   }

   /* Get transmit time slot of DTI tsdevh3 */
   if (dt_getxmitslot(tsdevh3, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh3));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[2].chan_num  = (int)scts;        /* SCbus time slot returned */
   cdt[2].chan_sel  = MSPN_TS;          /* ...from dt_getxmitslot() */
   cdt[2].chan_attr = MSPA_TARIFF;      /* Conferee receives periodic tariff tone */

   /* Establish 3 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for DTI tsdevh1 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh2 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh3 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[2].chan_lts;

   if (dt_listen(tsdevh3,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh3));
      exit(1);
   }

   /* Select tsdevh1 as conferee to remove from conference */
   /* Unlisten the listening device tsdevh1 */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Prepare SCbus time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Get transmit time slot of DTI tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Prepare the MS_CDT structure */
   cdt[0].chan_num = (int)scts;
   cdt[0].chan_sel = MSPN_TS;

   /* And remove tsdevh1 from the conference */
   if (dcb_remfromconf(dspdevh, confid, &cdt[0]) == -1) {
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Unlisten the remaining listening time slots */
   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   if (dt_unlisten(tsdevh3) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh3));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid)) == -1) {
      printf("Cannot delete conference %d : Error Message = %s", confid,
             ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }
   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }
   if (dt_close(tsdevh3) == -1){
      printf("Error closing tsdevh3\n");
      exit(1);
   }
   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D3 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_addtoconf( )**
- **dcb_delconf( )**
- **dcb_estconf( )**

Name:	int dcb_setbrdparm(devh,param,valuep)	
Inputs:	int devh	• DCB/SC board device handle
	unsigned char param	• device parameter name
	void * valuep	• pointer to parameter value
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Configuration	
Mode:	synchronous	

■ Description

The **dcb_setbrdparm( )** function sets DCB/SC board device parameters.

Parameter	Description
devh:	The valid board device handle returned by a call to **dcb_open( )**.
param:	The parameter whose value is to be altered.
valuep:	The address of the integer or DCB_VOL structure containing the values to be assigned to the parameter.

Parameters are specified in the **param** field and may have the IDs and descriptions as shown below:

Active Talker Feature (**MSG_ACTID**)	Active talker feature can be enabled or disabled by setting to ACTID_ON or ACTID_OFF.
Volume Control Digits (**MSG_VOLDIG**)	Defines the volume control status and volume digits. Possible values are UP, DOWN, and RESET digits.

The parameters are disabled by default and must be enabled using the **dcb_setbrdparm( )** function.

For MSG_ACTID, **valuep** points to an integer value. For MSG_VOLDIG, **valuep** points to an DCB_VOL structure defined as follows:

```
typedef  struct dcb_vol{
          unsigned char vol_control;   /* Feature status ON/OFF */
          unsigned char vol_up;        /* Digit to increase volume */
          unsigned char vol_reset;     /* Digit to reset volume to default level */
          unsigned char vol_down;      /* Digit to decrease volume */
}DCB_VOL;
```

vol_control indicates whether the volume control is turned ON or OFF. vol_up indicates the digit used for increasing the volume level. vol_down indicates the digit used for decreasing the volume, and vol_reset indicates the digit used to reset the volume level to the default.

■ Cautions

1. All parameter values must be integers or DCB_VOL structures, but since this routine expects a void pointer to **valuep**, the address must be cast as a void*.
2. This function fails when:
   - The device handle is invalid.
   - The parameter specified is invalid.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

main()
{

   int  bddevh;                  /* Board dev descriptor variables */
   int  valuep = ACTID_ON;
   DCB_VOL  volume;             /* Volume control  */

   /* open DCB board 1 */
   if ( (bddevh = dcb_open("dcbB1", 0)) == -1) {
      printf("Cannot open device dcbB1. errno = %d\n", errno);
      exit(1);
   }

   /* Enable Active talker feature */
   if (dcb_setbrdparm(devh, MSG_ACTID, &valuep) == -1) {
      printf("Error setting board param:0x%x\n ",ATDV_LASTERR(devh));
      exit(1);
   }

   volume.vol_control = ON;
   volume.vol_up   = 2;
   volume.vol_reset= 5;
   volume.vol_down     = 8;

   if (dcb_setbrdparm(devh, MSG_VOLDIG,(void *)&volume) == -1) {
      printf("Error getting board param:0x%x\n ", ATDV_LASTERR(devh));
      exit(1);
   }

   /*
    * Continue processing
    */

   /* Done processing - Close device */
   if ( dcb_close(bddevh) == -1) {
      printf("Cannot close device dcbB1. errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_getbrdparm( )**

Name:	int dcb_setcde(devh,confid,cdt)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	MS_CDT *cdt	• pointer to conference descriptor table element
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_setcde( )** function changes the attributes of a conferee in an existing conference.

