SCbus Routing Software Reference

for Windows

05-0439-004

COPYRIGHT NOTICE

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

Publication Date: September, 2001

Part Number: 05-0439-004

Dialogic, an Intel Company
1515 Route 10
Parsippany NJ 07054
U.S.A.

For **Technical Support**, visit the Dialogic support website at:
http://support.dialogic.com

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

Table of Contents

1. Introduction to SCbus Functions

SCbus Overview

This guide describes the SCbus routing functions. These functions provide the ability to establish communications between Dialogic devices. An explanation of an SCbus Windows demo program is also provided in this guide. For information on SCbus routing, refer to the *SCbus Routing Guide*.

NOTE: The SCbus routing functions only support SCbus hardware configurations.

The SCbus is a real-time, high-speed, time division multiplexed (TDM) communications bus that provides 1024 time slots for transmission of digital information between SCbus products. The SCbus allows high-density systems to efficiently share resources so that multiple technologies can be connected to each port as needed.

Each SCbus product consists of several devices. Each of these devices can communicate via the SCbus with any other device connected to the SCbus. For example:

- a D/41ESC board provides 4 on-board analog loop start interface devices and 4 voice devices, for a total of 8 devices communicating over the SCbus.
- a D/160SC-LS board provides 16 on-board analog loop start interface devices and 16 voice devices, for a total of 32 devices communicating over the SCbus.
- a D/240SC-T1 board provides a T-1/DSX-1 interface for processing 24 T-1 time slots/calls and 24 voice devices, for a total of 48 devices communicating over the SCbus.
- a D/300SC-E1 board provides an E-1 interface for processing 30 E-1 time slots/calls and 30 voice devices, for a total of 60 devices communicating over the SCbus.

- a VFX/40ESC board provides 4 on-board analog loop start interface devices and 4 FAX/voice devices, for a total of 8 devices communicating over the SCbus.
- a MSI/240SC board provides 24 on board station devices, for a total of 24 devices communicating over the SCbus.

All devices connected to the SCbus have a transmit (TX) channel and a receive (RX)(listen) channel. At system initialization, each transmit channel is assigned to a specific and unique SCbus time slot. This transmit channel assignment cannot be changed by the application.

Since all transmit channels are pre-assigned, routing an SCbus device only requires connecting the receive (listen) channel of the device to an SCbus time slot. The connected device then listens to all data transmitted over that SCbus time slot. This receive channel can be moved (disconnected and connected) to a different SCbus time slot at any time by the application.

NOTE: The SCbus routing functions only support SCbus configurations.

1.1. SCbus Routing Function Overview

The SCbus routing functions comprise network /resource (nr_) SCbus convenience functions and individual SCbus routing functions. For most routing applications, the following network/resource SCbus convenience functions should suffice. The SCbus convenience functions include all of the functionality of the individual SCbus routing functions.

nr_scroute( )	• Makes half or full duplex connection between two Dialogic devices.
nr_scunroute( )	• Breaks half or full duplex connection between two Dialogic devices.

The network/resource SCbus convenience functions are not part of the voice library. Their C source code is provided in a separate file called *sctools.c* in *<install drive>:\<install directory>\dialogic\sctools* directory.

The individual SCbus routing functions provide the ability to program each phase of connecting or disconnecting the receive channel of a device to the transmit

channel of another device or to build your own convenience functions. These individual SCbus routing functions can be characterized by their prefix and by their suffix.

The prefix of the individual SCbus routing functions identify the type of device to which the function applies:

ag_	• analog device (loop-start interface)
dt_	• digital device(T-1 or E-1 digital service interface)
dx_	• voice device
fx_	• FAX device
ms_	• MSI device

The suffix of the individual SCbus routing functions identify the operation or task performed by the function:

_getctinfo( )	• Returns information about a device (analog device, voice device, digital device, or other technology device). This function is not used for routing.
_getxmitslot( )	• Returns the SCbus time slot information into an SC_TSINFO structure that includes the number of the SCbus time slot connected to the transmit channel of the specified device.
_listen( )	• Connects the listen (receive) channel of the specified device to an SCbus time slot.
_unlisten( )	• Disconnects the listen (receive) channel of the specified device from an SCbus time slot.

The individual SCbus functions described in this reference are:

ag_getctinfo( )	• returns information about analog device
ag_getctinfo( )	• returns information about analog device
ag_getxmitslot( )	• returns SCbus time slot information into an SC_TSINFO structure that includes the number of the SCbus time slot connected to the analog

	transmit channel
ag_listen( )	• connects analog receive (listen) channel to an SCbus time slot
ag_unlisten( )	• disconnects analog receive (listen) channel from SCbus time slot
dt_getctinfo( )	• returns information about digital interface device
dt_getxmitslot( )	• returns SCbus time slot information into an SC_TSINFO structure that includes the number of the SCbus time slot connected to the digital transmit channel
dt_listen( )	• connects digital receive (listen) channel to an SCbus time slot
dt_unlisten( )	• disconnects digital receive (listen) channel from SCbus time slot
dx_getctinfo( )	• returns information about voice device
dx_getxmitslot( )	• returns SCbus time slot information into an SC_TSINFO structure that includes the number of the SCbus time slot connected to the voice transmit channel
dx_listen( )	• connects receive voice (listen) channel to an SCbus time slot
dx_unlisten( )	• disconnects voice receive (listen) channel from SCbus time slot
fx_getxmitslot( )	• returns SCbus time slot information into an SC_TSINFO structure that includes the number of the SCbus time slot connected to the FAX transmit channel
fx_listen( )	• connects FAX receive (listen) channel to an SCbus time slot
fx_unlisten( )	• disconnects FAX receive (listen) channel from SCbus time slot
ms_getxmitslot( )	• returns SCbus time slot information into an SC TSINFO structure that

	includes the number of the SCbus time slot connected to the MSI station
ms_listen( )	• connects MSI receive station to an SCbus time slot
ms_unlisten( )	• disconnects a MSI station from SCbus time slot

Product Terminology

The following product naming conventions are used throughout this guide:

Antares refers to the series of Dialogic 32-channel standalone open DSP (Digital Signal Processor) platforms, which can serve as call processing devices in telecomputing servers created under the industry-standard SCbus.

Antares 2000/50 refers to the Dialogic 32-channel standalone open DSP (Digital Signal Processor) platform that provides 512 KB of SRAM and 4 MB of Global DRAM.

Antares 3000/50 refers to the Dialogic 32-channel standalone open DSP (Digital Signal Processor) platform that provides 512 KB of SRAM and 8 MB of Global DRAM.

Antares 6000/50 refers to the Dialogic 32-channel standalone open DSP (Digital Signal Processor) platform that provides 2 MB of SRAM and 8 MB of Global DRAM.

CPI/400-PCI refers to the Gammalink four-channel analog fax board from Dialogic.

D/21D refers to the Dialogic 2-channel voice board with on-board analog loop start interface.

D/21H refers to the next-generation Dialogic 2-channel voice board occupying a half-sized PC ISA slot with on-board analog loop start interface.

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

D/41H refers to the next-generation Dialogic 4-channel voice board occupying a half-sized PC ISA slot with on-board analog loop start interface.

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

D/42-xx refers to the Dialogic 4-channel voice board with PBX integration.

D/80SC refers to the Dialogic 8-channel voice board for use with a network interface board.

D/160SC refers to the Dialogic 16-channel voice board for use with a network interface board.

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/240SC-2T1 refers to the Dialogic 24-channel voice board with dual on-board T-1 digital interfaces.

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

D/300SC-2E1-75 refers to the Dialogic 30-channel voice board with dual on-board E-1 digital interfaces.

D/320SC refers to the Dialogic 32-channel voice board for use with a network interface board.

D/480SC-2T1 refers to the Dialogic 48-channel voice board with two on-board T-1 digital interfaces.

D/600SC-2E1-75 refers to the Dialogic 60-channel voice board with two on-board E-1 digital interfaces.

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

DCB/320SC refers to the Dialogic Conferencing board which uses DSP technology to provide 32 conferencing resources in a single PC slot. Each conferencing resource can be dynamically assigned to conferences of any size from a minimum of 1 to a maximum of 32 conferees.

DCB/640SC refers to the Dialogic Conferencing board which uses DSP technology to provide 64 conferencing resources in a single PC slot. Each conferencing resource can be dynamically assigned to conferences of any size from a minimum of 1 to a maximum of 32 conferees.

DCB/960SC refers to the Dialogic Conferencing board which uses DSP technology to provide 96 conferencing resources in a single PC slot. Each conferencing resource can be dynamically assigned to conferences of any size from a minimum of 1 to a maximum of 32 conferees.

