



# **Modular Station Interface API for Linux and Windows Operating Systems**

**Programming Guide**

---

*October 2004*



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

This Modular Station Interface API for Linux and Windows Operating Systems Programming Guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without express written consent of Intel Corporation.

Copyright © 1998 - 2004, Intel Corporation

BunnyPeople, Celeron, Chips, Dialogic, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel Centrino, Intel Centrino logo, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Xeon, Intel XScale, IPLink, Itanium, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, Sound Mark, The Computer Inside., The Journey Inside, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\* Other names and brands may be claimed as the property of others.

Publication Date: October 2004

Document Number: 05-1907-002

Intel Converged Communications, Inc.  
1515 Route 10  
Parsippany, NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website at:  
<http://developer.intel.com/design/telecom/support>

For **Products and Services Information**, visit the Intel Telecom Products website at:  
<http://www.intel.com/design/network/products/telecom>

For **Sales Offices** and other contact information, visit the Where to Buy Intel Telecom Products page at:  
<http://www.intel.com/buy/wtb/wtb1028.htm>



# Contents

---

	<b>Revision History</b> .....	1
	<b>About This Publication</b> .....	3
	Purpose .....	3
	Intended Audience .....	3
	How to Use This Publication .....	3
	Related Information .....	4
<b>1</b>	<b>Modular Station Interface API Description</b> .....	5
1.1	MSI-Supported Boards .....	5
1.2	Modular Station Interface API Features .....	5
1.2.1	Station Connectivity .....	6
1.2.2	Conferencing .....	6
1.2.3	Extended Connection .....	6
1.2.4	Resource Allocation .....	7
1.2.5	TDM Bus Routing .....	8
<b>2</b>	<b>Event Handling</b> .....	9
<b>3</b>	<b>Error Handling</b> .....	11
<b>4</b>	<b>Application Development Guidelines</b> .....	13
4.1	General Guidelines .....	13
4.1.1	Using Symbolic Defines .....	13
4.1.2	Including Header Files .....	13
4.1.3	Checking Return Codes .....	14
4.2	Initialization .....	14
4.2.1	Configuring DM3 Boards .....	14
4.2.2	Configuring MSI/SC Boards .....	14
4.2.3	Setting Event Mask on MSI Stations .....	15
4.2.4	Initializing MSI Stations .....	16
4.2.5	Terminating .....	16
4.3	Aborting .....	16
4.4	Standard Attribute Functions .....	16
<b>5</b>	<b>Building Applications</b> .....	19
5.1	Include Files .....	19
5.2	Compiling and Linking .....	19
<b>6</b>	<b>MSI/SC Platform Description</b> .....	21
6.1	MSI Hardware Overview .....	21
6.2	Typical Applications .....	21
6.3	Compatibility with other MSI boards .....	22
6.4	MSI Board Functional Description .....	23
	<b>Glossary</b> .....	25

<b>Index</b> .....	29
--------------------	----

## *Figures*

---

1	MSI Board Block Diagram . . . . .	24
---	-----------------------------------	----

## Tables

---

1	MSI Inputs for Event Management Functions .....	9
2	MSI Returns from Event Management Functions.....	9
3	Standard Attribute Functions .....	16



## Revision History

---

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-1907-002	October 2004	<a href="#">Error Handling</a> chapter: Updated system error handling, removed errno, and errno.h (PTR 28013). <a href="#">Include Files</a> section: Removed errno.h from list of files (PTR 28013). <a href="#">Including Header Files</a> section: Removed errno.h from list of files (PTR 28013).
05-1907-001	January 2003	Initial version of document. Much of the information contained in this document was previously published in the <i>MSI/SC Software Reference for Linux and Windows</i> , document number 05-1218-004.







# About This Publication

---

The following topics provide information about this publication:

- [Purpose](#)
- [Intended Audience](#)
- [How to Use This Publication](#)
- [Related Information](#)

## Purpose

This publication provides programming guidelines for the Modular Station Interface (MSI) API library. It is a companion document to the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference*, which provides details on functions and parameters of the MSI software.

## Intended Audience

This guide is intended for software developers who will access the modular station interface software. This may include any of the following:

- Distributors
- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

## How to Use This Publication

Refer to this publication after you have installed the hardware and the system software which includes the modular station interface software. This publication assumes that you are familiar with the Linux\* or Windows\* operating system and the C programming language.

The information in this guide is organized as follows:

[Chapter 1, “Modular Station Interface API Description”](#), introduces the modular station interface software and its key features.