Parameter	Description
devh:	The DCB/SC DSP device handle.
Confid:	The conference identifier number.
Cdt:	Pointer to the conference descriptor table element.

The conference descriptor table is an array of the MS_CDT structure. The MS_CDT structure has the following format:

```
typedef struct {
          int chan_num;      /* SCbus time slot number */
          int chan_sel;      /* time slot selector */
          int chan_attr;     /* attribute description */
} MS_CDT;
```

The chan_num denotes the time slot number of the device to be included in the conference. The chan_sel defines the meaning of the chan_num. At present, chan_sel must be set to the following value:

• MSPN_TS	SCbus time slot number

The chan_attr is a bitmask describing the conferee's properties within the conference. It can have one or more of the following values ORed together:

• MSPA_NULL	No special attributes.
• MSPA_RO	Conferee participates in conference in receive-only mode.
• MSPA_TARIFF	Conferee receives periodic tone for duration of call.
• MSPA_COACH	Conferee is a coach. Coach is heard by pupil only.
• MSPA_PUPIL	Conferee is a pupil. Pupil hears everyone including coach.

NOTE: If the conferee attributes of more than one conferee are to be set, this function must be called multiple times.

Table 6. Valid Attribute Combinations

Pupil	**Coach**	**Periodic Tone**	**Receive-only mode**
			X
		X	
		X	X
	X		
X			
X		X	

NOTES: **1.** Only one coach and one pupil are allowed in a conference at any time.

2. The default MSPA_NULL must be used if channel attributes are not set.

3. Invalid attribute combinations may lead to unexpected results.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- The conference ID is invalid.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define   NUM_PARTIES   2

main()
{
   int  dspdevh;               /* DCB/SC DSP device handle */
   MS_CDT  cdt[NUM_PARTIES];   /* Conference descriptor table */
   int  confid;                /* Conference ID */
   int  tsdevh1, tsdevh2;      /* DTI time slot device handle */
   long  scts;                 /* SCbus transmit time slot */
   SC_TSINFO  tsinfo;          /* Time slot information structure */

   /* Open DCB/SC board 1, DSP 3 device */
   if ((dspdevh = dcb_open("dcbB1D3",0)) == -1) {
      printf("Cannot open dcbB1D3 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1: errno=%d", errno);
      exit(1);
   }

   /* Prepare time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* get transmit time slot of DTI tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts;     /* SCbus time slot returned */
   cdt[0].chan_sel  = MSPN_TS;       /* by dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_TARIFF;   /* Conferee will receive period tariff tones */

   /* Open DTI board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2 : errno = %d", errno);
      exit(1);
   }

   /* Get transmit time slot of DTI tsdevh2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message = %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[1].chan_num  = (int)scts;    /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;      /* returned from getxmitslot */
   cdt[1].chan_attr = MSPA_PUPIL;   /* Conferee may be coached later */

   /* Establish a 2 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
      printf("Error Message = %s",ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for tsdevh1 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for tsdevh2 */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Now change the attribute of the last added conferee */
   /* NOTE : scts still contains the transmit time slot of tsdevh2 */
   cdt[0].chan_num = (int)scts;
   cdt[0].chan_sel = MSPN_TS;
   cdt[0].chan_attr = MSPA_TARIFF;

   if((dcb_setcde(dspdevh, confid, &cdt[0])) == -1) {
     printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
     exit(1);
   }

   /* Perform 'unlistens' on the listening DTI time slots */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
              ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }

   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D3 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h*, *msilib.h* or *dcblib.h*.

■ See Also

- **dcb_addtoconf( )**
- **dcb_estconf( )**
- **dcb_getcde( )**

Name:	int dcb_setdigitmsk(devh,confid,bitmask,action)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference identifier
	unsigned int bitmask	• event bitmask
	unsigned int action	• change type
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Configuration	
Mode:	synchronous	

■ Description

The **dcb_setdigitmsk( )** function enables specific digit detection for a conference. This current bitmask is examined by a call to **dcb_getdigitmsk( )**.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.
bitmask:	The digit bitmask.
action:	Specifies how the digit mask is changed. Possible values are: • CBA_SETMSK - enables notification of events specified in **bitmask** and disables notification of previously set events. • CBA_ADDMSK - enables messages from the conference specified in bitmask, in addition to previously set events. • CBA_SUBMSK - disables messages from the conference specified in bitmask.

NOTE: If MSG_VOLDIG is enabled to give transparent volume control to the conferees, the digits for volume increase, decrease, and reset will not cause digit events to be generated. As a result, the application will not know if the volume changes.