Dialog/4 refers to the Dialogic 4-channel voice board with an on-board analog loop start interface.

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

DTI/240SC refers to the Dialogic 24-channel voice board with on-board T-1 digital interface and ISDN network interface.

DTI/241SC refers to the Dialogic 24-channel voice board with on-board T-1 digital interface and ISDN network interface. This board includes optional tone signaling and outbound call progress analysis.

DTI/300SC refers to the Dialogic 30-channel voice board with on-board E-1 digital interface and ISDN network interface.

DTI/301SC refers to the Dialogic 30-channel voice board with on-board E-1 digital interface and ISDN network interface. This board includes optional tone signaling and outbound call progress analysis.

DTI/480SC refers to the Dialogic 48-channel voice board with dual T-1 digital interface and ISDN primary rate network interface.

DTI/600SC refers to the Dialogic 60-channel voice board with dual E-1 digital interface and ISDN primary rate network interface.

LSI/81SC refers to the Dialogic 8-channel voice board with on-board analog loop start interface.

LSI/161SC refers to the Dialogic 16-channel voice board with on-board analog loop start interface.

MSI/SC refers to the Dialogic modular station interface that connects up to 24 analog station devices to digital time slots. This board series supports conferencing applications.

MSI/80SC refers to the Dialogic modular station interface board with 8 analog station interfaces.

MSI/80SC-R refers to the Dialogic MSI/80SC board with built-in ringing capability.

MSI/160SC refers to the Dialogic modular station interface board with 16 analog station interfaces.

MSI/160SC-R refers to the Dialogic MSI/160SC board with built-in ringing capability.

MSI/240SC refers to the Dialogic modular station interface board with 24 analog station interfaces.

MSI/240SC-R refers to the Dialogic MSI/160SC board with built-in ringing capability.

Proline/2V refers to the feature rich Dialogic 2-channel voice board occupying a 2/3-size PC ISA slot with on-board analog loop start interface.

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

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

Stingray is an open-architecture platform for computer telephony that integrates the switching and voice processing resources under a single hardware and software architecture. The Stingray is a two-board assembly consisting of a D/80SC-4LS board and an MSI/SC board connected by the SCbus (SCbus connectivity provides the switching between trunks and stations and also allows expansion for additional boards).

VFX/40E refers to the Dialogic voice and fax resource board offering 4-ports of enhanced analog loop start call processing in a single slot.

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.

VFX/40ESC+ refers to the Dialogic voice/fax board with four-channel design incorporating an SCbus or PEB interface. The VFX/40ESC Plus is a next generation VFX/40ESC with additional DSP RAM that provides expanded voice processing capability. The VFX/40ESC Plus provides the voice features of the D/41E and includes support for Caller ID, 11KHz linear .WAV files, and dial pulse detection.

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

2. SCbus Convenience Functions

SCbus Convenience Function Overview

The Dialogic *sctools.h* library provides SCbus convenience functions that make the routing process easier for application developers by allowing them to perform a combination of SCbus individual functions using only two function calls. Using the **nr_scroute( )** and **nr_scunroute( )** functions, the application specifies only two device handles and device types, and whether the connection will be full or half-duplex. The SCbus convenience functions are described in this chapter.

SCbus Convenience Function Descriptions

Name:	int nr_scroute(devh1,devtype1,devh2,devtype2,mode)	
Inputs:	int devh1	• device handle
	unsigned short devtype1	• specifies the type of device for devh1
	int devh2	• device handle
	unsigned short devtype2	• specifies the type of device for devh2
	unsigned char mode	• specifies half or full duplex connection
Returns:	0 on success -1 on error	
Includes:	stdio.h, windows.h, srllib.h, dxxxlib.h and sctools.h. Optional - dtilib.h, msilib.h, and faxlib.h.	
Category:	Routing convenience	
Mode:	Synchronous	

■ Description

The **nr_scroute( )** convenience function connects two Dialogic devices. This function makes a full or half-duplex connection between two Dialogic devices. This function is provided in a separate C source file called *sctools.c* in the *<install drive>:\<install directory>\dialogic\sctools* directory. A library version is also available as *sctools.lib* in the *<install drive>:\<install directory>\dialogic\lib* directory.

The nr_sc prefix to the function signifies network (analog and digital) devices and resource (voice and fax) devices accessible via the SCbus. See the *Digital Network Interface Software Reference* for digital interface device details and the *FAX Software Reference* for FAX device details.

NOTE: DTI, MSI, or FAX functionality may be conditionally compiled in or out of the function using the DTISC and/or FAXSC defines in the makefile provided with the function. If DTI or FAX functionality is being compiled in, then you must link with the DTI or FAX library. See the *Digital Network Interface Software Reference* for digital interface device details and the *FAX Software Reference* for FAX device details.

Parameter	Description
devh1:	Dialogic device handle.

Parameter	Description	
devtype1:	Specifies the type of device for devh1.	
	SC_VOX	voice channel device
	SC_LSI	analog channel device
	SC_DTI	digital time slot device
	SC_MSI	MSI station device
	SC_FAX	fax channel device
devh2:	Dialogic device handle.	
devtype2:	Specifies the type of device for **devh2**. See **devtype1** for list of defines.	
mode:	Specifies half or full duplex connection. The mode parameter contains one of the following defines from sctools.h to specify full or half duplex:	
	SC_FULLDUP	specifies make a full duplex connection
	SC_HALFDUP	specifies make a half duplex connection.
	The default mode value is SC_FULLDUP. When SC_HALFDUP is specified, then the function returns with the second device listening to the SCbus time slot connected to the first device.	

■ Cautions

1. The devtype1 and devtype2 parameters must match the types of the device handles in devh1 and devh2.
2. If you have not defined DTISC and FAXSC when compiling the *sctools.c* file, you cannot use this function to route digital channels, MSI stations, or fax channels.