[Chapter 2, “Event Handling”](#), defines events returned by the modular station interface library and discusses different ways of handling events.

[Chapter 3, “Error Handling”](#), presents information on how to obtain error codes and handle errors.

[Chapter 4, “Application Development Guidelines”](#), discusses methods of developing MSI-based applications.

[Chapter 5, “Building Applications”](#), contains library header file information and guidelines for compiling and linking your code.

[Chapter 6, “MSI/SC Platform Description”](#), describes the Intel® Dialogic® MSI/SC-Global Series boards, including a hardware overview and function block diagram.

The [Glossary](#) provides definitions of key terms used in this document.

## Related Information

For more information, refer to the following:

- *Modular Station Interface API Library Reference*
- *Voice API Library Reference*
- *Voice API Programming Guide*
- *Standard Runtime Library API Library Reference*
- *Standard Runtime Library API Programming Guide*
- <http://developer.intel.com/design/telecom/support/> (for technical support)
- <http://www.intel.com/design/network/products/telecom/> (for product information)

# Modular Station Interface API Description

---

1

This chapter contains the following topics:

- [MSI-Supported Boards . . . . . 5](#)
- [Modular Station Interface API Features . . . . . 5](#)

## 1.1 MSI-Supported Boards

The modular station interface (MSI) software is supported on more than one type of hardware platform. The following boards use the MSI software:

- Intel® NetStructure™ High Density Station Interface (HDSI) boards
- Intel® Dialogic® Integrated (DI) Station Interface boards
- Intel® Dialogic MSI/SC-Global Series boards (supported on Intel® Dialogic System Release 5.x only)

*DM3 boards* is a collective name used in this document to refer to products that are based on the Intel® Dialogic DM3 mediastream architecture. The Intel® NetStructure High Density Station Interface (HDSI) boards and Intel® Dialogic Integrated (DI) Station Interface boards are DM3 boards.

*Springware boards* refer to boards based on earlier-generation architecture, such as the Intel® Dialogic MSI/SC-Global Series boards.

Although an MSI feature may be supported on both DM3 and Springware boards, there may be some restrictions on its use. For example, some library functions or parameter values may not be supported. For details, see the function reference descriptions in the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference*.

## 1.2 Modular Station Interface API Features

The following features of the modular station interface software are discussed in this section:

- [Station Connectivity](#)
- [Conferencing](#)
- [Extended Connection](#)
- [Resource Allocation](#)
- [TDM Bus Routing](#)

## 1.2.1 Station Connectivity

The key functionality of the modular station interface (MSI) software is station connectivity, which includes the following features:

- detecting when a station goes on-hook, off-hook, or hookflash
- ringing phones with standard or distinctive rings
- ringing phones and sending frequency shift keyed (FSK) data (DM3 boards only)
- providing zip tone (MSI boards only)

**Note:** The MSI software is not capable of taking a station on-hook or off-hook. Instead, the MSI software reports what has happened on the channel or rings the phone.

## 1.2.2 Conferencing

**Note:** Conferencing is supported on Intel® Dialogic® Integrated (DI) Station Interface boards and MSI/SC-Global Series boards only.

The MSI software provides the following conferencing features:

### Monitor

This feature allows a conference to be monitored by several people without interrupting the conference.

### Coach

Typically a supervisor. The supervisor, or coach, while monitoring a conversation between a pupil and a client, can talk to the pupil in confidence. The pupil, however, cannot reply in confidence to the coach.

### Pupil

Typically, the agent who is heard by all parties in the conference. The pupil is the only conference participant who hears the coach.

**Note:** A conference may include only one coach and one pupil at any given time. There may be more than one client, but conference size is limited to the maximum number of participants permitted in the conference.

## 1.2.3 Extended Connection

**Note:** Extended connection is supported on Intel® Dialogic® MSI/SC-Global Series boards only.

A connection is defined as a full-duplex, TDM bus routing between two parties. One of the parties in the connection must be a station on an MSI-supported board. An extended connection allows a coach to join a connection at anytime without interrupting the conversation between the pupil and the client.

An extended connection is a connection where there is a third party, herein referred to as the *connection extender*, that can always hear what the other two parties are saying. The connection extender's input in the connection is application defined. If the connection extender has no input, it is in monitor mode. If the connection extender can talk to only one party, designated as the pupil, it

is in coach mode. If the connection extender can talk to both members of the connection, it is in participant mode.

- Notes:**
1. A connection may be set up using the TDM bus routing convenience function, `nr_scroute()`. For details on this function, refer to the *Voice API Library Reference*.
  2. The application is responsible for setting up a connection prior to extending it. The MSI software does not check for the presence of a connection between parties in order to extend it.