The **bitmask** determines the digits to be detected. Upon detection of a digit, a DCBEV_DIGIT event is generated on a DCB/SC DSP handle. The **sr_getevtdatap( )** function can be used to retrieve the following structure:

```
typedef struct {
  unsigned char dsp;                    /* DCB/SC DSP number */
  int  confid;                          /* Conference ID */
  int  chan_num;                        /* Channel/time slot number /
  int  chan_sel;                        /* Meaning of chan_num */
  int  chan_attr;                       /* Attribute of chan_num */
  unsigned char digits [DCB_MAXDIGS+1]; /* ASCIIZ string of detected digits */
  unsigned char dig_type;               /* Type of digits(s) detected */
} DCB_DIGITS;
```

The possible values for **bitmask** are:

CBMM_ZERO	Detect digit 0
CBMM_ONE	Detect digit 1
CBMM_TWO	Detect digit 2
CBMM_THREE	Detect digit 3
CBMM_FOUR	Detect digit 4
CBMM_FIVE	Detect digit 5
CBMM_SIX	Detect digit 6
CBMM_SEVEN	Detect digit 7
CBMM_EIGHT	Detect digit 8
CBMM_NINE	Detect digit 9
CBMM_STAR	Detect digit *
CBMM_POUND	Detect digit #
CBMM_A	Detect digit A
CBMM_B	Detect digit B
CBMM_C	Detect digit C
CBMM_D	Detect digit D
CBMM_ALL	Detect ALL digits

For example, to enable notification of the digits specified in the **bitmask** parameter and disable notification of previously set digits:

- specify the digits to enable in the **bitmask** field
- specify the CBA_SETMSK **bitmask** in the **action** field

To enable an additional digit specified in **bitmask** without disabling the currently enabled digits:

- specify the digits in **bitmask**
- specify CBA_ADDMSK in the **action** field

To disable digits in **bitmask** without disabling any other digits:

- specify the digits in **bitmask**
- specify CBA_SUBMSK in the **action** field

To disable all currently enabled digits:

- specify 0 in **bitmask**
- specify CBA_SETMSK in the **action** field

To enable an event handler for a specified event, follow these steps:

1. Call **sr_enbhdlr( )**. This function specifies the event and the application defined event handler that is called from a signal handler.
2. Call **dcb_setdigitmsk( )**. This function sets the digit message mask.

NOTE: The request for an event to be posted to an event handler must be specified using both the **sr_enbhdlr( )** and **dcb_setdigitmsk( )** functions.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- The action specified is invalid.
- Invalid conference ID.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"
```

```
#define   NUM_PARTIES   2

main()
{
   int  dspdevh;            /* DCB/SC DSP device handle */
   int  confid;             /* Conference Identifier */
   unsigned int bitmask;    /* Digit bitmask */
   int  tsdevh1, tsdevh2;   /* DTI time slot device handles */
   MS_CDT  cdt[NUM_PARTIES]; /* Conference descriptor table */
   long scts;               /* SCbus transmit time slot */

   /* Open DCB/SC board 1, DSP 2 device */
   if ((dspdevh = dcb_open("dcbB1D2",0)) == -1) {
      printf("Cannot open dcbB1D2. errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1 : errno = %d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Retrieve the SCbus transmit time slot for tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up the MS_CDT structure */
   cdt[0].chan_num  = (int)scts;     /* SCbus time slot returned */
   cdt[0].chan_sel  = MSPN_TS;       /* by dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_TARIFF;   /* Conferee receives periodic tariff tones */

   /* Open board 1, tslot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2 : errno = %d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Retrieve the SCbus transmit time slot for tsdevh2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Set up the MS_CDT structure */
   cdt[1].chan_num  = (int)scts;     /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;       /* by dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_TARIFF;   /* Conferee receives periodic tariff tones */

   /* Establish a 2 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == 1) {
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh1 device */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh2 device */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1){
      printf("Error Message = %s",ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /*
    * Enable DTMF detection for digits 1,3,5 only */
    */
   if (dcb_setdigitmsk(dspdevh, confid, CBMM_ONE|CBMM_THREE|CBMM_FIVE,
       CBA_SETMSK)) == -1) {
       printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
       exit(1);
   }

   /*
    *  Continue processing
    */

   /* Perform 'unlistens' on all DTI listening time slots */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d. Error Message = %s", confid,
              ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* And close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }

   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }
```

```
if(dcb_close(dspdevh) == -1) {
  printf("Cannot close dcbB1D2 : errno = %d", errno);
  exit(1);
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_getdigitmsk( )**

Name:	int dcb_unmonconf(devh,confid)	
Inputs:	int devh	• DCB/SC DSP device handle
	int confid	• conference ID
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
	dcblib.h	
Category:	Conference Management	
Mode:	synchronous	

■ Description

The **dcb_unmonconf( )** function removes a monitor from a conference.