■ Source Code

```
/*
 * Include files.
 */
#include <windows.h>
#include <stdio.h>
#include <srllib.h>
#include <dxxxlib.h>
```

```
#ifdef DTISC
#include <dtilib.h>
#include <msilib.h>
#endif

#ifdef FAXSC
#include <faxlib.h>
#endif

#include "sctools.h"


/*
 * Function prototypes
 */
static void nr_scerror(char *,...);


#if ( defined( __STDC__ ) || defined( __cplusplus ) )
int nr_scroute( int devh1, unsigned short devtype1,
                int devh2, unsigned short devtype2, unsigned char mode )
#else
int nr_scroute( devh1, devtype1, devh2, devtype2, mode )
   int            devh1;
   unsigned short devtype1;
   int            devh2;
   unsigned short devtype2;
   unsigned char  mode;
#endif
{
   SC_TSINFO sc_tsinfo;       /* SCBus Timeslots information structure */
   long      scts;            /* SCBus Timeslot */

   /*
    * Setup the SCBus Timeslots information structure.
    */
   sc_tsinfo.sc_numts = 1;
   sc_tsinfo.sc_tsarrayp = &scts;

   /*
    * Get the SCBus timeslot connected to the transmit of the first device.
    */
   switch (devtype1) {
   case SC_VOX:
      if (dx_getxmitslot(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: dx_getxmitslot ERROR: %s\n",
                                       ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;

   case SC_LSI:
      if (ag_getxmitslot(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: ag_getxmitslot ERROR: %s\n",
                                       ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;

#ifdef DTISC
   case SC_DTI:
      if (dt_getxmitslot(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: dt_getxmitslot ERROR: %s\n",
                                       ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
```

```
         return -1;
      }
      break;

   case SC_MSI:
      if (ms_getxmitslot(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: ms_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;
#endif

#ifdef FAXSC
   case SC_FAX:
      if (fx_getxmitslot(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: fx_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;
#endif

   default:
      nr_scerror("nr_scroute: %s: ERROR: Invalid 1st device type\n",
                                                                ATDV_NAMEP(devh1));
      return -1;
   }

   /*
    * Make the second device type listen to the timeslot that the first
    * device is transmitting on.  If a half duplex connection is desired,
    * then return.  Otherwise, get the SCBus timeslot connected to the
    * transmit of the second device.
    */
   switch (devtype2) {
   case SC_VOX:
      if (dx_listen(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot dx_listen %d ERROR: %s\n",
                                    ATDV_NAMEP(devh2),scts,ATDV_ERRMSGP(devh2));
         return -1;
      }

      if (mode == SC_HALFDUP) {
         return 0;
      }
if (dx_getxmitslot(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: dx_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         return -1;
      }
      break;

   case SC_LSI:
      if (ag_listen(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot ag_listen %d ERROR: %s\n",
                                    ATDV_NAMEP(devh2),scts,ATDV_ERRMSGP(devh2));
         return -1;
      }

      if (mode == SC_HALFDUP) {
         return 0;
      }

      if (ag_getxmitslot(devh2, &sc_tsinfo) == -1) {
```

```
         nr_scerror("nr_scroute: %s: ag_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         return -1;
      }
      break;

#ifdef DTISC
   case SC_DTI:
      if (dt_listen(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot dt_listen %d ERROR: %s\n",
                                    ATDV_NAMEP(devh2),scts,ATDV_ERRMSGP(devh2));
         return -1;
      }

      if (mode == SC_HALFDUP) {
         return 0;
      }

      if (dt_getxmitslot(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: dt_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         return -1;
      }
      break;

   case SC_MSI:
      if (ms_listen(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot ms_listen %d ERROR: %s\n",
                                    ATDV_NAMEP(devh2),scts,ATDV_ERRMSGP(devh2));
         return -1;
      }

      if (mode == SC_HALFDUP) {
         return 0;
      }

      if (ms_getxmitslot(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: ms_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         return -1;
      }
      break;
#endif

#ifdef FAXSC
   case SC_FAX:
      if (fx_listen(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot fx_listen %d ERROR: %s\n",
                                    ATDV_NAMEP(devh2),scts,ATDV_ERRMSGP(devh2));
         return -1;
      }

      if (mode == SC_HALFDUP) {
         return 0;
      }

      if (fx_getxmitslot(devh2, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: fx_getxmitslot ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         return -1;
      }
      break;
#endif

   default:
```

```
      nr_scerror("nr_scroute: %s: ERROR: Invalid 2nd device type\n",
                                                                  ATDV_NAMEP(devh2));
      return -1;
   }

   /*
    * Now make the first device listen to the SCBus timeslot that the
    * second device is transmitting on.
    */
   switch (devtype1) {
   case SC_VOX:
      if (dx_listen(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot dx_listen %d ERROR: %s\n",
                                 ATDV_NAMEP(devh1),scts,ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;

   case SC_LSI:
      if (ag_listen(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot ag_listen %d ERROR: %s\n",
                                 ATDV_NAMEP(devh1),scts,ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;

#ifdef DTISC
   case SC_DTI:
      if (dt_listen(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot dt_listen %d ERROR: %s\n",
                                 ATDV_NAMEP(devh1),scts,ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;

   case SC_MSI:
      if (ms_listen(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot ms_listen %d ERROR: %s\n",
                                 ATDV_NAMEP(devh1),scts,ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;
#endif

#ifdef FAXSC
   case SC_FAX:
      if (fx_listen(devh1, &sc_tsinfo) == -1) {
         nr_scerror("nr_scroute: %s: Cannot fx_listen %d ERROR: %s\n",
                                 ATDV_NAMEP(devh1),scts,ATDV_ERRMSGP(devh1));
         return -1;
      }
      break;
#endif
   }

   return 0;
}


void nr_scerror(char *fmt,   )
#ifdef PRINTON
   va_list args;

   /*
    * Make args point to the 1st unnamed argument and then print to stderr.
```

```
    */
   va_start(args, fmt);
   fmt = va_arg(args, char *);
   vfprintf(stderr, fmt, args);
   va_end(args);
#endif
}
```

See Also

- **nr_scunroute( )**

Name:	int nr_scunroute(devh1, devtype1, devh2, devtype2, mode)	
Inputs:	int devh1	• device handle
	unsigned short devtype1	• specifies the type of device for devh1
	int devh2	• device handle
	unsigned short devtype2	• specifies the type of device for devh2
	unsigned char mode	• specifies half or full duplex connection
Returns:	0 on success -1 on error	
Includes:	stdio.h, windows.h, srllib.h, dxxxlib.h and sctools.h. Optional dtilib.h, msilib.h, and faxlib.h.	
Category:	Routing convenience	
Mode:	Synchronous	

■ Description

The **nr_scunroute( )** convenience function breaks the connection between two devices. This function disconnects the full or half-duplex connection between two devices. This function is provided in a separate C source file called *sctools.c* in the *<install drive>:\<install directory>\dialogic\sctools* directory. The library version is in *sctools.lib* in the *<install drive>:\<install directory>\dialogic\lib* directory. The nr_sc prefix to the function signifies network (analog and digital) devices and resource (voice and FAX) devices accessible via the SCbus. See the *Digital Network Interface Software Reference* for digital interface device details and the *FAX Software Reference* for FAX device details.

NOTE: DTI, MSI, or FAX functionality may be conditionally compiled in or out of the function using the DTISC and/or FAXSC defines in the makefile provided with the function. If DTI or FAX functionality is being compiled in, then you must link with the DTI or FAX library. See the *Digital Network Interface Software Reference* for digital interface device details and the *FAX Software Reference* for FAX device details.

Parameter	Description
devh1:	Dialogic device handle.
devtype1:	Specifies the parameters for devh1.
	SC_VOX voice channel device

Parameter	Description	
	SC_LSI	analog channel device
	SC_DTI	digital time slot device
	SC_MSI	MSI station device
	SC_FAX	fax channel device
devh2:	Dialogic device handle.	
devtype2:	Specifies the type of device for **devh2**. See **devtype1** for list of defines.	
mode:	Specifies half or full duplex connection. The mode parameter contains one of the following defines from sctools.h to specify full or half duplex:	
	SC_FULLDUP	full duplex connection
	SC_HALFDUP	half duplex connection.
	The default mode value is SC_FULLDUP. When SC_HALFDUP is specified, then the function returns with the second device NOT listening (receive disconnected) to the SCbus time slot connected to the first device.	

■ Cautions

1. The devtype1 and devtype2 parameters must match the types of the device handles in devh1 and devh2.
2. If you have not defined DTISC and FAXSC when compiling the *sctools.c* file, you cannot use this function to route digital time slots, MSI stations, or fax channels.

■ Source Code

```
/*
 * Include files.
 */
#include <windows.h>
#include <stdio.h>
#include <srllib.h>
#include <dxxxlib.h>

#ifdef DTISC
#include <dtilib.h>
#include <msilib.h>
#endif

#ifdef FAXSC
```

```
#include <faxlib.h>
#endif

#include "sctools.h"


/*
 * Function prototypes
 */
static void nr_scerror(char *, ...);

#if ( defined( __STDC__ ) || defined( __cplusplus ) )
int nr_scunroute( int devh1, unsigned short devtype1,
                  int devh2, unsigned short devtype2, unsigned char mode )
#else
int nr_scunroute( devh1, devtype1, devh2, devtype2, mode )
   int            devh1;
   unsigned short devtype1;
   int            devh2;
   unsigned short devtype2;
   unsigned char  mode;
#endif
{
   short rc = 0;              /* Return code from the function */


   /*
    * Disconnect the receive of the second device from the SCBus timeslot.
    */
   switch (devtype2) {
   case SC_VOX:
      if (dx_unlisten(devh2) == -1) {
         nr_scerror("nr_scunroute: %s: dx_unlisten ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         rc = -1;
      }
      break;

   case SC_LSI:
      if (ag_unlisten(devh2) == -1) {
         nr_scerror("nr_scunroute: %s: ag_unlisten ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         rc = -1;
      }
      break;

#ifdef DTISC
   case SC_DTI:
      if (dt_unlisten(devh2) == -1) {
         nr_scerror("nr_scunroute: %s: dt_unlisten ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         rc = -1;
      }
      break;

case SC_MSI:
      if (ms_unlisten(devh2) == -1) {
         nr_scerror("nr_scunroute: %s: ms_unlisten ERROR: %s\n",
                                        ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         rc = -1;
      }
      break;
#endif

#ifdef FAXSC
```

```
   case SC_FAX:
      if (fx_unlisten(devh2) == -1) {
         nr_scerror("nr_scunroute: %s: fx_unlisten ERROR: %s\n",
                                          ATDV_NAMEP(devh2),ATDV_ERRMSGP(devh2));
         rc = -1;
      }
      break;
#endif

   default:
      nr_scerror("nr_scunroute: %s: ERROR: Invalid 2nd device type\n",
                                                                 ATDV_NAMEP(devh2));
      rc = -1;
   }

   /*
    * A half duplex connection has already been broken.  If this is all that
    * is required, then return now.
    */
   if (mode == SC_HALFDUP) {
      return rc;
   }

   /*
    * Disconnect the receive of the first device from the SCBus timeslot.
    */
   switch (devtype1) {
   case SC_VOX:
      if (dx_unlisten(devh1) == -1) {
         nr_scerror("nr_scunroute: %s: dx_unlisten ERROR: %s\n",
                                          ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         rc = -1;
      }
      break;

   case SC_LSI:
      if (ag_unlisten(devh1) == -1) {
         nr_scerror("nr_scunroute: %s: ag_unlisten ERROR: %s\n",
                                          ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         rc = -1;
      }
      break;

#ifdef DTISC
   case SC_DTI:
      if (dt_unlisten(devh1) == -1) {
         nr_scerror("nr_scunroute: %s: dt_unlisten ERROR: %s\n",
                                          ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         rc = -1;
      }
      break;

   case SC_MSI:
      if (ms_unlisten(devh1) == -1) {
         nr_scerror("nr_scunroute: %s: ms_unlisten ERROR: %s\n",
                                          ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         rc = -1;
      }
      break;
#endif

#ifdef FAXSC
   case SC_FAX:
      if (fx_unlisten(devh1) == -1) {
         nr_scerror("nr_scunroute: %s: fx_unlisten ERROR: %s\n",
```

```
                                        ATDV_NAMEP(devh1),ATDV_ERRMSGP(devh1));
         rc = -1;
      }
      break;
#endif

   default:
      nr_scerror("nr_scunroute: %s: ERROR: Invalid 1st device type\n",
                                                              ATDV_NAMEP(devh1));
      rc = -1;
   }

   return rc;
}

void nr_scerror(char *fmt, ...)
{
#ifdef PRINTON
   va_list args;
   char   *fmt;

   /*
    * Make args point to the 1st unnamed argument and then print to stderr.
    */
   va_start(args, fmt);
   fmt = va_arg(args, char *);
   vfprintf(stderr, fmt, args);
   va_end(args);
#endif
}
```