## 1.2.4 Resource Allocation

Each MSI-supported board has 32 resources managed by the application.

- Notes:**
1. For details on the functions below, see the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference*.
  2. The channel selector of the party does not affect the resource usage.
  3. Conferencing is supported on Intel® Dialogic® Integrated (DI) Station Interface boards and MSI/SC-Global Series boards only. Each of these hardware platforms has different conference size limits.  
For DI boards: refer to the Release Notes for applicable conference size limits.  
For MSI/SC boards: a conference is limited to eight parties. A monitor is counted as one of the eight parties.
  4. For MSI/SC boards: when zip tone support is enabled, 31 conferencing resources are available.

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

### `ms_setbrdparm()`

When `parm_id` = MSG\_ZIPENA and `value` = MS\_ZIPENABLE, one resource will be used

When `parm_id` = MSG\_ZIPENA and `value` = MS\_ZIPDISABLE, one resource will be freed

### `ms_estconf()`

Uses the total number of parties in the conference

### `ms_addtoconf()`

Uses one resource every time a party is added to a conference

### `ms_remfromconf()`

Frees one resource

### `ms_delconf()`

Frees all resources in use by the conference

### `ms_monconf()`

Uses one resource

### `ms_unmonconf()`

Frees one resource

### `ms_estxtdcon()`

Uses three resources

### `ms_delxtdcon()`

Frees three resources

### 1.2.5 TDM Bus Routing

TDM routing functions are used in time division multiplexing (TDM) bus configurations, which include the CT Bus and SCbus. A TDM bus is resource sharing bus that allows information to be transmitted and received among resources over multiple time slots.

TDM routing functions enable the application to make or break a connection between voice, telephone network interface, and other resource channels connected via TDM bus time slots. Each device connected to the bus has a transmit component that can transmit on a time slot and a receive component that can listen to a time slot.

The transmit component of each channel of a device is assigned to a time slot at system initialization and download. To listen to other devices on the bus, the receive component of the device channel is connected to any one time slot. Any number of device channels can listen to a time slot.

**Note:** When you see references to the SCbus or SCbus routing, this information also applies to the CT Bus. That is, the physical interboard connection can be either SCbus or CT Bus. The SCbus protocol is used and the SCbus routing API applies to all the boards regardless of whether they use an SCbus or CT Bus physical interboard connection.

The Event Management functions in the Standard Runtime Library (SRL) retrieve and handle MSI termination events for the `ms_setevtmsk()`, `ms_tstcom()`, and `ms_tstdat()` functions. For a description of MSI events and event types, see the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference*.

The following tables list Event Management functions and their relevance to MSI-based applications. For function details, refer to the *Standard Runtime Library API Library Reference*.

**Table 1. MSI Inputs for Event Management Functions**

Event Management Function	MSI specific Input	Value
<code>sr_enbhdlr()</code> Enable event handler	event type	MSEV_NORING MSEV_RING MSEV_SIG EVT DTEV_COMRSP DTEV_DATRSP
<code>sr_dishdlr()</code> Disable event handler	event type	Same as above
<code>sr_waitevt()</code> Wait for next event	N/A	N/A
<code>sr_waitevtEx()</code> Extended wait event	list of device handles	N/A

**Table 2. MSI Returns from Event Management Functions**

Event Management Function	MSI specific Input	Value
<code>sr_getevtdev()</code> Get Dialogic device handle	device	MSI device handle
<code>sr_getevttype()</code> Get event type	event type	MSEV_NORING MSEV_RING MSEV_SIG EVT DTEV_COMRSP DTEV_DATRSP
<code>sr_getevtlen()</code> Get event length	event length	Number of bytes in the data returned
<code>sr_getevtdatap()</code> Get pointer to event data	event data	Pointer to variable containing the value of a selected bitmask





This chapter describes error handling for the modular station interface (MSI) software.

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

- 0  
function success
- 1  
function error

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

- Notes:**
1. The **ms\_open()** function is the exception to the above error-handling rules. An **ms\_open()** function call returns a device handle if the function call is successful. A device handle is a non-zero value. If **ms\_open()** fails, the return code is -1, and it indicates a system error.
  2. The Standard Attribute functions **ATDV\_LASTERR()** and **ATDV\_ERRMSGP()** can be used to obtain the last error that occurred on a device. Refer to the *Standard Runtime Library API Library Reference* for more information on these functions.
  3. If the error returned by **ATDV\_LASTERR()** is E\_MSSYSTEM, an operating system error has occurred.