Parameter	Description
devh:	The DCB/SC DSP device handle.
confid:	The conference identifier.

NOTES: **1.** Calling this function frees one resource.

2. Dialogic recommends that the appropriate **xx_unlisten( )** function be called for each conferee listening to the monitored signal before **dcb_unmonconf( )** is called.

■ Cautions

This function fails when:

- The device handle specified is invalid.
- It is called for a non-DCB/SC board.
- An invalid conference is specified.
- A monitor does not exist in the conference.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
#include "dcblib.h"
#include "errno.h"

#define NUM_PARTIES     2

main()
{
   int  dspdevh;                     /* DCB/SC DSP device handle */
   MS_CDT  cdt[NUM_PARTIES];         /* Conference descriptor table */
   int  confid;                      /* Conference ID */
   int  tsdevh1, tsdevh2, tsdevh3;   /* DTI time slot device handles */
   long  lts, scts;                  /* listen/transmit time slots */
   SC_TSINFO  tsinfo;                /* Time slot information structure */

   /* Open DCB/SC board 1, DSP 1 device */
   if ((dspdevh = dcb_open("dcbB1D1",0)) == -1) {
      printf("Cannot open dcbB1D1 : errno = %d", errno);
      exit(1);
   }

   /* Open DTI board 1, time slot 1 */
   if ((tsdevh1 = dt_open("dtiB1T1",0)) == -1) {
      printf( "Cannot open dtiB1T1 : errno = %d", errno);
      exit(1);
   }

   /* Prepare the time slot information structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &scts;

   /* Get transmit time slot of DTI tsdevh1 */
   if (dt_getxmitslot(tsdevh1, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[0].chan_num  = (int)scts;    /* SCbus time slot returned */
   cdt[0].chan_sel  = MSPN_TS;      /* by dt_getxmitslot() */
   cdt[0].chan_attr = MSPA_NULL;    /* Conferee has no special attributes */

   /* Open DTI board 1, time slot 2 */
   if ((tsdevh2 = dt_open("dtiB1T2",0)) == -1) {
      printf( "Cannot open dtiB1T2 : errno = %d", errno);
      exit(1);
   }

   /* Open board 1, time slot 3 */
   if ((tsdevh3 = dt_open("dtiB1T3",0)) == -1) {
      printf( "Cannot open dtiB1T3: errno=%d", errno);
      exit(1);
   }

   /* get transmit time slot of DTI TS device 2 */
   if (dt_getxmitslot(tsdevh2, &tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Set up CDT structure */
   cdt[1].chan_num  = (int)scts;    /* SCbus time slot returned */
   cdt[1].chan_sel  = MSPN_TS;      /* by dt_getxmitslot() */
   cdt[1].chan_attr = MSPA_NULL;    /* Conferee has no special attributes */

   /* Establish a 2 party conference */
   if (dcb_estconf(dspdevh, cdt, NUM_PARTIES, MSCA_ND, &confid) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh1 device */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[0].chan_lts;

   if (dt_listen(tsdevh1,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }

   /* Do a listen for the DTI tsdevh2 device */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &cdt[1].chan_lts;

   if (dt_listen(tsdevh2,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Now monitor the conference on SCbus time slot lts */
   if((dcb_monconf(dspdevh, confid, &lts)) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Prepare a time slot info structure */
   tsinfo.sc_numts = 1;
   tsinfo.sc_tsarrayp = &lts;

   /* And let a DTI time slot, tsdevh3, monitor the conference */
   if (dt_listen(tsdevh3,&tsinfo) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh4));
      exit(1);
   }

   /* Perform an 'unlisten' for the DTI time slot */
   if (dt_unlisten(tsdevh3) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh4));
      exit(1);
   }

   /* Now remove the monitoring */
   if((dcb_unmonconf(dspdevh,confid)) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* Perform 'unlistens' for the remaining DTI time slots */
   if (dt_unlisten(tsdevh1) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh1));
      exit(1);
   }
   if (dt_unlisten(tsdevh2) == -1){
      printf("Error Message : %s", ATDV_ERRMSGP(tsdevh2));
      exit(1);
   }

   /* Delete the conference */
   if(dcb_delconf(dspdevh, confid) == -1) {
      printf("Cannot delete conference %d : Error Message = %s", confid,
           ATDV_ERRMSGP(dspdevh));
      exit(1);
   }