See Also

- **nr_scroute( )**

3. SCbus Function Reference

SCbus Function Reference Overview

A detailed reference of the library functions used for connecting the receive channel of a device to a transmitting channel of another device via the SCbus is provided in this chapter.

SCbus Function Descriptions

Name:	int ag_getctinfo(chdev, ct_devinfop)	
Inputs:	int chdev	• analog channel handle
	CT_DEVINFO *ct_devinfop	• pointer to device information structure
Returns:	0 on success -1 on error	
Includes:	dxxxlib.h	
Mode:	Synchronous	

■ Description

The **ag_getctinfo( )** function returns information about the analog device. This function provides information about an analog channel such as on a D/41ESC or D/160SC-LS board.

Parameter	Description
chdev:	Specifies the valid analog channel handle obtained when the channel was opened using **dx_open( )**.
ct_devinfop:	Specifies a pointer to the data structure CT_DEVINFO.

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

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

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

- The **ct_prodid** field contains a valid Dialogic product identification number for the device.
- The **ct_devfamily** specifies the device family and may contain either:

  CT_DFD41E analog channel of a D/41ESC board

  CT_DFSPAN analog channel of a D/160SC-LS board

- The **ct_devmode** field specifies a device mode field that is valid only for the D/41ESC board and may contain either:

CT_DMRESOURCE	analog channel not in use
CT_DMNETWORK	analog channel available to process calls from the telephone network

- The **ct_nettype** field specifies the type of network interface for the device. The two valid values are:

CT_NTNONE	D/41ESC board configured as a resource device; voice channels available for call processing; analog channels are disabled.
CT_NTANALOG	analog and voice devices on board are handling call processing

- The **ct_busmode** specifies the bus architecture used to communicate with other devices in the system. The two valid values are:

CT_BMPEB	PEB (PCM Expansion Bus) architecture
CT_BMSCBUS	SCbus architecture

- The **ct_busencoding** field describes the PCM encoding used on the bus. Valid values are:

CT_BEULAW	Mu-law encoding
CT_BEALAW	A-law encoding

■ Cautions

This function will fail if an invalid channel handle is specified.

■ Example

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

main()
{
   int  chdev;                              /* Channel device handle */
```

```
   CT_DEVINFO ct_devinfo;       /* Device information structure */

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

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADEXTTS	Cbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler library index number
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Linux system error

See also

- **dt_getctinfo( )**
- **dx_getctinfo( )**

Name:	int ag_getxmitslot(chdev, sc_tsinfop)	
Inputs:	int chdev	• analog channel handle
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 on error	
Includes:	dxxxlib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **ag_getxmitslot( )** function provides SCbus time slot number of the analog transmit channel. The SCbus time slot information is contained in an SC_TSINFO structure that also includes the number of the SCbus time slot connected to the analog transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **ag_getxmitslot( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
chdev:	Specifies the valid analog channel handle obtained when the channel was opened using **dx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

The sc_numts member of the SC_TSINFO structure must be initialized with the number of SCbus time slots requested which will be 1 for an analog channel. The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to a valid array of longs. Upon return from the function, the first element of the array will contain the number (between 0 and 1023) of the SCbus time slot on which the analog channel transmits.

An analog channel can transmit on only one SCbus time slot.

■ Cautions

This function will fail when:

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

■ Example

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


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

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

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

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADINDX	Invalid Switch Handler library index number
EDX_SH_BADLCTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)number
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel is already disconnected from SCbus time slot
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows system error

■ See Also

- **dt_listen( )**
- **dx_listen( )**
- **fx_listen( )**
- **ms_listen( )**

Name: int ag_listen (chdev, sc_tsinfop)
Inputs: int chdev • analog channel handle
SC_TSINFO *sc_tsinfop • pointer to SCbus time slot information structure
Returns: 0 on success
-1 on error
Includes: dxxxlib.h
Category: SCbus Routing
Mode: Synchronous

■ Description

The **ag_listen( )** function connects analog receive channel to SCbus time slot. This function uses the information stored in the SC_TSINFO structure to connect the analog receive (listen) channel to an SCbus time slot. This function sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the analog transmit channel

Due to analog signal processing on voice boards with on-board analog devices, a voice device and its corresponding analog device (analog device 1 to voice device 1, etc.) comprise a single channel. At system initialization, default SCbus routing is to connect these devices in full-duplex communications. See the *SCbus Routing Guide* for more information.

NOTE: SCbus convenience function **nr_scroute( )** includes **ag_listen( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
chdev:	Specifies the valid analog channel handle obtained when the channel was opened using **dx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

```
} SC_TSINFO;
```

The sc_numts member of the SC_TSINFO structure must be initialized with the number 1. The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to a valid array that contains a valid SCbus time slot number in its first array element. Upon return from the function, the analog receive channel will be connected to this SCbus time slot.

The SCbus time slot number contained in the array pointed to by sc_tsarrayp must be obtained, prior to calling **ag_listen( )**, by calling the appropriate **xx_getxmitslot( )** function.

Although multiple D/41ESC, D/160SC-LS, LSI/81SC, LSI/161SC, or VFX/40ESC channels may listen (be connected) to the same SCbus time slot, the analog receive (listen) channel can connect to only one SCbus time slot.

■ Cautions

This function will fail when:

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

■ Example

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


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

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

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

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function ATDV_LASTERR( ) to obtain the error code or use ATDV_ERRMSGP( ) to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel local time slot type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLTSCNCT	Channel is already connected to SCbus time slot
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows system error

■ See also

- **dx_getxmitslot( )**

- **dt_getxmitslot( )**
- **fx_getxmitslot( )**
- **ag_unlisten( )**

Name:	int ag_unlisten(chdev)	
Inputs:	int chdev	• analog channel handle
Returns:	0 on success -1 on error	
Includes:	dxxxlib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **ag_unlisten( )** function disconnects analog receive channel from SCbus. This function disconnects the analog receive (listen) channel from SCbus time slot it was listening to.

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

NOTE: SCbus convenience function **nr_scunroute( )** includes **ag_unlisten( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
chdev:	Specifies the valid analog channel handle obtained when the channel was opened using **dx_open( )**.