For a complete list of MSI errors, see the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference*.

## Error Handling Example

```
/* error handling routine */
void do_error( devh, funcname )

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

```
main( )
{
    .
    .
    .
    /* call to Dialogic MSI/SC library function */
    if (ms_setevtmsk( devh, MSEV_SIGMSK, 0, DTA_SETMSK ) != 0) {
        do_error( devh, "ms_setevtmsk()" );
    }
    /* successful function call -
       continue processing ... */
    .
    .
    .
}
```

- Notes:**
1. Calls to **ms\_open()** return either a -1 or a nonzero device handle. Therefore, when issuing the **ms\_open()** function, check for a return of a -1, which indicates a system error.
  2. Calls to **ATMS\_TSSGBIT()** return the pointer AT\_FAILUREP when the function fails.

# Application Development Guidelines

## 4

This chapter contains suggestions for programmers who are designing and coding an application which uses the modular station interface (MSI) software. This chapter includes the following MSI general guidelines and task-specific programming guidelines:

- [General Guidelines](#) ..... 13
- [Initialization](#) ..... 14
- [Aborting](#) ..... 16
- [Standard Attribute Functions](#) ..... 16

## 4.1 General Guidelines

The following general guidelines for writing MSI-based applications are explained in this section:

- [Using Symbolic Defines](#)
- [Including Header Files](#)
- [Checking Return Codes](#)

**Note:** These guidelines are not a comprehensive guide to developing or debugging MSI applications.

### 4.1.1 Using Symbolic Defines

We do not guarantee the numerical values of defines will remain the same as new versions of a software package are released. In general, do not use a numerical value in your application when an equivalent symbolic define is available. Symbolic defines are found in the *dtilib.h* and *msilib.h* files.

### 4.1.2 Including Header Files

Various header files must be included in your application to test for error conditions, to use library functions from other Intel® Dialogic® products, or to perform event-management and standard-attribute functions. An example is shown below. See [Chapter 5, “Building Applications”](#) for details.

```
#include <windows.h>      /* For Windows applications only */
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
```

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

### 4.1.3 Checking Return Codes

Most Intel® Dialogic® library functions return a value of -1 if they fail (extended attribute functions return -1 or AT\_FAILUREP). Therefore, any call to a library function should check for a return value indicating an error. This can be done by using a format similar to the following:

```
/* call to MSI/SC library function */

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

Using this technique ensures that all errors resulting from a library call will be trapped and handled properly by the application. In many cases, you can check for a return value of other than zero (0), however, this should only be used where a nonzero value is returned when the function fails. For details, see [Chapter 3, “Error Handling”](#).

## 4.2 Initialization

Before an MSI-based application can perform any processing or access devices, it should initialize the appropriate hardware to correspond with the physical configuration of your system and set other parameters needed to support the application. Tasks that are performed as a part of initialization generally include:

- [Configuring DM3 Boards](#)
- [Configuring MSI/SC Boards](#)
- [Setting Event Mask on MSI Stations](#)
- [Initializing MSI Stations](#)
- [Terminating](#)

### 4.2.1 Configuring DM3 Boards

Intel® NetStructure™ High Density Station Interface (HDSI) boards and Intel® Dialogic® Integrated (DI) Station Interface boards are collectively called DM3 boards. Refer to the *DM3 Configuration File Reference* for more information on configuring HDSI and DI boards, including FCD file descriptions and configuration parameter listings.

### 4.2.2 Configuring MSI/SC Boards

#### 4.2.2.1 Setting Hardware Configuration on MSI/SC Boards

Use `ms_setbrdparm()` to set hardware configuration, debounce times, minimum and maximum hook flash times, ring cadence patterns, and ziptone. See the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference* for a description of the parameters.

MSI/SC board parameters which may be modified include:

<b>MSG_DBONTM</b>	Debounce on time
<b>MSG_DBOFFTM</b>	Debounce off time
<b>MSG_MINFLASH</b>	Minimum hook flash time
<b>MSG_MAXFLASH</b>	Maximum hook flash time
<b>MSG_PDRNGCAD</b>	Select the predefined ring cadence pattern
<b>MSG_RING</b>	Ring capability support information
<b>MSG_RNGCAD</b>	Ring cadence pattern
<b>MSG_UDRNGCAD</b>	Set user-defined ring cadence pattern
<b>MSG_ZIPENA</b>	Ziptone enable
<b>MSCB_ND</b>	Notify on add
<b>MSCB_ZIP</b>	Zip tone notification

#### 4.2.2.2 Changing Default Ring on MSI Boards

To change ring options on MSI boards, you must download a parameter file using the Intel® Dialogic Configuration Manager (DCM). To change the default active ring cycle, you must modify the *RING.PRM* parameter file. To set the frequency at which the individual stations ring, you must modify the *RINGFREQ.PRM* file. For more details, see the Intel DCM on-line help.

