Dialogic
Making Innovation Thrive™

Dialogic® NaturalAccess™ SS7 Monitor Software Developer's Reference Manual

August 2009 64-0466-01

www.dialogic.com

Copyright and legal notices

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by 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 provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Dialogic, Dialogic Pro, Brooktrout, Diva, Cantata, SnowShore, Eicon, Eicon Networks, NMS Communications, NMS (stylized), Eiconcard, SIPcontrol, Diva ISDN, TruFax, Exnet, EXS, SwitchKit, N20, Making Innovation Thrive, Connecting to Growth, Video is the New Voice, Fusion, Vision, PacketMedia, NaturalAccess, NaturalCallControl, NaturalConference, NaturalFax and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and product mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Using the AMR-NB resource in connection with one or more Dialogic products mentioned herein does not grant the right to practice the AMR-NB standard. To seek a patent license agreement to practice the standard, contact the VoiceAge Corporation at http://www.voiceage.com/licensing.php.

Revision history

Revision	Release date	Notes
9000-62185-10	June 2002	LBG, Beta
9000-62185-11	August 2002	LBG
9000-62185-12	January 2004	SRR, SS7 Monitor 2.0 Beta
9000-62185-13	April 2004	SRR, SS7 Monitor 2.0
9000-62185-14	July 2006	LBZ, SS7 Monitor 2.3
64-0466-01	August 2009	LBG, SS7 Monitor
Last modified: August 3, 2009		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

Table Of Contents

1 Introduction

The *Dialogic® NaturalAccess™ SS7 Monitor Software Developer's Reference Manual* describes the interface between a host processor and a TX Series board, regardless of the operating system (Windows or UNIX) or device driver employed on the host.

Subsequent sections of this manual describe the general characteristics and operation of the Dialogic® NaturalAccess™ SS7 Monitor Software (NA SS7 Monitor Software) management functions and utility programs, as well as define the structure of the messages between NA SS7 Monitor Software and the application.

This manual targets programmers and system integrators who develop media server applications. This manual defines telephony terms where applicable, but assumes that the reader is familiar with basic telephony and Internet data communication concepts, switching, and the C programming language.

Note: The product(s) to which this document pertains is/are among those sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") in December 2008. Certain terminology relating to the product(s) has been changed, whereas other terminology has been retained for consistency and ease of reference. For the changed terminology relating to the product(s), below is a table indicating the "New Terminology" and the "Former Terminology". The respective terminologies can be equated to each other to the extent that either/both appear within this document.

Former terminology	Current terminology
NMS SS7	Dialogic® NaturalAccess™ Signaling Software
Natural Access	Dialogic® NaturalAccess™ Software
SS7 Monitor	Dialogic® NaturalAccess™ SS7 Monitor Software

2 SS7 Monitor Software overview

System requirements

To install and use NA SS7 Monitor Software to monitor network traffic, your installation must have the following components:

- A monitoring server with an installed TX Series board.

  Refer to the appropriate TX Series board installation manual for information on installing and configuring the TX Series board for NA SS7 Monitor Software mode.

- A protected monitoring connection point.

  The connection point must not distort the signal more than 20 dB when using any TX Series board.

- Cables to connect the TX Series board to the SS7 network at the protected monitoring connection point.

  Refer to the appropriate TX Series board installation manual for information on the types of cables required.

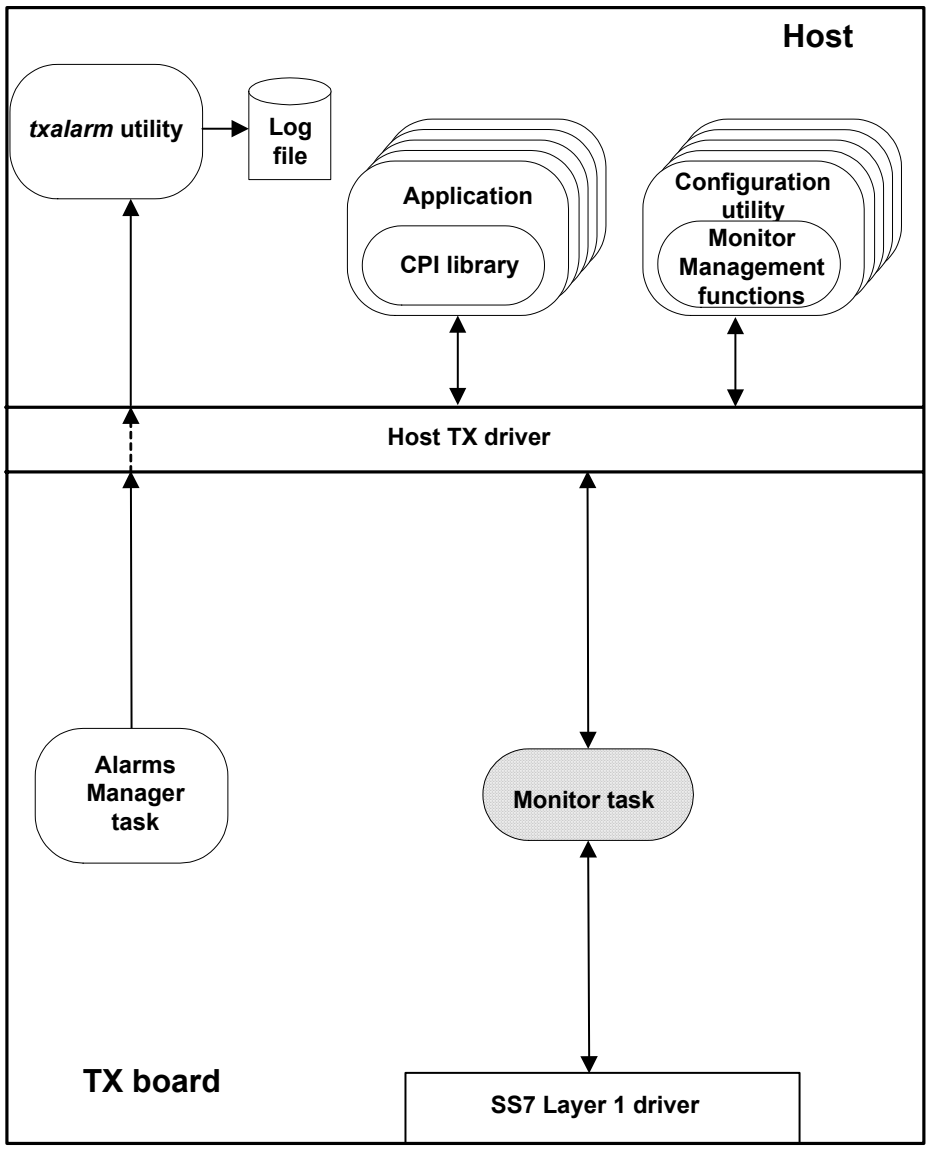
- Windows or UNIX operating system.
- Natural Access software development environment.