■ Cautions

This function will fail when:

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

■ Example

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

```
#include <errno.h>


main( )
{
   int chdev;              /* Voice Channel handle */

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function ATDV_LASTERR( ) to obtain the error code or use ATDV_ERRMSGP( ) to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel local time slot type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel already disconnected from SCbus time slot
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present

Equate	Returned When
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows system error

■ See also

- **ag_listen( )**

Name:	int dt_getctinfo(devh, ct_devinfop)	
Inputs:	int devh	• digital network interface device handle
	CT_DEVINFO *ct_devinfop	• pointer to device information structure
Returns:	0 on success -1 on error	
Includes:	srllib.h dtilib.h	
Mode:	Synchronous	

■ Description

The **dt_getctinfo( )** function gets SCbus digital interface information. This function returns information about the digital network interface device associated with the specified digital channel (time slot) (dtiBxTx) .

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

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

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

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

- The **ct_prodid** field contains a valid Dialogic product identification number for the device.

- The **ct_devfamily** specifies the device family and contains:

CT_DFSPAN specifies a D/240SC-T1 or D/300SC-E1 digital interface device

- The **ct_devmode** field is not valid for D/240SC-T1 or D/300SC-E1 devices.
- The **ct_nettype** field specifies the type of network interface for the device. The two valid values are:

CT_NTT1 specifies a D/240SC-T1 T-1 digital channel

CT_NTE1 specifies a D/300SC-E1 E-1 digital channel

- The **ct_busmode** specifies the bus architecture used to communicate with other devices in the system. The two valid values are:

CT_BMPEB PEB (PCM Expansion Bus) architecture

CT_BMSCBUS SCbus architecture

- The **ct_busencoding** field describes the PCM encoding used on the bus. Valid values are:

CT_BEULAW Mu-law encoding

CT_BEALAW A-law encoding

■ Cautions

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

■ Example

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

main( )
{
   int devh;                  /* Digital interface device handle */
   CT_DEVINFO ct_devinfo;     /* Device information structure */

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

```
    exit(1);
  }

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

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

■ Errors

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

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

■ See also

- **ag_getctinfo( )**

- **dx_getctinfo( )**

Name:	int dt_getxmitslot (devh, sc_tsinfop)	
Inputs:	int devh	• digital network interface device handle
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 on error	
Includes:	srllib.h dtilib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **dt_getxmitslot( )** function provides SCbus time slot number of the digital transmit channel. The SCbus time slot information is contained in a SC_TSINFO structure that also includes the number of the SCbus time slot connected to the digital network interface device transmit channel (T-1/E-1time slot).

NOTE: SCbus convenience function **nr_scroute( )** includes **dt_getxmitslot( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

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

The SC_TSINFO structure is declared as follows:

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

The sc_numts field of the SC_TSINFO structure must be initialized with the number of SCbus time slots requested (1 for a digital network interface device time slot). The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to a valid array of longs. Upon return from the function,

the first element of the array will contain the number (between 0 and 1023) of the SCbus time slot on which the digital network interface device transmits.

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

■ Cautions

This function will fail when:

- An invalid digital channel (T-1/E-1 time slot/device handle) is specified.
- A PEB time slot is requested.

■ Example

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


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

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

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

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

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

■ Errors

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

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

■ See also

- **ag_listen( )**
- **fx_listen( )**
- **dx_listen( )**

- **ms_listen( )**

Name:	int dt_listen (devh, sc_tsinfop)	
Inputs:	int devh	• digital network interface device handle
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 on error	
Includes:	srllib.h dtilib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **dt_listen( )** function connects digital receive channel to SCbus time slot. This function uses the information stored in the SC_TSINFO structure to connect the digital receive (listen) channel (T-1/E-1 time slot) to an SCbus time slot. This functon sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the digital transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **dt_listen( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

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

The SC_TSINFO structure is declared as follows:

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

The sc_numts field of the SC_TSINFO structure must be set to 1. The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array. The first element of this array must contain a valid SCbus time slot number

(between 0 and 1023) which was obtained by issuing an **xx_getxmitslot( )** function (xx = ag, dt, dx or fx). Upon return from the **dt_listen( )** function, the digital receive channel will be connected to this time slot.

Although multiple SCbus device channels may listen (be connected) to the same SCbus time slot, a digital receive (listen) channel can connect to only one SCbus time slot.

■ Cautions

This function will fail when:

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

■ Example

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

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

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

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

   /* Get SCbus time slot connected to transmit of voice channel 1 on board 1 */
   if (dx_getxmitslot(voxh, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(voxh));
     exit(1);
   }
   /* Open board 1 time slot 1 on digital interface device */
   if ((dtih = dt_open("dtiB1T1", 0)) == -1) {
      printf("Cannot open time slot dtiB1T1.  errno = %d", errno);
      xit(1);
   }

   /* Connect the receive of digital channel (time slot) 1 on board 1 to SCbus transmit
      time slot of voice channel 1*/
```

```
   if (dt_listen(dtih, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(dtih));
     exit(1);
}
```

■ Errors

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

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

See also

- **dt_unlisten( )**
- **ag_getxmitslot( )**
- **dx_getxmitslot( )**
- **fx_getxmitslot( )**
- **ms_getxmitslot( )**

Name:	int dt_unlisten(devh)	
Inputs:	int devh	• digital network interface device handle
Returns:	0 on success -1 on error	
Includes:	srllib.h dtilib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **dt_unlisten( )** function disconnects digital receive channel from SCbus. This function disconnects the digital receive (listen) channel (T-1/E-1 time slot) from the SCbus time slot it was listening to.

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

NOTE: SCbus convenience function **nr_scunroute( )** includes **dt_unlisten( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

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

■ Cautions

This function will fail when:

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

■ Example

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

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

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

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

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

■ Errors

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

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

Equate	Returned When
EDT_SH_LIBNOTINIT	Switch Handler library is uninitialized
EDT_SH_MISSING	Switch Handler is not present
EDT_SH_NOCLK	Switch Handler clock fallback failed
EDT_SYSTEM	Windows system error
EDT_TMOERR	Timed out waiting for reply from firmware

■ See also

- **dt_listen( )**

Name:	int dx_getctinfo(chdev, ct_devinfop)	
Inputs:	int chdev	• voice channel handle
	CT_DEVINFO *ct_devinfop	• pointer to device information structure
Returns:	0 on success -1 on error	
Includes:	dxxxlib.h	
Mode:	Synchronous	

■ Description

The **dx_getctinfo( )** function returns information about voice device This function provides information about a voice channel.

Parameter	Description
chdev:	Specifies the valid voice channel handle obtained when the channel was opened using **dx_open( )**.
ct_devinfop:	Specifies a pointer to the data structure CT_DEVINFO that will contain the voice channel device information.

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

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

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

- The **ct_prodid** field contains a valid Dialogic product identification number for the device.
- The **ct_devfamily** specifies the device family and may contain either:

CT_DFD41E voice channel of a D/41ESC board

CT_DFSPAN	voice channel of a D/240SC, D/320SC, D/240SC-T1, D/300SC-E1, or D/160SC-LS board

- The **ct_devmode** field specifies a device mode field that is valid only for the D/41ESC board and may contain either:

CT_DMRESOURCE	analog channel not in use
CT_DMNETWORK	analog channel available to process calls from the telephone network

- The **ct_nettype** field specifies the type of network interface for the device. Valid values are:

CT_NTNONE	D/41ESC board configured as a resource device: voice channels are available for call processing; analog channels are disabled.
CT_NTANALOG	analog and voice devices on board are handling call processing
CT_NTT1	D/240SC-T1 T-1 digital channel.
CT_NTE1	D/300SC-E1 E-1 digital channel.

- The **ct_busmode** specifies the bus architecture that the device is using to communicate with other devices in the system. Following are the two valid values:-

CT_BMPEB	PEB (PCM Expansion Bus) architecture
CT_BMSCBUS	SCbus architecture

- The **ct_busencoding** field describes the PCM encoding being used on the bus. Valid values are:-

CT_BEULAW	Mu-law encoding
CT_BEALAW	A-law encoding

■ Cautions

This function will fail if an invalid voice channel handle is specified.

■ Example

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

```
main( )
{
   int chdev;                  /* Channel device handle */
   CT_DEVINFO ct_devinfo;      /* Device information structure */