   /* And close all open devices */
   if (dt_close(tsdevh1) == -1){
      printf("Error closing tsdevh1\n");
      exit(1);
   }
   if (dt_close(tsdevh2) == -1){
      printf("Error closing tsdevh2\n");
      exit(1);
   }
   if (dt_close(tsdevh3) == -1){
      printf("Error closing tsdevh3\n");
      exit(1);
   }
   if (dcb_close(dspdevh) == -1){
      printf("Cannot close dcbB1D1 : errno = %d\n", errno);
      exit(1);
   }
}
```

■ Errors

If the function does not complete successfully, it will return -1 to indicate an error. Use the Standard Attribute functions **ATDV_LASTERR( )** to obtain the applicable error value, or **ATDV_ERRMSGP( )** to obtain a more descriptive error message.

Refer to the error type tables found in *Chapter 2. DCB/SC Library Function Overview* of this guide. Error defines can be found in *dtilib.h, msilib.h* or *dcblib.h*.

■ See Also

- **dcb_estconf( )**
- **dcb_monconf( )**

4. Application Guidelines

This chapter contains suggestions to guide programmers in designing and coding a Dialogic DCB/SC application for Windows NT.

4.1. Writing a Simple Application

This section provides DCB/SC general and task-specific programming guidelines:

- General Guidelines
- Initialization
- Compiling and Linking
- Aborting

NOTE: These guidelines are not a comprehensive guide to developing or debugging DCB/SC applications. The examples following each function description in *Chapter 3. Function Reference* illustrate proper use of the DCB/SC functions.

4.1.1. General Guidelines

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

- Using symbolic defines
- Including header files
- Checking return codes

Using Symbolic Defines

Dialogic does not guarantee 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. Symbolic defines are found in the *dtilib.h*, *msilib.h,* and *dcblib.h* files.

Including Header Files

Various header files must be included in your application to test for error conditions, to use library functions from other Dialogic products, or to perform event-management and standard-attribute functions. An example is shown below. See *Chapter 2. DCB/SC Library Function Overview* for details.

```
#include <windows.h>
#include <errno.h>
#include “srllib.h”
#include “dtilib.h”
#include “msilib.h”
#include “dcblib.h”
```

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

Checking Return Codes

Most Network and DCB/SC 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 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 DCB/SC library function */

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

Using this technique ensures that all errors resulting from a 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 non-zero value is returned when the function fails. For details, see *Chapter 2. DCB/SC Library Function Overview* and *Chapter 3. Function Reference*.

```
/* error handling routine */
void do_error( devh, funcname )
   int devh;                 /* Dialogic device handle */
   char *funcname;           /* function name */
{
   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 );
      ( invalid call remove)
   } else {
   printf( "Error value = %d\nError message = %s\n",
            errorval, ATDV_ERRMSGP( devh ) );
   }
   return;
}

main( )
{
   .
   .
   .
   /* example call to Dialogic DCB/SC library function */
   if (dcb_setdigitmsk( devh, confid, CBMM_ALL)!= 0) {
       do_error( devh, "dcb_setdigitmsk()" );
   }
   /* successful function call - continue processing */
   .
   .
   .
}
```

NOTE: Calls to **dcb_open( )** return either -1 or a non-zero device handle. Therefore, when issuing the **dcb_open( )** function, check for a return of -1. The specific error can be found in the global variable **errno**, contained in *errno.h*.

4.1.2. Initialization

As a first step, a DCB/SC application must intialize parameters.

Set Configuration

Use **dcb_setbrdparm( )** to set active talker and volume control digits. Specific setting choices include:

Active Talker Feature (**MSG_ACTID**)	Active Talker feature can be enabled or disabled by setting to ACTID_ON or ACTID_OFF.
Volume Control Digits (**MSG_VOLDIG**)	Defines the volume control status and volume up/down/reset digits.

If the resource assignment table updates are desired by the application, enable event generation using **dcb_evtstatus( )**.

4.1.3. Terminating

When your process completes, devices should be shut down in an orderly fashion. Tasks that are performed to terminate an application generally include:

- Disabling events
- Stop listening to time slots
- Deleting all conferences
- Closing devices

NOTE: SRL Event Management functions (such as **sr_dishdlr( )**, which disables an event handler) must be called before closing the device that is sending the handler event notifications (see *Appendix A* for SRL details).

4.1.4. Compiling and Linking

To compile and link your application, follow the instructions for your version of the Windows NT Compiler.

If using Microsoft Visual C++, link with the following libraries when using the DCB/SC board:

```
libsrlmt.lib
libdtimt.lib
```

By default, the files are found in <install drive%>:<install directory>\dialogic\lib. Depending on your application, you may need to link with other libraries. Refer to the appropriate documentation for other Dialogic products.

If using Borland C++, you must include the cross-compatibility library files in your file set. The files are found in <install drive%>:<install directory>\dialogic\lib. You may also use the x-compatibility files when using Microsoft Visual C++.

4.1.5. Aborting

If you abort a DCB/SC Windows NT application by pressing the interrupt key, the Windows NT system will terminate the current process but may leave devices in an unknown state. As a result, you may encounter errors the next time the application runs.

To avoid errors of this type, your application should include an event handler that traps the interrupt key and performs the actions listed in *Section 4.1.3. Terminating*.

Appendix A

Standard Runtime Library: DCB/SC Entries and Returns

The Standard Runtime Library is a device-independent library containing Event Management functions, Standard Attribute functions, and the DV_TPT Termination Parameter table. SRL functions and data structures are described in detail in the *Standard Runtime Library Programmer's Guide* and the *Voice Software Reference for Windows NT*.

This appendix lists the DCB/SC entries and returns for each of the Standard Runtime Library (SRL) components.

Table 7. Guide to Appendix A

SRL Component	DCB/SC Data Listed in Appendix A
Event Management functions	DCB/SC inputs for Event Management functions .
	DCB/SC returns from Event Management functions.
Standard Attribute functions	DCB/SC values returned by the Standard Attribute functions.
DV_TPT table	Termination conditions and related data.
	Not used by the DCB/SC device.

NOTE: The header file for this library is *srllib.h*. It must be included in the application code prior to including *dtilib.h*, *dcblib.h*, and *msilib.h*. For example:

```
#include “srllib.h”
#include “dtilib.h”
#include “msilib.h”
#include “dcblib.h”
```

Event Management Functions

The Event Management functions retrieve and handle DCB/SC events. The functions are listed in the following tables.

Table 8. DCB/SC Inputs for Event Management Functions

Event Management Function	DCB/SC specific Input	Value
sr_enbhdlr( ) Enable event handler	evt_type	DCBEV_CTU DCBEV_DIGIT
sr_dishdlr( ) Disable event handler	evt_type	Same as above.
sr_waitevt( ) Wait for next event		N/A
sr_waitevtEx( ) Wait for next event		N/A

Table 9. DCB/SC Returns from Event Management Functions

Event Management Function	DCB/SC specific Input	Value
sr_getevtdev( ) Get Dialogic device handle	device	DCB/SC device handle.
sr_getevttype( ) Get event type	event type	DCBEV_DIGIT DCBEV_CTU
sr_getevtlen( ) Get event length	event length	Number of bytes in the data returned.
sr_getevtdatap( ) Get pointer to event data	event data	Pointer to DCB_DIGITS structure for DCBEV_DIGIT. Pointer to DCB_CT structure, the updated resource table for DCBEV_CTU.

Standard Attribute Functions

Standard Attribute functions return general device information such as the device name, or the last error that occurred on the device.

Table 10. Standard Attribute Functions

Standard Attribute Function	Information Returned for DCB/SC
ATDV_ERRMSGP( )	Pointer to string describing the error that occurred during the last function call on a device.
ATDV_LASTERR( )	The error that occurred during the last function call on a specified device.
ATDV_NAMEP( )	Pointer to device name (dcbBb).

Appendix B
Related Publications

Below is a list of Dialogic publications to read for information on products related to the DCB/SC.

Dialogic References

- *Digital Network Interface Software Reference for Windows NT*
- *DCB/SC Quick Install Card*
- *Voice Software Reference for Windows NT*
- *MSI/SC Software Reference for Windows NT*
- *Dialogic Product and Services Guide 1997*

Glossary

analog: In this guide, analog refers to agent communications between a headset or a telephone and the MSI/SC or to the *loop-start* type of network interface.

asynchronous function: Allows program execution to continue without waiting for a task to complete. See *synchronous function*.

baseboard: A term used in voice processing to mean a printed circuit board without any daughterboards attached.

channel: 1. When used in reference to a Dialogic digital expansion board, a data path, or the activity happening on that data path. 2. When used in reference to the CEPT telephony standard, one of 32 digital data streams (30 voice, 1 framing, 1 signaling) carried on the 2.048 Mhz/sec E-1 frame. (See *time slot*.) 3. When used in reference to a bus, an electrical circuit carrying control information and data.

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.

daughterboard: In the context of this guide, the DCB/SC daughterboard assembly. The daughterboard enables the DCB/SC hardware to interface to analog station devices.

DCB/SC: The Dialogic single slot, DSP-based conferencing solution. Each product in the DCB/SC series contains one, two, or three DSP(s). Each Digital Signal Processor (DSP) supports up to 32 conferees, for a maximum of 96 (3 sets of 32) conferencing resources.

device: Any computer peripheral or component that is controlled through a software device driver.

digital: Information represented as binary code.

DIP switch: A switch usually attached to a printed circuit board with two settings- on or off. DIP switches are used to configure the board in a semipermanent way.

driver: A software module that provides a defined interface between a program and the hardware.

DTMF: Dual Tone Multi-Frequency. DTMF refers to the combination of two tones which represents a number on a telephone key pad. Each pushbutton has its own unique combination of tones.

E-1: Another name given to the CEPT digital telephony format devised by the CCITT that carries data at the rate of 2.048 Mbps (DS-1level). This service is available in Europe and some parts of Asia.

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

Extended Attribute functions: Class of functions that take one input parameter (a valid Dialogic device handle) and return device-specific information.

host PC: The system PC in which Dialogic hardware and software are installed and applications are run and/or developed.

loop start interfaces: Devices, such as an analog telephones, that receive an analog electric current. For example, taking the receiver off hook closes the current loop and initiates the calling process.

MSI: Modular Station Interface. A PEB-based Dialogic expansion board that interfaces PEB time slots to analog station devices by way of modular daughterboards.

MSI/SC: Modular Station Interface. An SCbus-based Dialogic expansion board that interfaces SCbus time slots to analog station devices.

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

rfu: Reserved for future use.

SCbus: Signal Computing bus. A hardwired connection between Switch Handlers (SC2000 chips) on SCbus-based products for transmitting information over 1024 time slots to all devices connected to the SCbus.

SCSA: Signal Computing System Architecture. A generalized open-standard architecture describing the components and specifying the interfaces for a signal processing system for the PC-based voice processing, call processing and telecom switching industry.

Signal Computing System Architecture: See *SCSA*.

Standard Attribute functions: Class of functions that take one input parameter (a valid Dialogic device handle) and return generic information about the device. The Dialogic SRL contains Standard Attribute functions for all Dialogic devices. Standard Attribute function names are case-sensitive and must be in capital letters. See *Extended Attribute functions*.

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

Index

A

Aborting an application, 106