NA SS7 Monitor Software components

A typical NA SS7 Monitor Software implementation consists of the following components:

- SS7 monitor task running on a TX Series board.
- TX Alarms Manager task that collects unsolicited alarms (status changes) that the monitor task generates and forwards to the host for application-specific alarm processing.
- TX driver for the host operating system that provides low-level access to the TX board from the host processor.
- *txalarm* application on the host for displaying alarms from the monitor task, and optionally saving them to disk.
- NA SS7 Monitor Software interface that is composed of a set of messages passed between the application and the monitor task on the TX communications processor (CP). These messages enable the application to initialize and receive data.
- *moncfg* utility that downloads text file configurations to the monitor task. *moncfg* is provided in source and executable form.
- *monmgr* utility that enables you to manage the monitor task. *monmgr* is provided in source and executable form.
- *monapp* is a sample application. *monapp* is provided in source and executable form.
- NA SS7 Monitor Software management functions that provide a series of calls for managing the monitor task. The *moncfg* and *monmgr* utilities use these functions.

The following illustration shows the NA SS7 Monitor Software architecture in a typical system with separate host applications handling the data and control interface, system configuration, and system alarms:



NA SS7 Monitor Software interface

The NA SS7 Monitor Software interface consists of messages that govern communications between an application on the host and the monitor task on the TX board.

Messages perform the following tasks:

Message	Task
Bind Request	Establishes the host application as the NA SS7 Monitor Software user. Binding consists of a single request from the application to NA SS7 Monitor Software for which there is no response. Binds user application as sole user of NA SS7 Monitor Software.
Data Indication	Transfers data between the application and NA SS7 Monitor Software. Includes the only host-bound indication in this implementation: the data indication. This indication can include one or more SS7 signaling units (SUs) in the packet. Presents monitored message signaling units (MSUs) to the host.
Unbind Request	Stops the application from receiving data indications from the monitor task. At this time, all signaling units (SUs) are discarded. Unbinds user application as sole user of NA SS7 Monitor Software.

TX device drivers

TX device drivers are programs that implement the physical interface between the host and the TX board communications processor (CP). The TX device drivers establish a set of virtual channels. Channels enable multiple processes on both the host and the TX board CP to exchange messages without interfering with each other.

Processes and channels

A process (either on the host or on the CP) that receives messages registers with the TX device driver for a specific channel number. For a host-resident process, the channel registration can be implicit with the device open request, or it can be a separate operation. Channel numbers are assigned statically at compile time.

In the application, specify a channel number from the range of supported values. The TX board currently supports 256 channels (numbered 0 to 255). Channels 0 to 31 and 128 to 255 are reserved for use by Dialogic standard products. Channels 32 to 127 are available for application use. There is no built-in facility to dynamically learn the channel assigned to any particular function.

Only one process receives messages for a specific channel. Each message contains a header that the sender of the message supplies. The header identifies the destination board and channel, the source board and channel, and the length of the message.

Each process in an SS7 implementation, both on the CP and on the host, registers for the channel that corresponds to a layer (or application) entity ID. The monitor task registers for channel 0x5f. To send a message to the monitor task on TX board ***n***, set the destination channel to 0x5f, the destination board to ***n***, and the source channel and board to the sender's entity ID and board number. The host is board number 0 (zero). For more information, refer to *Entity and instance IDs* on page 13.

CPI Library interface

Applications and TX device drivers can use the CPI Library interface to communicate. To initialize the CPI Library interface, call cpi_init before using any other CPI calls. Issue a call to cpi_open, cpia_open, or cpix_open for each board the application wants to communicate with.

cpi_open creates a synchronous interface suitable for configuration but not recommended for high traffic load applications. cpia_open and cpix_open create asynchronous interfaces that are optimized for high traffic applications. For the synchronous interface, cpi_send and cpi_get_data are then called to send and retrieve data packets to and from the board. For the asynchronous interface, cpia_send and cpia_get_data are then called to send and retrieve data packets to and from the board. For more information, refer to the *Dialogic® TX Series SS7 Boards CPI Library Developer's Reference Manual*.

Entity and instance IDs

Each application must have a unique entity and instance ID to route messages among the various processes in the system.

Entity IDs are single byte values in the 0x00 to 0x3f range, assigned by the application developer. Allocate entity IDs as follows:

Range	Usage
0x00 - 0x1F	Reserved for system utilities, configuration utilities, and management utilities.
0x20 - 0x3F	Reserved for applications.