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


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

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

}
```

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADTYPE	Invalid local time slot channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present

Equate	Returned When
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Linux System Error

See Also

- **ag_getctinfo( )**
- **dt_getctinfo( )**

Name: int dx_getxmitslot(chdev, sc_tsinfop)
Inputs: int chdev • voice channel handle
SC_TSINFO *sc_tsinfop • pointer to SCbus time slot information structure
Returns: 0 on success
-1 on error
Includes: dxxxlib.h
Category: SCbus Routing
Mode: Synchronous

■ Description

The **dx_getxmitslot( )** function provides SCbus time slot number of voice transmit channel. The SCbus time slot information is contained in a SC_TSINFO structure that also includes the number of the SCbus time slot connected to the voice transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **dx_getxmitslot( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
chdev:	Specifies the voice channel device handle obtained when the channel was opened using **dx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

The sc_numts member of the SC_TSINFO structure must be initialized with the number of SCbus time slots requested (1 for a voice channel). The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array of longs. Upon return from the function, the first element of the array will contain the number (between 0 and 1023) of the SCbus time slot on which the voice channel transmits.

A voice channel can transmit on only one SCbus time slot.

■ Cautions

This function will fail when:

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

■ Example

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

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

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

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

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

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

■ Errors

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

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel is already disconnected from SCbus
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows System Error

See Also

- **ag_listen( )**
- **dt_listen( )**
- **fx_listen( )**

Name: int dx_listen(chdev, sc_tsinfop)
Inputs: int chdev — voice channel handle
SC_TSINFO *sc_tsinfop — pointer to SCbus time slot information structure
Returns: 0 on success
-1 on error
Includes: dxxxlib.h
Category: SCbus Routing
Mode: Synchronous

■ Description

The **dx_listen( )** function connects voice receive channel to SCbus time slot. This function uses the information stored in the SC_INFO structure to connect the voice receive (listen) channel to an SCbus time slot. This functon sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the voice transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **dx_listen( )** functionality. See the *SCbus Routing Guide* for more information on cconvenience functions.

Parameter	Description
chdev:	Specifies the voice channel handle obtained when the channel was opened using **dx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

The sc_numts field of the SC_TSINFO structure must be set to 1. The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array. The first element of this array must contain a valid SCbus time slot number (between 0 and 1023) which was obtained by issuing an **xx_getxmitslot( )** function (xx = ag, dt, dx, or fx). Upon return from the **dx_listen( )** function, the voice receive channel will be connected to the SCbus time slot.

Although multiple voice channels may listen (be connected) to the same SCbus time slot, the receive of a voice channel can connect to only one SCbus time slot.

■ Cautions

This function will fail when:

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

■ Example

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

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

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

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

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLTSCNCT	Channel is already connected to SCbus
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows System Error

■ See Also

- **ag_getxmitslot( )**
- **dt_getxmitslot( )**
- **fx_getxmitslot( )**
- **dx_unlisten( )**

Name:	int dx_unlisten(chdev)	
Inputs:	int chdev	• voice channel handle
Returns:	0 on success	
	-1 on error	
Includes:	dxxxlib.h	
Category:	SCbus Routing	
Mode:	Synchronous	

■ Description

The **dx_unlisten( )** function disconnects voice receive channel from SCbus. This function disconnects the voice receive (listen) channel from the SCbus time slot it was listening to.

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

NOTE: SCbus convenience function **nr_scunroute( )** includes **dx_unlisten( )** functionality. See the *SCbus Routing Guide* for more information on cconvenience functions.

Parameter	Description
chdev:	Specifies the voice channel handle obtained when the channel was opened using **dx_open( )**.

■ Cautions

This function will fail when:

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

■ Example

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

```
main( )
{
   int chdev;                  /* Voice Channel device handle */

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

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

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel already disconnected from SCbus
EDX_SH_LIBBSY	Switch Handler library busy

Equate	Returned When
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock failback failed
EDX_SYSTEM	Windows System Error

■ See Also

- **dx_listen( )**

Name:	int fx_getxmitslot(dev,sc_tsinfop)	
Inputs:	int dev	• FAX channel device handle
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 on failure	
Includes:	srllib.h dxxxlib.h faxlib.h	
Category:	SCbus Routing	
Mode:	synchronous	

■ Description

The **fx_getxmitslot( )** function provides SCbus time slot number of the FAX transmit channel. It returns the SCbus time slot information contained in a SC_TSINFO structure that includes the number of the SCbus time slot connected to the FAX transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **fx_getxmitslot( )** functionality. See the *SCbus Routing Software Reference* for more information on convenience functions.

Parameter	Description
dev:	Specifies the channel device handle obtained when the FAX device was opened using **fx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

The sc_numts member of the SC_TSINFO structure must be initialized with the number of SCbus time slots requested (1 for a FAX channel). The sc_tsarrayp member of the SC_TSINFO structure must be initialized with a pointer to a valid

array. Upon return from the function, the first element of the array will contain the number (between 0 and 1023) of the SCbus time slot on which the FAX channel transmits.

A FAX channel on a VFX/40ESC board can transmit on only one SCbus time slot.

■ Cautions

This function will fail when:

- An invalid FAX channel device handle is specified.
- The function is called for a device operating in PEB mode.

■ Example

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

main( )
{
   int dev;                /* Fax channel device handle. */
   SC_TSINFO sc_tsinfo;    /* Timeslot information structure. */
   long scts;              /* SCbus time slots. */
   .
   .
   /* Open the FAX channel resource device. */
   if ((dev = fx_open("dxxxB7C1", NULL)) == -1) {
      /* Error opening device. */
      printf("Error opening channel, errno = %d\n", errno);
      exit(1);
   }
   /* Fill in the SC_TSINFO structure time slot information. */
   sc_tsinfo.sc_numts = 1;
   sc_tsinfo.sc_tsarray = &scts;
   /* Get FAX device channel SCbus transmit time slot. */
   if (fx_getxmitslot(dev, &sc_tsinfo) == -1) {
      printf("Error message = %s", ATDV_ERRMSGP(dev));
```

```
    exit(1);
}

printf("Fax channel is transmitting on SCbus time slot %d\n", scts);
.
.
```

■ Errors

If this function returns -1, use **ATDV_LASTERR( )** and **ATDV_ERRMSGP( )** to retrieve one of the following error reasons (see the *Voice Software Reference - Features Guide* for error message information):

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel is already disconnected from SCbus
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows System Error

See also

- **ag_listen( )**
- **dt_listen( )**
- **dx_listen( )**
- **fx_listen( )**

Name:	int fx_listen(dev,sc_tsinfop)	
Inputs:	int dev	• FAX channel device handle
	SC_TSINFO *sc_tsinfop	• pointer to SCbus time slot information structure
Returns:	0 on success -1 on failure	
Includes:	srllib.h dxxxlib.h faxlib.h	
Category:	SCbus Routing	
Mode:	synchronous	

■ Description

The **fx_listen( )** function connects FAX listen channel to SCbus time slot. This function uses the information stored in the SC_TSINFO structure to connect the FAX receive (listen) channel to an SCbus time slot. This function sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the FAX transmit channel.

NOTE: SCbus convenience function **nr_scroute( )** includes **fx_listen( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
dev*:*	Specifies the valid FAX channel device handle obtained when the channel was opened using **fx_open( )**.
sc_tsinfop:	Specifies a pointer to the data structure SC_TSINFO.

The SC_TSINFO structure is declared as follows:

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

The sc_numts member of the SC_TSINFO structure must be set to 1. The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array. The first element of this array must contain a valid SCbus time slot number (between 0 and 1023) which was obtained by issuing a **xx_getxmitslot**( )

function (xx = ag, dl, dt or fx). Upon return from the **fx_listen**( ) function, the FAX receive channel will be connected to this time slot.

Although multiple SCbus device channels may listen (be connected) to the same SCbus time slot, the FAX receive (listen) channel can connect to only one SCbus time slot.

■ Cautions

This function will fail when:

- An invalid FAX channel device handle is specified.
- An invalid SCbus time slot is specified.
- The function is called for a device operating in PEB mode.

■ Example

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

main( )
{
   int voxdev;          /* Voice channel device handle. */
   int dev;          /* Fax channel device handle. */
   SC_TSINFO sc_tsinfo;     /* SCbus time slot information structure. */
   long scts;          /* SCbus time slot. */
   .
   .
   /* Open the FAX channel device. */
   if ((dev = fx_open("dxxxB7C1", NULL)) == -1) {
      /* Error opening device. */
      printf("Error opening channel, errno = %d\n", errno);
      exit(1);
   }
   /* Open the VOICE channel device on the D/160SC-LS. */
   if ((voxdev = dx_open("dxxxB1C1", NULL)) == -1) {
      /* Error opening device. */
      printf("Error opening channel, errno = %d\n", errno);
```

```
    exit(1);
}
  .
  .
/*
 * Break the full-duplex connection between the Voice
 * channel device and the Network analog device.
 * Use the SCbus routing cconvenience function nr_scunroute( ).
 */
if (nr_scunroute(voxdev, SC_VOX, voxdev, SC_LSI, SC_FULLDUP)== -1) {
    /* Error during SCbus unroute. */
    printf("Error unrouting channel\n");
    printf("Error - %s (error code %d)\n",
       ATDV_ERRMSGP(voxdev), ATDV_LASTERR(voxdev));
    if (ATDV_LASTERR(voxdev) == EDX_SYSTEM) {
        printf("errno = %d\n", errno);
    }
}
  /*
   * Set full-duplex connection between the FAX
   * channel device and the Network analog device.
   */


/* Fill in the SC_TSINFO structure time slot information. */
sc_tsinfo.sc_numts = 1;
sc_tsinfo.sc_tsarray = &scts;


/* Get Network analog device's SCbus transmit time slot. */
if (ag_getxmitslot(voxdev, &sc_tsinfo) == -1) {
    printf("Error message = %s", ATDV_ERRMSGP(voxdev));
    exit(1);
}
/*
 * Connect the FAX channel to "listen" to the Network
 * channel's SCbus transmit time slot. Pass the time slot
 * information in the SC_TSINFO structure to fx_listen( ).
 */
if (fx_listen(dev, &sc_tsinfo) == -1) {
   printf("Error message = %s", ATDV_ERRMSGP(dev));
   exit(1);
}
  .
  .
/* Complete full-duplex connection between the FAX channel device
 * and the Network channel device using fx_getxmitslot( )
 * and ag_listen( ).
 */
  .
  .
```

```
/* Call FAX API functions for FAX transfers. */
  .
  .
```