Active talkers, 3

Application guidelines, 103
  aborting, 106
  compliling and linking, 106
  general guidelines, 103
  initialization, 105
  terminating, 106
  writing an application, 103

Attribute combinations, 22, 36, 89

Auxiliary functions, 13
  dcb_evtstatus( ), 13, 41
  dcb_getatibits( ), 13, 46
  dcb_gettalkers( ), 13, 68

B

Baseboard
  D/41ESC, 10

C

Channel attribute returns, 54

Compatibility, 4

Compiling and linking, 106

Conference descriptor table, 36, 53, 59, 88

Conference management functions, 14
  dcb_addtoconf( ), 14, 20
  dcb_delconf( ), 14, 28
  dcb_estconf( ), 14, 35
  dcb_getcde( ), 14, 53
  dcb_getcnflist( ), 14, 58
  dcb_monconf( ), 14, 72
  dcb_remfromconf( ), 14, 80
  dcb_setcde( ), 14, 88
  dcb_unmonconf( ), 14, 99

Configuration functions, 14
  dcb_dsprescount( ), 14, 32
  dcb_getbrdparm( ), 14, 50
  dcb_getdigitmsk( ), 14, 63
  dcb_setbrdparm( ), 14, 85
  dcb_setdigitmsk( ), 14, 93

D

D/160SC-LS, 9

D/240SC, 9

D/240SC-T1, 9

D/300SC, 9

D/300SC-E1, 9

D/41ESC, 9
  baseboard, 10

Daughterboard
  FAX/40E, 10

DCB introduction, 1

DCB/SC
  DCB/320SC, 1, 113
  DCB/640SC, 1, 113
  DCB/960SC, 1, 113

dcb_addtoconf( ), 20

DCB_CDT structure, 20, 36, 53, 80, 88

dcb_close( ), 26

dcb_delconf( ), 28

dcb_dsprescount( ), 32

dcb_estconf( ), 35

dcb_evtstatus( ), 41

dcb_getatibits( ), 46

dcb_getbrdparm( ), 50

dcb_getcde( ), 53

dcb_getcnflist( ), 58

dcb_getdigitmsk( ), 63

dcb_gettalkers( ), 68

dcb_monconf( ), 72

dcb_open( ), 77

dcb_remfromconf( ), 80

dcb_setbrdparm( ), 85

dcb_setcde( ), 88

dcb_setdigitmsk( ), 93

dcb_unmonconf( ), 99

Defines, 103

Device Management functions, 15
- dcb_close( ), 15, 26
- dcb_open( ), 15, 77

DIALOG/HD, 10

Dialogic Audio Conferencing, 1
- compatibility, 4
- features, 1
- functional description, 4
- software features, 2

Documentation
- conventions, 19
- related publications, 113

DV_TPT Termination Parameter table, 109

E

Error handling, 15
- errors defined in dtilib.h, 15
- errors defined in msilib.h, 18

Event Management functions, 109, 110

F

FAX/40E
- daughterboard, 10

Function reference, 19
- documentation conventions, 19

Functional Description, 4

Functions
- auxiliary, 13
- categories, 13
- conference management, 14
- configuration, 14
- device management, 15

G

General guidelines, 103
- header files, 104
- return codes, 104
- symbolic defines, 103

H

Header files, 104

How to use this guide, 10

I

Include files, 18, 104

Initialization of an application, 105

L

Library functions, 13
- auxiliary functions, 13
- categories, 13
- conference management functions, 13, 14
- configuration functions, 13, 14
- device management functions, 13, 15
- error handling, 15

function reference, 19
include files, 18
overview, 13

M

MSI/SC, 10
MSI/SC-R, 10
ringing, 10

N

Numpty, 37, 68

O

Organization of this guide, 1

P

Product terminology, 9

Products covered by this guide, 9

R

Related publications, 113

Resource allocation, 4
functions and results, 4

Return codes, 104

S

SCbus, 10

Software features, 2
active talkers, 3
resource allocation, 4
volume control, 3

SpanCard, 10

Standard Attribute functions, 109, 111

Standard Runtime Library, 109

Symbolic defines, 103

T

Terminating an application, 106

V

Valid Attribute Combinations, 22, 36, 89

VFX/40ESC, 10

Volume control, 3

W

Writing an application, 103

Dialogic Sales Offices

North American Sales
1-800-755-4444 or 201-993-3030
fax: 201-631-9631

Corporate Headquarters
1515 Route 10
Parsippany, NJ 07054-4596
USA
201-993-3000
fax: 201-993-3093

Northeastern US
800 South Street Suite 100
Waltham, MA 02154

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

North Central US
1901 North Roselle Road
Suite 800
Schaumburg, IL 60195

South Central US
3307 Northland Drive
Suite 300
Austin, TX 78731

Western US
1314 Chesapeake Terrace
Sunnyvale, CA 94089

Northwestern US
19125 North Creek Parkway
#120
Bothell, WA 98011

GammaLink Division
3120 Scott Blvd. Suite 110
Santa Clara, CA 95054

Computer-Telephone Division
100 Unicorn Park Drive
Woburn, MA 01801

Spectron Microsystems Division
315 Bollay Drive
Santa Barbara, CA 93117
805-968-5100fax: 805-968-9770

Dialogic On-Line Information Retrieval System (fax-on-demand)
1-800-755-5599 or 201-993-1063

GammaLink Fax-on-Demand
408-734-9906

computer telephony BBS
(ctBBS)
201-993-0864

CTI@Dialogic WWW Site
http://www.dialogic.com

Dialogic Sales Internet
sales@dialogic.com

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

Latin America & Caribbean
Dialogic Latin America and Caribbean
Av. R. S. Peña
730   3º Piso
Oficina 34
(1035) Buenos Aires, Argentina
541-328-1531 or -9943
fax: 541-328-5425

European Headquarters (serving Europe, Middle East, & Africa)

Dialogic Telecom Europe N.V.-S.A.
Airway Park
Lozenberg 23
Building D, 3rd floor
B-1932 Sint Stevens Woluwe
Belgium
32-2-712-4311
fax : 32-2-712-4300

DTE On-Line Information Retrieval System
32-2-712-4322

DTE BBS
32-2-725-7846

Germany, Switzerland, & Austria
Dialogic Telecom Deutschland GmbH
Industriestrasse 1
D82110 Munich
Germany
49-89-894-362-0
fax: 49-89-894-362-77

France
Dialogic Telecom France S.r.l.
42, Avenue Montaigne
F-75008
Paris
France
33-1-53-67-52-80
fax: 33-1-53-67-52-79

Italy
Dialogic Telecom Italy S.r.l.
Strada Pavese, 1/3
I-20089 Rozzano
Milan
Italy
39-2-575-54302
fax: 39-2-575-54310

United Kingdom, Ireland, & 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

People's Republic of China, Hong Kong, Macau, and Taiwan
Dialogic Beijing Representative Office
8 North Dongsanhuan Road
Landmark Building, Suite 1308
Chaoyang District
Beijing 100004
People's Republic of China
86-10-504-5364
fax: 86-10-506-7989

Japan & 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, & Australia
Dialogic SEA Pte. Ltd.
9 Temasek Blvd.
#37-02 Suntec Tower Two
Singapore 038989
65-339-9833
fax: 65-339-9211

DSEA BBS

65-291-3249

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

NOTES

NOTES

NOTES