Instance IDs identify the processor on which an entity executes. The host is always processor 0 (zero). For all host-resident NA SS7 Monitor Software user applications, the host instance ID is always 0 (zero). All tasks on TX board number 1 receive an instance ID of 1, all tasks on TX board number 2 receive an instance ID of 2, and so on.

Byte ordering considerations

The following byte ordering conventions are for 16-bit and 32-bit parameters in the messages to and from the monitor task. The host application is responsible for converting between network and host byte order, if necessary, in both the send and receive directions.

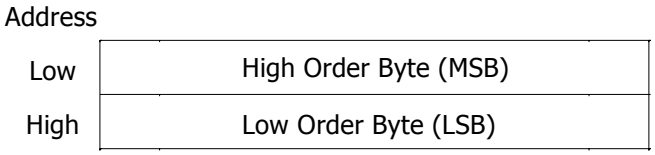
Note: The byte ordering convention used with NA SS7 Monitor Software is not the same convention used for short and long integers on Intel 80x86 machines.

The CPI library provides the following macros that enable applications to transparently handle host and TX board numeric representation issues:

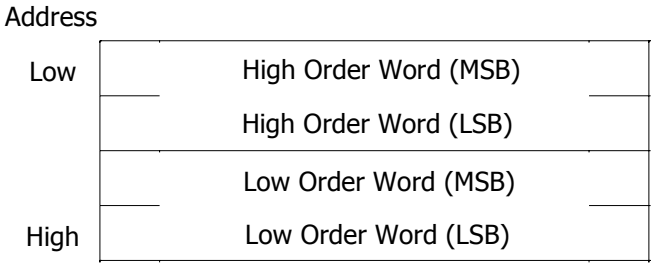
Macro	Task
cpi htocp l	Converts 32-bit value from the host format to the board format.
cpi htocp s	Converts 16-bit value from the host format to the board format.
cpi cptoh l	Converts 32-bit value from the board format to the host format.
cpi cptoh s	Converts 16-bit value from the board format to the host format.

The Message Length field is part of the dual-ported RAM (DPR) header interface between the host application and the host device driver, and must be passed in host native byte order, not the byte order shown in this topic. The Message Start and End offsets, on the other hand, are part of the interface between the host application and the NA SS7 Monitor Software process and must be passed in the network byte order shown in the following example:

16-bit fields



32-bit fields



The following nomenclature is used for data types:

Data type	Description
U8	Unsigned 8-bit quantity
S16	Signed 16-bit quantity
U32	Unsigned 32-bit quantity

3 Installing and configuring NA SS7 Monitor Software

Installation and configuration summary

Applications that monitor SS7 activity interact with the TX board through the CPI Library interface. The CPI library processes different types of messages. These messages are described in *NA SS7 Monitor Software interface* on page 12 and *Using the message reference* on page 55.

The SS7 Monitor task is configured by using management functions that are described in *Using the function reference* on page 25. A sample application that displays SS7 message data is described in *Sample monitoring application: monapp* on page 50.

The following table summarizes the steps for initially installing and configuring the NA SS7 Monitor Software:

Step	Description	For details, refer to...
1	Verify that monitor mode is enabled on the TX board and that the board is installed properly in the monitoring server. **Note:** TX 5000 Series boards do not require hardware enabling for monitor mode.	The installation manual for the specific TX Series board installed in the monitoring server
2	Connect the monitoring server to the network.	Connecting the monitor server to the network
3	Install NA SS7 Monitor Software.	*Installing Dialogic® NaturalAccess™ SS7 Monitor Software*
4	Configure the TDM.	*Dialogic® NaturalAccess™ Signaling Software Configuration Manual*
5	Configure the NA SS7 Monitor Software links.	*Configuring monitor links* on page 21
6	Download the NA SS7 Monitor Software and configuration files to the TX Series board.	*Downloading to the board* on page 23
7	Run the sample application to verify the installation.	*Running the sample application* on page 24

Connecting the monitor server to the network

This topic describes how to connect the NA SS7 Monitor Software depending on the installed TX board:

- Using TX 4000 boards
- Using TX 4000C boards
- Using TX 5000E boards

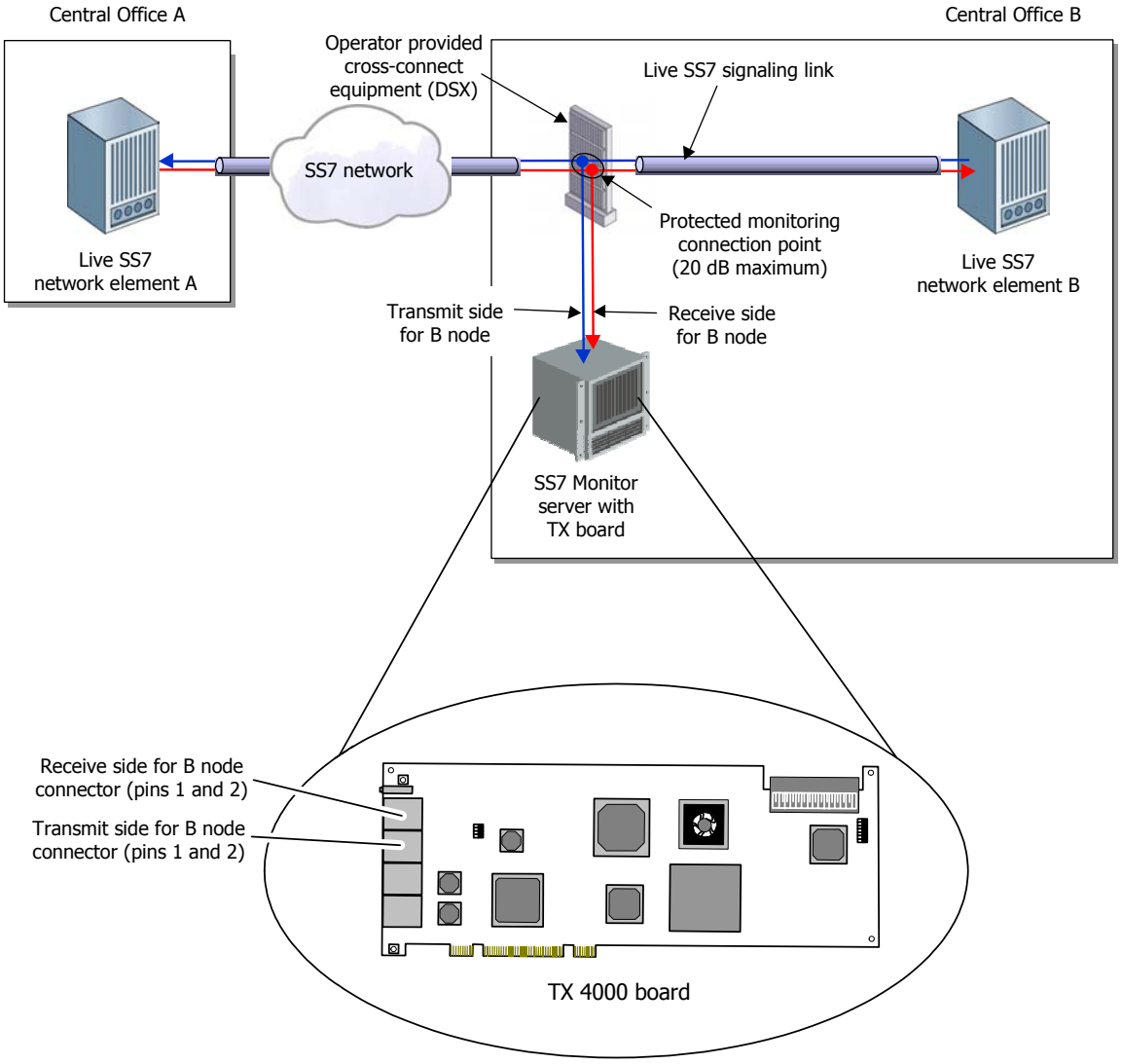
Using TX 4000 boards

If the monitor server has an installed TX 4000 board, ensure that DIP switch S3 on the TX board is set to enable monitor mode. For information on setting the DIP switch, refer to the *Dialogic® TX 4000 PCI SS7 Network Interface Board Installation Manual*.

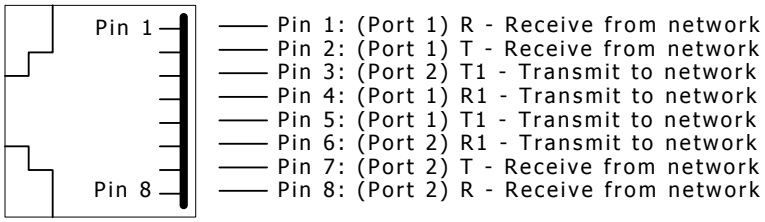
Connect the server to the SS7 network by attaching the cables to the 8-pin modular jacks on the TX 4000 end bracket. To connect the monitor server in this installation, use two separate cables: one cable for the transmit side of the B node connector and one cable for the receive side. Both cables use only pins 1 and 2.

Note: The protected monitoring connection point must not distort the signal more than 20 dB. This protection must be provided externally. If the loss is higher than 20 dB, the TX 4000 board may not recognize the signal. The 8-pin modular jack must receive a signal that has less than 20 dB of loss. This value includes the values from all cables and the protection circuitry.

The following illustration shows how to connect the monitor server with an installed TX 4000 board to the SS7 network and monitor SS7 traffic:



The following illustration shows the pinouts for 8-pin modular jacks on the TX 4000 board:



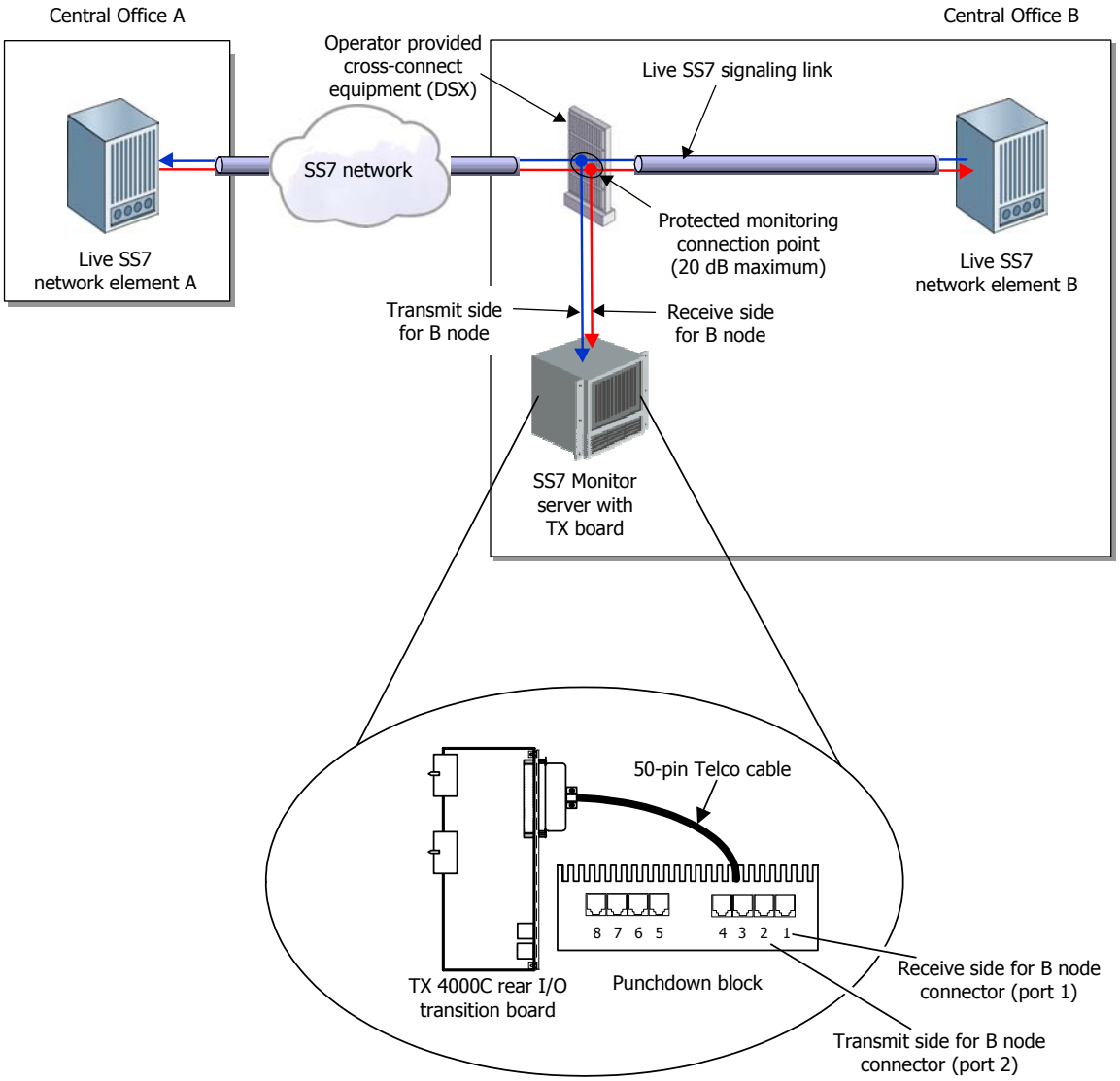
Using TX 4000C boards

If the monitor server has an installed TX 4000C board with an associated rear I/O transition board, ensure that the shunts are removed from the jumpers on the rear I/O transition board to enable monitor mode. For information on removing the shunts, refer to the *Dialogic® TX 4000C CompactPCI SS7 Network Interface Board Installation Manual*.

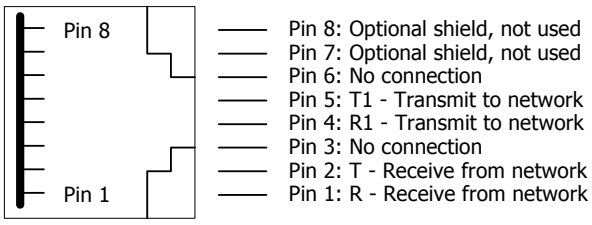
To connect the server to the SS7 network, attach a 50-pin Telco cable first to the RJ-48T trunk connector on the rear I/O transition board and then to a punchdown block. (If connecting to an E1 75 ohm trunk, connect the punchdown block to a 20-port balun.) Use two separate RJ-45 straight through cables from port 1 and port 2 on the punchdown block to connect to the network: one cable for the transmit side of the B node connector (port 2) and one cable for the receive side of the B node connector (port 1). For more information on establishing network connections, refer to the *Dialogic® TX 4000C CompactPCI SS7 Network Interface Board Installation Manual.*

Note: The protected monitoring connection point must not distort the signal more than 20 dB. This protection must be provided externally. If the loss is higher than 20 dB, the TX 4000C board may not recognize the signal. The RJ-48T connector must receive a signal that has less than 20 dB of loss. This value includes the values from all cables, the punchdown block, and the protection circuitry.

The following illustration shows how to connect the monitor server with an installed TX 4000C board to the SS7 network and monitor SS7 traffic:



The following illustration shows the pinouts of the eight RJ-48C connectors on the punchdown block:



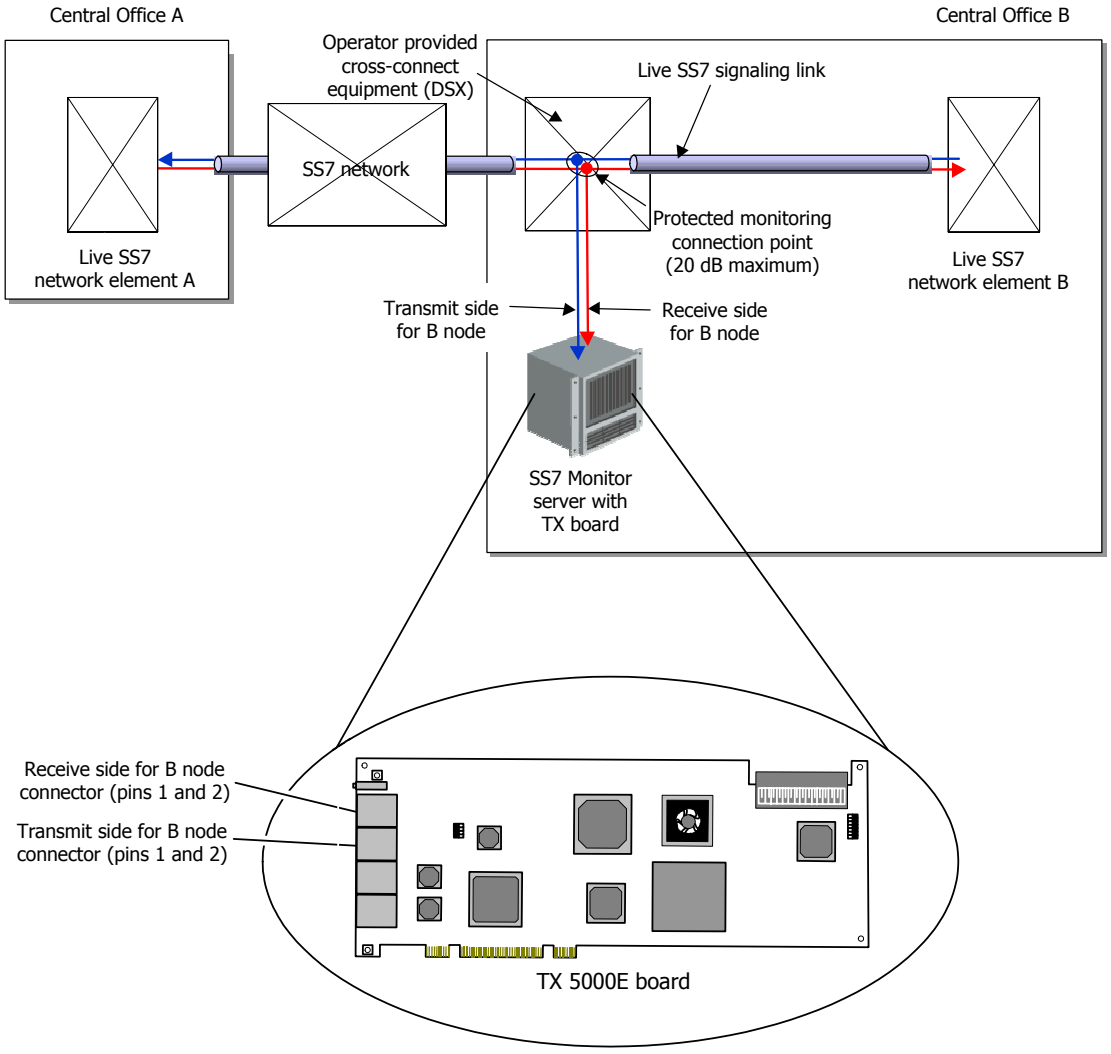
Using TX 5000E boards

There is no DIP switch to set for monitor mode on a TX 5000E board. Specify the monitoring function during configuration. For more information, refer to *Configuring TDM* on page 21.

Connect the server to the SS7 network by attaching the cables to the 8-pin modular jacks on the TX 5000E end bracket. To connect the monitor server in this installation, use two separate cables: one cable for the transmit side of the B node connector and one cable for the receive side. Both cables use only pins 1 and 2.

Note: The protected monitoring connection point must not distort the signal more than 20 dB. This protection must be provided externally. If the loss is higher than 20 dB, the TX 5000E board may not recognize the signal. The 8-pin modular jack must receive a signal that has less than 20 dB of loss. This value includes the values from all cables and the protection circuitry.

The following illustration shows how to connect the monitor server with an installed TX 5000E board to the SS7 network and monitor SS7 traffic:



Configuring TDM

The TX board sends and receives data on a T1, E1, or H.100/H.110 timeslot. A TDM configuration file is used to configure all T1/E1 trunks and to define ports. The configuration file identifies the timeslots to be monitored. TX Series boards are configured using the *txconfig* utility. A sample TDM configuration file is provided for TX boards *(txcfg1.txt*). The samples support a TX board using T1 lines.

To enable monitoring, set Loop Master to monitor. Do not set Loop Master to true/false on a per trunk basis to enable monitoring. The following example shows a *txcfg* file with Loop Master set to monitor:

```
clock net=1 -
#------------------------------------------------------------------------------
#     Trunk     Framing   Encoding   Buildout   Loop Master
t1cfg 1         esf       b8zs       0          monitor
t1cfg 2         esf       b8zs       0          monitor
t1cfg 3         esf       b8zs       0          monitor
t1cfg 4         esf       b8zs       0          monitor
#------------------------------------------------------------------------------
#    PortNum    L|H|E|T|J   Trunk   Channel     Speed
port 1          t1          1       0
port 2          t1          2       0
```

For more information, refer to the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual.*

Configuring monitor links

NA SS7 Monitor Software enables applications to configure the number of links, as well as the parameters associated with those links.

The following sample configuration file (*MONcp1.cfg*) is included with the NA SS7 Monitor Software and is located in *\Program Files\Dialogic\tx\config* for Windows and */opt/dialogic/tx/etc/* for UNIX:

```
# Monitoring general configuration section
RECV_LSSU           FALSE      # TRUE/FALSE - Send up LSSUs?
RECV_FISU           FALSE      # TRUE/FALSE - Send up FISUs?
MTP2HDR             FALSE      # TRUE/FALSE - Include MTP2 Header?
TIMER_TIND          5          # Max amount of time to wait before sending data inds
NUM_BUFS            500
SIZE_BUFS           1000

#
#---  Monitoring link parameters
LINK                0
LINK_TYPE           ANSI       # ANSI/ITU/ANSI_EXT/ITU_EXT
PORT_NUM            1
PORT_TYPE           TDM        # TDM
MAX_FRAME           272
NUM_BUFFERS         500
SIZE_BUFFERS        400
END

#
#---  Monitoring link parameters
LINK                1
LINK_TYPE           ANSI       # ANSI/ITU/ANSI_EXT/ITU_EXT
PORT_NUM            2
PORT_TYPE           TDM        # TDM
MAX_FRAME           272
NUM_BUFFERS         500
SIZE_BUFFERS        400
END
```

NA SS7 Monitor Software supports a configuration of 16 half-links on TX 4000/20 or TX 4000/20C boards, 32 half-links on TX 4000 or TX 4000C boards, and up to 128 half-links on TX 5000 Series boards. The maximum number of links allowed on a TX 5000 Series board is determined by the licensing options selected for the specific board. A half-link is the receive half of a full-duplex SS7 link. The application can pair two of these half-links into the two directions of one full-duplex link.