#### 4.2.2.3 Country-Specific Parameter Files for MSI Boards

A country-specific parameter file contains a unique set of coefficients so that specific gain and impedance values are met. When a country specific parameter file is not used, the firmware automatically defaults to U.S. specifications. Each parameter file contains the calculated data for the transmit/receive gain settings, impedance scaling settings, ring frequency settings and the programmable filters of a SLAC (Subscriber Line Audio Circuit) device.

### 4.2.3 Setting Event Mask on MSI Stations

Use the `ms_setevtmask()` function for setting and clearing the mask for on-hook transitions, off-hook transitions, and hook flash detection. The `MS_OFFHOOK`, `MS_ONHOOK`, and `MS_HOOKFLASH` equates are used for setting and clearing respective bits in the message mask.

## 4.2.4 Initializing MSI Stations

To set channel level parameters, the functions `ms_setbrdparm()` is used. See the *Modular Station Interface API for Linux and Windows Operating Systems Library Reference* for a description of the parameters.

## 4.2.5 Terminating

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

- Disabling events
- Resetting time slots
- Closing devices

The `ms_setevtmsk()` function can disable all currently enabled event notification masks.

**Note:** SRL Event Management functions such as `sr_dishdlr()`, (which disables an event handler), must be called before closing the device that is sending the handler event notifications.

## 4.3 Aborting

If you abort an MSI-based application by pressing the interrupt key, the system will terminate the current process but may leave devices in an unknown state. The next time you run your application, therefore, you may encounter errors.

To avoid errors of this type, your application should include an event handler that traps the interrupt key and performs the actions listed in [Section 4.2.5, “Terminating”](#), on page 16.

## 4.4 Standard Attribute Functions

The Standard Attribute functions in the Standard Runtime Library (SRL) return general device information such as the device name, or the last error that occurred on the device. The Standard Attribute functions and the MSI-specific information returned are listed below.

**Table 3. Standard Attribute Functions**

Standard Attribute Function	Information Returned for MSI
<code>ATDV_ERRMSGP()</code>	Pointer to string describing the error that occurred during the last function call on a device
<code>ATDV_IRQNUM()</code>	Interrupt number for a specified device
<code>ATDV_LASTERR()</code>	The error that occurred during the last function call on a specified device

Table 3. Standard Attribute Functions (Continued)

Standard Attribute Function	Information Returned for MSI
ATDV_NAMEP( )	Pointer to device name (MSIBb)
ATDV_SUBDEVS( )	Number of MSI subdevices. Note that the number of devices returned by this API is different for DI boards, HDSI boards, and MSI/SC boards.

See the *Standard Runtime Library API Library Reference* for details on these functions.





This chapter contains the following sections:

- [Include Files](#) ..... 19
- [Compiling and Linking](#) ..... 19

## 5.1 Include Files

Function prototypes and equates are defined in *dtilib.h* and *msilib.h*. Applications that use the MSI library functions for MSI support must include the following statements:

```
#include <windows.h>      /* For Windows applications only */
#include "srllib.h"
#include "dtilib.h"
#include "msilib.h"
```

Code that uses voice boards with the current version of the voice driver must include the following statements in the order shown:

```
#include <windows.h>      /* For Windows applications only */
#include "srllib.h"
#include "dxxlib.h"
#include "dtilib.h"
#include "msilib.h"
```

The current version of the voice driver requires that *srllib.h* precedes any other Intel® Dialogic® header file include statements.

## 5.2 Compiling and Linking

To compile and link your application, follow the syntax instructions for your version of the C Development Package.

Make sure when compiling or linking, the SRL library name is specified last. If your application includes MSI and Voice library functions, for instance, use either of the following two library orders on the command line for Linux:

```
-l dti -l dxxx -l srl
or
-l dxxx -l dti -l srl
```

Include the following libraries when using the MSI software on the Windows operating system:

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

Depending on your application, you may need to link with other libraries. Refer to the appropriate documentation for other Intel® Dialogic® products.