■ Errors

If this function returns -1, use **ATDV_LASTERR( )** and **ATDV_ERRMSGP( )** to retrieve one of the following error reasons (see the *Voice Software Reference - Features Guide* for error message information):

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLTSCNCT	Channel is already connected to SCbus
EDX_SH_LIBBSY	Switch Handler library busy-
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock fallback failed
EDX_SYSTEM	Windows System Error-

■ See also

- **ag_getxmitslot( )**
- **dt_getxmitslot( )**
- **dx_getxmitslot( )**
- **fx_unlisten( )**

Name:	int fx_unlisten(dev)	
Inputs:	int dev	• FAX channel device handle
Returns:	0 on success -1 on failure	
Includes:	srllib.h dxxxlib.h faxlib.h	
Category:	SCbus Routing	
Mode:	synchronous	

■ Description

The **fx_unlisten( )** function disconnects FAX receive channel from SCbus.

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

NOTE: SCbus convenience function **nr_scunroute( )** includes **fx_unlisten( )** functionality. See the *SCbus Routing Guide* for more information on convenience functions.

Parameter	Description
dev:	Specifies the FAX channel device handle obtained when the channel was opened using **fx_open( )**.

■ Cautions

This function will fail when:

- An invalid FAX channel device handle is specified.
- The function is called for a device operating in PEB mode.

■ Example

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

```
#include <errno.h>

main( )
{
     int dev;          /* Fax channel device handle. */
     .
     .
     /* Open the FAX channel resource. */
     if ((dev = fx_open("dxxxB7C1", NULL)) == -1) {
          /* Error opening device. */
          printf("Error opening channel, errno = %d\n", errno);
          exit(1);
     }
          .
     .
     /*
      * Disconnect the FAX channel device from "listening" to an
      * SCbus transmit time slot.
      */
     if (fx_unlisten(dev) == -1) {
          printf("Error message = %s", ATDV_ERRMSGP(dev));
          exit(1);
     }
     .
     .
```

■ Errors

If this function returns -1, use ATDV_LASTERR( ) and ATDV_ERRMSGP( ) to retrieve one of the following error reasons (see the *Voice Software Reference - Features Guide* for error message information):

Equate	Returned When
EDX_BADPARM	Parameter error
EDX_SH_BADCMD	Command is not supported in current bus configuration
EDX_SH_BADEXTTS	SCbus time slot is not supported at current clock

Equate	Returned When
	rate
EDX_SH_BADINDX	Invalid Switch Handler index number
EDX_SH_BADLCLTS	Invalid channel number
EDX_SH_BADMODE	Function not supported in current bus configuration
EDX_SH_BADTYPE	Invalid channel type (voice, analog, etc.)
EDX_SH_CMDBLOCK	Blocking command is in progress
EDX_SH_LCLDSCNCT	Channel already disconnected from SCbus
EDX_SH_LIBBSY	Switch Handler library busy
EDX_SH_LIBNOTINIT	Switch Handler library uninitialized
EDX_SH_MISSING	Switch Handler is not present
EDX_SH_NOCLK	Switch Handler clock failback failed
EDX_SYSTEM	Windows System Error

■ See also

- **fx_listen( )**

Name: int ms_getxmitslot(devh,tsinfop)

Inputs: int devh
- station handle

SC_TSINFO *tsinfop
- pointer to the SCbus time slot information structure

Returns: 0 on success
-1 on failure

Includes: srllib.h
dtilib.h
msilib.h

Category: SCbus Routing

Mode: asynchronous

■ Description

The **ms_getxmitslot( )** function returns the SCbus time slot information contained in a SC_TSINFO structure that includes the number of the SCbus time slot connected to the MSI/SC station transmit channel.

Parameter	Description
devh:	The station device handle.
tsinfop:	The pointer to the SCbus time slot information structure.

On return from the function, the SC_TSINFO structure contains the number of SCbus time slots connected to the transmit of the station and points to the array that contains the SCbus time slots (between 0 and 1023). The SC_TSINFO structure is declared as follows:

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

For MSI/SC station devices, the sc_numts element of the SC_TSINFO array should be set to 1. The sc_tsarrayp must point to an array of type long and, upon return, its first element will contain the SCbus time slot on which the station is transmitting data.

NOTE: The transmit of an MSI/SC station device can be connected to only one external SCbus time slot.

■ Cautions

This function fails when:

- An invalid station device handle is specified.
- The device is operating in PEB mode.

Even though the **ms_getxmitslot( )** function provides a valid SCbus time slot of where the station transmit channel connects, the actual transmission of data will not occur until the station receive (listen) channel is connected to a valid SCbus time slot.

■ Example

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

int     chdev;             /* Station dev descriptor */
SC_TSINFO     tsinfo;      /* Time slot information structure */
long     scts;             /* SCbus time slot */

/* Open board 1, station 1 device */
if ((chdev1 = ms_open("msiB1C1",0)) == -1) {
     printf( "Cannot open MSI B1, C1: errno=%d", errno);
     exit(1);
}

/* Set up SC_TSINFO structure */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarray = &scts;

/* Get time slot on which MSI board 1, channel 1 is xmitting */
if (ms_getxmitslot(chdev1,&tsinfo) == -1) {
     printf("Error message = %s", ATDV_ERRMSGP(chdev));
     exit(1);
}

printf("msiB1C1 is transmitting on SCbus time slot %d",scts);
```

■ Errors

If the function returns a -1 indicating an error, the error code may be retrieved by calling the SRL standard attribute function **ATDV_LASTERR( )**. A more descriptive error message may be retrieved by using the function **ATDV_ERRMSGP( )**.

See Also

- **ms_listen( )**

Name:	int ms_listen(devh,tsinfop)	
Inputs:	int devh	• station handle
	SC_TSINFO *tsinfop	• pointer to the SCbus time slot information structure
Returns:	0 on success -1 on failure	
Includes:	srllib.h dtilib.h msilib.h	
Category:	SCbus Routing	
Mode:	asynchronous	

■ Description

The **ms_listen( )** function connects the receive of a station to an SCbus time slot. This function uses the information stored in the SC_TSINFO structure to connect the digital receive (listen) channel (T-1/E-1 time slot) to an SCbus time slot. This functon sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the digital transmit channel.

Parameter	Description
devh:	The board device handle.
tsinfop:	The pointer to the SCbus time slot information structure.

A pointer to the SC_TSINFO structure is passed to the function. It contains two fields: the first field specifies the total number of SCbus time slots to which the connection is to be made, and the second is a pointer to an array which contains the SCbus time slots (between 0 and 1023) to be connected to the receive of the station device. The SC_TSINFO structure is declared as follows:

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

For MSI/SC station devices, the sc_numts element of the SC_TSINFO array should be set to 1. The sc_tsarrayp must point to an array of type long. The first element of the array will contain a valid SCbus time slot number (0 to 1023)

which was obtained by issuing the appropriate **xx_getxmitslot( )** function call (xx = ag, dt, dx or fx).

NOTES: 1. An MSI/SC station device can listen to only one time slot at a time. However, multiple devices may listen to a time slot at the same time.

NOTES: 2. Dialogic recommends that you unlisten before invoking this function.

■ Cautions

This function fails when:

- An invalid station handle is specified.
- The SCbus time slot number is invalid.
- The device is operating in PEB mode.

■ Example

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

int     chdev1, tsdev4;    /* Chan dev descriptor variables */
SC_TSINFO     tsinfo;      /* Time slot information structure */
long     scts;             /* SCbus time slot */

/* Open board 1, channel 1 device */
if ((chdev1 = ms_open("msiB1C1",0)) == -1) {
     printf( "Cannot open MSI B1, C1: errno=%d", errno);
     exit(1);
}

/* Open board 1, time slot 4 device */
if ((tsdev1 = dt_open("dtiB1T4",0)) == -1) {
     printf( "Cannot open DTI B1, T4: errno=%d", errno);
     exit(1);
}
/* Set up SC_TSINFO structure */
tsinfo.sc_numts = 1;
tsinfo.sc_tsarray = &scts;

/* Get time slot on which DTI board 1, time slot 4 is xmitting */

if (dt_getxmitslot(tsdev4,&tsinfo) == -1) {
     printf("Error message = %s", ATDV_ERRMSGP(chdev));
     exit(1);
}

/* Make MSI board 1, station 1 listen to transmit time slot
     of DTI Board 1 time slot 4 on SCbus */
if (ms_listen(chdev1,&tsinfo) == -1) {
     printf("Error Message = %s",ATDV_ERRMSGP(tsdev4));
     exit(1);
}
```

Errors

If the function returns a -1 indicating an error, the error code may be retrieved by calling the SRL standard attribute function **ATDV_LASTERR( )**. A more descriptive error message may be retrieved by using the function **ATDV_ERRMSGP( )**.