NA SS7 Monitor Software also supports high speed links (4 half-links). To support this feature, the TDM configuration file must be properly configured for high speed links by using an asterisk (*) for channel number. In the NA SS7 Monitor Software configuration, the application must set the link type to use extended sequence numbers using ANSI_EXT or ITU_EXT for the LINK_TYPE.

txcfg1.txt example for high speed links

```
#    PortNum    L|H|E|T|J   Trunk   Channel     Speed
port 1          t1          1       *
port 2          t1          2       *
```

MONcp1.cfg example for high speed links

```
#---  Monitoring link parameters
LINK               0
LINK_TYPE          ANSI_EXT   # ANSI/ITU/ANSI_EXT/ITU_EXT
PORT_NUM           1
PORT_TYPE          TDM        # TDM
MAX_FRAME          272
NUM_BUFFERS        500
SIZE_BUFFERS       400
END
```

Downloading to the board

Once the software and links are set up properly, download the software and the configuration to the TX board by using the *monload* batch file. The *monload* file accepts the board number to load. This loader file resets the TX board and downloads the necessary software to the TX board, including support tasks for alarms and debugging, and the monitor task. Finally, the configuration for the monitor task is downloaded.

The following sample code is a Windows version of *monload.bat*:

```
@echo off
REM ***************************************************************************
REM           TX Series COMMUNICATIONS PROCESSOR BOOT FILE (MONITOR)
REM
REM  Execute this file to perform the following:
REM     - Reset the TX board
REM     - Synchronize the on-board flash image with the installed software
REM     - Download TDM configuration
REM     - Download all TX-based tasks
REM     - Configure MONITOR
REM ***************************************************************************
if "%DIALOGIC_ROOT%"=="" set DIALOGIC_ROOT=\Program Files\Dialogic
if "%TX_ROOT%"=="" set TX_ROOT=%DIALOGIC_ROOT%\tx
REM ***************************************************************************
REM  Define all other script parameters
REM
if "%TXCONFIG%"=="" set TXCONFIG=%TX_ROOT%\config
if "%TXUTIL%"=="" set TXUTIL=%TX_ROOT%\bin
if "%TXCP%"=="" set TXCP=%TX_ROOT%\cp
REM ***************************************************************************
REM  Process arguments - Get the board number
REM
set BRD=1
if not "%1"=="" set BRD=%1
REM ***************************************************************************
REM  Clear driver statistics
REM
"%TXUTIL%\txstats" -b %BRD% -z -q
REM ***************************************************************************
REM  Get the model number (TX board type)
REM
"%TXUTIL%\cpmodel" -b %BRD%
if %errorlevel% == 4000 goto set4000
if %errorlevel% == 5000 goto set5000
if %errorlevel% == 5020 goto set5000
if %errorlevel% == 5500 goto set5000
echo ERROR! TX board number %BRD% not available.
goto end
REM ***************************************************************************
REM  Setup for TX 4000 Family
REM
:set4000
set TASKTYPE=elf
set TXKERNEL=cpk4000.fls
goto resetboard
REM ***************************************************************************
REM  Setup for TX 5000 Family
REM
:set5000
set TASKTYPE=elf
set TXKERNEL=cpk5000.elf
REM ***************************************************************************
REM  Reset the TX board
REM
:resetboard
REM  Reset TX board (and verify TX flash image in sync with installed software)
"%TXUTIL%\txflash" -s "%TXCP%\%TXKERNEL%" -b %BRD%
if %errorlevel% == 1 goto failedreset
```

```
REM ************************************************************************
REM  Enable the following to use txmon
REM
REM "%TXUTIL%\cplot" -c %BRD% -f "%TXCP%\txmon.%TASKTYPE%" -n txmon -p 19 -a
REM ************************************************************************
REM  Load TDM configuration
REM
"%TXUTIL%\txconfig" -b %BRD% -f "%TXCONFIG%\txcfg%BRD%.txt"
REM ************************************************************************
REM  Load the monitor layer
REM
"%TXUTIL%\cplot" -c %BRD% -f "%TXCP%\monitor.%TASKTYPE%" -n monitor -p 20 -a -s 12000
REM ************************************************************************
REM  Configure the monitor layer
REM
"%TXUTIL%\moncfg" -b %BRD% -f "%TXCONFIG%\MONcp%BRD%.cfg"
goto end
REM ************************************************************************
REM  Report reset error
REM
:failedreset
echo ERROR! Unable to reset TX board number %BRD%.
goto end
REM ************************************************************************
REM  Exit load script
:end
set TXUTIL=
set TXCP=
set TXCONFIG=
set BRD=
set TASKTYPE=
set TXKERNEL=
```

The *txalarm* utility is also included to help debug the configuration and user applications. For more information, refer to the *Dialogic® TX Series SS7 Boards TX Utilities Manual.*

```
Board loaded:
<06/10/2002 11:12:39> monitor   3      1 Monitor task initialized!
```

Running the sample application

When the board is loaded, you can start the sample application that comes with your system. The *monapp* sample application binds to the monitor task and a start monitoring message is sent to the monitor task. At this point, any data indications sent to the *monapp* application display on the screen. The following sample output is from *txalarm* at application bind time:

```
Application bound:
<06/10/2002 11:13:59> monitor   3   1 User application now bound (ent = 32)
```

When *monapp* is exited, a stop monitoring message is sent to the monitor task, and data indications are no longer sent to the host. The following sample output is from *txalarm* when the application unbinds:

```
Application unbound:
<06/10/2002 11:13:59> monitor   3      1 User application unbound
```

For more information, refer to *Sample monitoring application: monapp* on page 50.