The modular station interface (MSI) software is supported on more than one type of hardware platform. This chapter describes the Intel® Dialogic® MSI/SC-Global Series boards, including the following topics:

- [MSI Hardware Overview](#) . . . . . 21
- [Typical Applications](#) . . . . . 21
- [Compatibility with other MSI boards](#) . . . . . 22
- [MSI Board Functional Description](#) . . . . . 23

## 6.1 MSI Hardware Overview

MSI/SC, MSI/SC-Global, and MSI/PCI-Global Series boards make up this hardware family.

- The MSI/SC product series consists of an ISA form factor baseboard that includes eight integrated station interfaces and allows up to two additional daughterboard modules, each with an additional eight station interfaces. The baseboard-only product is referred to as an MSI/80SC. A baseboard with one daughterboard module is referred to as an MSI/160SC (16 station interfaces), and a baseboard with two daughterboard modules is the MSI/240SC (24 station interfaces). The MSI/SC feature set is based on the Intel® Dialogic® MSI-C board; however, its hardware architecture is based on the D/41ESC board. This allows the MSI/SC to take advantage of the D/41ESC BLT (board locator technology) circuit, programmable interrupts, and shared RAM interface. In addition, the MSI/SC board provides a ringing option, making it capable of generating AC voltage sufficient to ring standard 2500 type telephones.
- The MSI/SC-Global boards support a ringer for each station attached to the board with two RENs (Ringer Equivalency Numbers) per station. MSI/SC-Global also features an international approvability.
- The MSI/PCI-Global product is a modular station interface board containing a digital signal processor (DSP) and the H.100 (CT Bus) connector for operation in SCbus mode. The MSI/PCI is also completely API- and feature-compatible with the MSI/SC ISA products. The MSI/80PCI-GBL consists of a baseboard and one SI/80PCI-GBL daughterboard module supporting eight station interface circuits and the MSI/160PCI-GBL is a baseboard product with two SI/80PCI-GBL daughterboard modules providing 16 station interface circuits.

## 6.2 Typical Applications

The MSI/SC boards and software allow an application program operating in the host PC to communicate between SCbus-compatible devices and analog station devices. Applications for the MSI/SC boards include:

- Inbound/outbound telemarketing

- Operator services such as billing automation, directory assistance, and intercept treatments
- Customer service
- Automatic call distribution (ACD)
- Dictation/transcription
- Local information services

Conferencing resources serve the SCbus with advanced features such as:

- Two-party to eight-party conferencing
- Up to 32 resources of total conferencing (four to 16 conferences)
- Conferences of any combination of stations and network channels
- Hidden training for smooth entry of new participants without disruptive training noise
- Monitoring an agent without disrupting the conversation
- Coaching feature to allow a supervisor to speak to an agent without the client hearing the supervisor; client can hear the agent at all times (no switching)
- Tone generation:
  - Zip tone indicates incoming call to agents using headsets
  - Notification tones when a party is added to or removed from a conference (as required by the law in many states)
- User programmable periodic notification tones to indicate units of time that expired during a call
- Programmable volume control for station devices
- Programmable ring cadences

## **6.3 Compatibility with other MSI boards**

The MSI board feature set is based on the Intel® Dialogic® MSI-C board and its hardware design enables the MSI to take advantage of the Board Locator Technology (BLT) circuit, programmable interrupts, and shared RAM interface.

Using the MSI, developers can build large call center configurations without the need for DMX boards or crossover cables. Other significant differences between the MSI and the MSI-C board include:

- Relay interlock for loop and ring voltages (only on the MSI/SC-R boards)
- Ability to ring 2500-type telephones
- On-board ring voltage generator (only on the MSI/SC-R boards)
- Ringing SLICs (Subscriber Line Interface Circuits) (MSI/SC-Global and MSI/PCI-Global boards)
- Use of the 1024 time slots available on the SCbus

Conference-specific differences between the MSI boards and the MSI-C board include:

- Pupil/coach feature

- Hidden training
- Extended connections

## 6.4 MSI Board Functional Description

The MSI baseboard and each daughterboard contain eight line interfaces and eight Coder/DECoders (CODECs). Each line interface provides loop-start current to one 2500 series equivalent station device.

The line interface also separates the inbound signal into an audio signal which is sent to the CODEC, and an on-hook/off-hook signal which is forwarded by the control processor to the application program.

The CODEC converts inbound audio to 8-bit PCM data and outbound PCM data to analog audio. The gain may be individually set for each station device.

A cross-point switch on the MSI board routes PCM data between the station devices, the DSP, and the SCbus.