See Also

- **ms_getxmitslot( )**
- **ms_unlisten( )**

Name:	int ms_unlisten(devh)	
Inputs:	int devh	• station handle
Returns:	0 on success	
	-1 on failure	
Includes:	srllib.h	
	dtilib.h	
	msilib.h	
Category:	SCbus Routing	
Mode:	asynchronous	

■ Description

The **ms_unlisten( )** function disconnects the receive of a station from an SCbus time slot.

Parameter	Description
devh:	The station device handle.

■ Cautions

This function fails when:

- An invalid station device handle is specified.
- The device is operating in PEB mode.

■ Example

```
#include     "srllib.h"
#include     "dxxxlib.h"
#include     "dtilib.h"

int     chdev          /* Station device handle */

/* Open board 1, channel 1 */
if ((chdev = ms_open("msiB1C1",0)) == -1) {
     printf("Cannot open channel msiB1C1. errno = %d", errno);
     exit(1);
}

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

■ Errors

When the function fails and returns a -1, the specific error code may be retrieved by calling the SRL standard attribute function **ATDV_LASTERR( )** to obtain the error code. A more descriptive error message may be retrieved using the function **ATDV_ERRMSGP( )**.

■ See Also

- **ms_listen( )**

4. SCbus Data Structure Reference

SCbus Data Structure Overview

The data structures used by individual SCbus routing functions are described in this chapter. These structures are used to control the operation of functions and to return information. The data structures are defined in the *voxlib.h* file.

The SCbus data structures include the SCbus time slot information (SC_TSINFO) structure that contains the numeric quantity of SCbus time slots (typically 1) assigned to a device and a pointer to an array that contains the SCbus time slot number(s).

The **xx_getxmitslot( )** functions get the information to fill the data structure. Then the **xx_listen( )** functions use this information to connect two Dialogic devices.

4.1. Channel/Time slot DEVice INFOrmation (CT_DEVINFO)

The CT_DEVINFO structure supplies information about a device. This structure is used by the SCbus routing functions identified by the suffix **_getctinfo( )**. On return from the function, the CT_DEVINFO structure contains the relevant information and is defined as follows:

```
typedef struct {
          unsigned long      ct_prodid;
          unsigned char      ct_devfamily;
          unsigned char      ct_devmode;
          unsigned char      ct_nettype;
          unsigned char      ct_busmode;
          unsigned char      ct_busencoding;
          unsigned char      ct_rfu[7];          /* reserved for future use */
} CT_DEVINFO;
```

Valid values for each of the members of the CT_DEVINFO structure are defined in *voxlib.h*.

- The **ct_prodid** field contains a valid Dialogic product identification number for the device [length: 4 (unsigned long)].
- The **ct_devfamily** specifies the device family [length: 1 (unsigned char)] and may contain either:

CT_DFD41E	analog or voice channel of a D/41ESC or VFX/40ESC board
CT_DFSPAN	analog channel of a D/160SC-LS board, a voice channel of a D/240SC, D/320SC, D/240SC-T1, D/300SC-E1 or D/160SC-LS board, or a digital channel of a D/240SC-T1 or D/300SC-E1 board
CT_DFMSI	a station on an MSI board

- The **ct_devmode** field specifies a device mode field [length: 1 (unsigned char)] that is valid only for the D/41ESC or VFX/40ESC board and may contain either:

CT_DMRESOURCE	analog channel not in use
CT_DMNETWORK	analog channel available to process calls from the telephone network

- The **ct_nettype** field specifies the type of network interface for the device [length: 1 (unsigned char)]. Valid values are:

CT_NTNONE	D/41ESC or VFX/40ESC board configured as a resource device: voice channels are available for call processing; analog channels are disabled.
CT_NTANALOG	analog and voice devices on board are handling call processing
CT_NTT1	D/240SC-T1 T-1 digital network interface
CT_NTE1	D/300SC-E1 E-1 digital network interface
CT_NTMSI	MSI/SC station interface

- The **ct_busmode** specifies the bus architecture used to communicate with other devices in the system [length: 1 (unsigned char)]. The two valid values are:

CT_BMPEB	PEB (PCM Expansion Bus) architecture
CT_BMSCBUS	SCbus architecture

- The **ct_busencoding** field describes the PCM encoding used on the bus [length: 1 (unsigned char)]. Valid values are:

CT_BEULAW	Mu-law encoding
CT_BEALAW	A-law encoding

4.2. SCbus Time Slot INFOrmation (SC_TSINFO)

The SC_TSINFO structure contains the numeric quantity defining how many SCbus time slots are associated with a particular device (typically 1) and a pointer to an array that will hold the actual SCbus time slot number(s) (valid numbers are 0 to 1023). The SC_TSINFO structure is used by the SCbus routing functions identified by the suffix:

- **_getxmitslot( )** to supply SCbus time slot information about a device and
- **_listen( )** to use this time slot information to connect two Dialogic devices.

The SC_TSINFO structure is defined as follows:

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

The sc_numts field of the SC_TSINFO structure must be initialized with the number of SCbus time slots, typically 1. The sc_tsarrayp field must be initialized with a pointer to an array of long integers.

Appendix A

SCbus Routing Function Summary

ag_functions

ag_getctinfo( )	returns information about analog device
ag_getxmitslot( )	returns SCbus time slot information contained in a structure that includes the number of the SCbus time slot connected to the analog transmit channel
ag_listen( )	connects analog receive (listen) channel to an SCbus time slot
ag_unlisten( )	disconnects analog receive (listen) channel from SCbus time slot

dt_ functions

dt_getctinfo( )	returns information about digital interface device
dt_getxmitslot( )	returns SCbus time slot information contained in a structure that includes the number of the SCbus time slot connected to the digital transmit channel
dt_listen( )	connects digital receive (listen) channel to an SCbus time slot
dt_unlisten( )	disconnects digital receive (listen) channel from SCbus time slot

dx_ functions

dx_getctinfo( )	returns information about voice device
dx_getxmitslot( )	returns SCbus time slot information contained in a structure that includes the number of the SCbus time slot connected to the voice transmit channel
dx_listen( )	connects voice receive (listen) channel to an SCbus time slot
dx_unlisten( )	disconnects voice receive (listen) channel from SCbus time slot

fx_ functions

fx_getxmitslot( )	returns SCbus time slot information contained in a structure that includes the number of the SCbus time slot connected to the FAX transmit channel
fx_listen( )	connects FAX receive (listen) channel to an SCbus time slot
fx_unlisten( )	disconnects FAX receive (listen) channel from SCbus time slot

ms_ functions

ms_getxmitslot( )	returns SCbus time slot information contained in a structure that includes the number of the SCbus time slot connected to the MSI station
ms_listen( )	connects an MSI station to an SCbus time slot
ms_unlisten( )	disconnects MSI station from SCbus time slot

nr_ functions

nr_scroute( )	makes half or full-duplex connection between two Dialogic devices
nr_scunroute( )	breaks half or full-duplex connection between two Dialogic devices

Appendix B

SCbus-related Publications

This appendix lists publications you should refer to for additional information on Dialogic products and SCbus technology.

SCbus Technology

- *SCbus Routing Guide*
- *SCbus Configuration Planning Guide*

Dialogic Software References

- *Voice Software Reference - Features Guide*
- *Voice Software Reference - Programmer's Guide*
- *Voice Software Reference - Standard Runtime Library*
- *Voice Software Installation Reference*
- *Digital Network Interface Software Reference*
- *FAX Software Reference*
- *MSI Software Reference*

Index

H

I

L

M

N

P

R

S

T

V