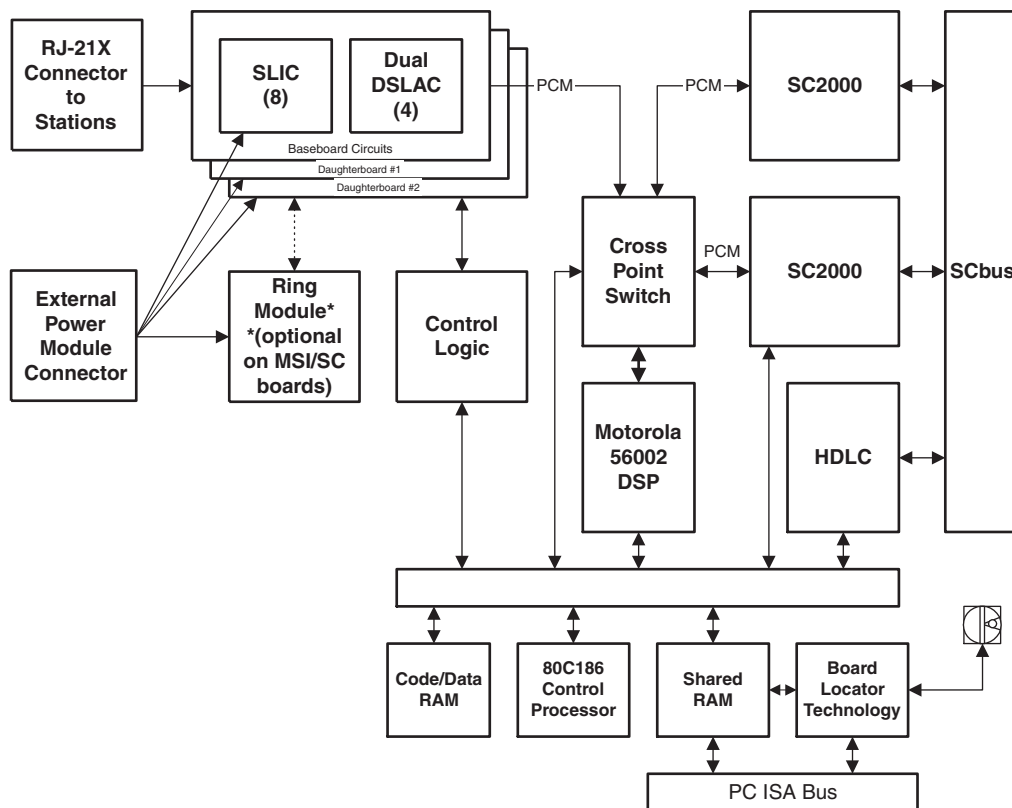
The MSI board conferencing feature allows conferences to be established between SCbus time slots and/or station interfaces.

The control microprocessor executes commands received from the host PC, and controls all operations of the MSI board. Communications between the control microprocessor and the host PC is accomplished via a shared RAM interface mechanism.

Operations demanding real time responses are interrupt driven. All MSI boards installed in the PC share the same interrupt line. When the system is initialized, the firmware needed to control all board operations is downloaded from the host PC to the on-board RAM. The downloadable firmware enables easy feature enhancements in the future.

The Board Locator Technology (BLT) circuit operates in conjunction with a rotary switch to determine and set non-conflicting slot and IRQ interrupt-level parameters. This feature eliminates the need to set jumpers or DIP switches.

Figure 1. MSI Board Block Diagram



**Note:** The optional ring module is only applicable to the MSI/SC-R boards. Ringing SLICs (Subscriber Line Interface Circuits) provide the ringing function for the MSI/SC-Global and MSI/PCI-Global boards.

## Glossary

---

**ACD:** Automatic call distributor. An automated (usually software-driven) system that connects incoming calls to agents based on a distribution algorithm. The system also gathers traffic analysis statistics, such as number of calls per hour, average time holding, and call length.

**agent:** An operator, transcriber, telemarketing or sales representative, or other employee. In this guide, agent refers to any person using an analog station device who can be connected to a caller or recorded message through the MSI board.

**A-Law:** A pulse code modulation (PCM) algorithm used in digitizing telephone audio signals in E1 areas.

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

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

**automatic call distributor:** See *ACD*.

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

**blocking mode:** When a telephone call cannot be completed, it is said that the call is “blocked”. In blocking mode, it is said that the caller is “receiving a busy”.

**channel:** 1. When used in reference to an Intel® Dialogic® digital expansion board, a data path, or the activity happening on that data path. 2. When used in reference to the CEPT telephony standard, one of 32 digital data streams (30 voice, 1 framing, 1 signaling) carried on the 2.048 MHz/sec E1 frame. (See *time slot*.) 3. When used in reference to a bus, an electrical circuit carrying control information and data.

**CT Bus:** Computer Telephony bus. A time division multiplexing communications bus that provides 4096 time slots for transmission of digital information between CT Bus products. See *TDM bus*.

**data structure:** C programming term for a data element consisting of fields, where each field may have a different type definition and length. The elements of a data structure usually share a common purpose or functionality, rather than being similar in size, type, etc.

**daughterboard:** In the context of this guide, the MSI daughterboard assembly. The daughterboard enables the MSI hardware to interface to analog station devices.

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

**digital:** Information represented as binary code.

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

**DM3:** Refers to Intel® Dialogic® mediastream processing architecture, which is open, layered, and flexible, encompassing hardware as well as software components. A whole set of products from Intel are built on DM3 architecture. Contrast with *Springware*, which is earlier-generation architecture.

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

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

**E1:** Another name given to the CEPT digital telephony format devised by the CCITT that carries data at the rate of 2.048 Mbps (DS-1 level). This service is available in Europe and some parts of Asia.

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

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

**flash:** A signal generated by a momentary on-hook condition. This signal is used by the voice hardware to alert a telephone switch that special instructions will follow. It usually initiates a call transfer. See also *hook state*.

**frequency shift keying (FSK):** A frequency modulation technique used to send digital data over voice band telephone lines.

**full-duplex:** Transmission in two directions simultaneously, or more technically, bi-directional, simultaneous two-way communications.

**hook flash:** See *flash*.

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

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

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

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

**Mu-Law:** The PCM coding and companding standard used in Japan and North America (T1 areas).

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





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

**on-hook:** Condition or state of a telephone line when a handset on the line is returned to its cradle (or an equivalent condition occurs). See also *hook state*.

**PCM:** Pulse Code Modulation. The most common method of encoding an analog voice signal into a digital bit stream. PCM refers to one technique of digitization. It does not refer to a universally accepted standard of digitizing voice.

**rfu:** Reserved for future use.

**SCbus (Signal Computing Bus):** A hard-wired connection between switch handlers on SCbus-based products. SCbus is a third generation TDM (time division multiplexed) resource sharing bus that allows information to be transmitted and received among resources over 1024 time slots. See *TDM Bus*.

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

**Signal Computing System Architecture:** See *SCSA*.

**Springware:** Software algorithms built into the downloadable firmware that provides the voice processing features available on all Intel® Dialogic® voice boards. The term “Springware” is also used to refer to a whole set of boards from Intel built using this architecture. Contrast with *DM3*, which is newer-generation architecture.

**SRL:** Standard Runtime Library containing Event Management functions, Standard Attribute functions, and data structures that are used by all Intel® NetStructure™ and Intel® Dialogic® devices.

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

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

**T1:** The digital telephony format used in North America and Japan that carries data at the rate of 1.544 Mbps (DS-1 level).

**TDM bus:** Time division multiplexing bus. A resource sharing bus such as the SCbus or CT Bus that allows information to be transmitted and received among resources over multiple data lines.

**TDM bus routing functions:** Used to set up communications between devices connected to the TDM bus. These functions enable an application to connect or disconnect (make or break) the receive (listen) channel of a device to or from a TDM bus time slot.

**time slot:** In a digital telephony environment, a normally continuous and individual communication (for example, someone speaking on a telephone) is (1) digitized, (2) broken up into pieces consisting of a fixed number of bits, (3)

combined with pieces of other individual communications in a regularly repeating, timed sequence (multiplexed), and (4) transmitted serially over a single telephone line. The process happens at such a fast rate that, once the pieces are sorted out and put back together again at the receiving end, the speech is normal and continuous. Each individual pieced-together communication is called a time slot.

**zip tone:** Short burst of a specified tone to an ACD agent headset usually indicating a call is being connected to the agent console.



# Index

---

## A

aborting an application 16

## C

check return codes 14

Coach 6

CODEC 23

Compatibility 22

compile and link 19

Conferencing  
    features 22

## E

Error handling 11

Event Management functions 9

Extended connection 6

## F

features  
    conferencing 22

Functional description 23

## G

General guidelines 13

## H

hardware configuration 14

## I

Include files 19

Initialization 14  
    set event mask 15

## M

Monitor 6

MSI  
    hardware configuration 14

MSI hardware  
    control microprocessor 23  
    cross-point switch 23

MSI/SC  
    conferencing 22  
    typical applications 21

## P

Pupil 6

## S

Standard Attribute functions 16

## T

terminating an application 16

Tone generation 22