4 NA SS7 Monitor Software function reference

Using the function reference

This section provides an alphabetical reference to the NA SS7 Monitor Software management functions. A prototype of each function is shown with the function description, details of all arguments, and return values. A typical function includes:

Prototype	The prototype is followed by a listing of the function's arguments. Dialogic data types include: • U8 8-bit unsigned • S16 16-bit signed • U32 32-bit unsigned • Bool 8-bit unsigned If a function argument is a data structure, the complete data structure is defined.
Return values	The return value for a function is either MON_SUCCESS or an error code. For asynchronous functions, a return value of MON_SUCCESS (zero) indicates the function was initiated; subsequent events indicate the status of the operation.

MonClearFilters

Clears a filter configuration in the monitor task that was previously set on one or all links.

Prototype

MON_STATUS **MonClearFilters** (U8 ***board***, S16 ***linkNo***)

Argument	Description
board	TX board number on which the desired monitor task resides.
linkNo	Link number from the configuration file. If this parameter is set to MON_ALL_LINKS, the filters for all links are cleared.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonSetFilter

MonGetFilters

Retrieves a filter configuration from the monitor task that has been set for one specific link.

Prototype

MON_STATUS **MonGetFilters** (U8 ***board***, MonFilterList ****pList***, S16 ***linkNo***)

Argument	Description
board	TX board number on which the desired monitor task resides.

plist	Pointer to a MonFilterList structure where the data link configuration values are filled:
	`typedef struct _MonFilterList`
	`{`
	`  U8 numActive;`
	`  U8 spare;`
	`  U16 spare2;`
	`  MonFilterDef filter[MON_MAXFILTERS];`
	`} MonFilterList;`

linkNo	Link number from the configuration file.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonSetFilter

MonGetGenCfg

Retrieves a general configuration from the monitor task.

Prototype

MON_STATUS **MonGetGenCfg** (U8 ***board***, MonGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pGenCfg	Pointer to a MonGenCfg structure where the data link configuration values are filled:
	`typedef struct _MonGenCfg`
	`{`
	`  U8  recvLssu;  /* Send up LSSUs? */`
	`  U8  recvFisu;  /* Send up FISUs? */`
	`  U8  mtp2Hdr;  /* Include the mtp2Hdr */`
	`  U8  spare;`
	`  U16  numBuffers;  /* Number of buffers in limited pool */`
	`  U16  sizeBuffers;  /* Size of each buffer in limited pool */`
	`  TimerCfg tInd;  /* Maximum time to wait before sending up a data indication */`
	`} MonGenCfg;`

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonInitGenCfg

MonGetGenStats

Retrieves and potentially resets general statistical information from the monitor task.

Prototype

MON_STATUS **MonGetGenStats** (U8 ***board***, MonGenStats ****pStats***, Bool ***bReset***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pStats	Pointer to a MonGenStats structure where the data link configuration values are filled:
	`typedef struct _MonGenStats`
	`{`
	`  S16  action;  /* Action to take (ZEROSTS or NOZEROSTS) */`
	`  S16  spare1;  /* Alignment */`
	`  U32  numAppCongDrop;  /* Number of messages dropped due to congestion`
	`  * (limited pool up to host is full) */`
	`} MonGenStats;`

bReset	Flag that indicates whether or not to set statistics to zero after retrieval.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

MonGetLinkCfg

Retrieves the configuration of one link.

Prototype

MON_STATUS **MonGetLinkCfg** (U8 ***board***, MonLinkCfg ****pLinkCfg***, S16 ***linkNo***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pLinkCfg	Pointer to a MonLinkCfg structure where the data link configuration values are filled:
	`typedef struct`
	`{`
	`  U16  linkNum;  /* Link Number */`
	`  U16  linkType;  /* ANSI/ITU */`
	`#define MON_LINK_ANSI  0`
	`#define MON_LINK_ITU  1`
	`#define MON_LINK_ANSI_EXT  2`
	`#define MON_LINK_ITU_EXT  3`
	`  U16  portNum;  /* Port number (from tdmcfg or txconfig) */`
	`  U16  portType;  /* Port type (tdm v. serial) */`
	`#define MON_PTYPE_SERIAL  0`
	`#define MON_PTYPE_TDM  4`
	`  U16  dataEnc;  /* Data encoding (NRZ/NRZI) */`
	`#define DAT_NRZ  0  /* data format is NRZ */`
	`#define DAT_NRZI  1  /* data format is NRZI */`
	`  U16  maxFrameLen;  /* Maximum SS7 frame length */`
	`  U16  numBuffers;  /* Number of buffers in limited pool */`
	`  U16  sizeBuffers;  /* Size of each buffer in limited pool */`
	`} MonLinkCfg;`

linkNo	Retrieves the configuration information of this specific link number.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonInitLinkCfg

MonGetLinkStats

Retrieves the link statistics for one link.

Prototype

MON_STATUS **MonGetLinkStats** (U8 ***board***, MonLinkStats ****pstats***, S16 ***linkNo***, Bool ***bReset***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pstats	Pointer to MonLinkStats structure:
	`typedef struct _MonLinkStats /* MTP level 2 link statistics */`
	`{`
	`S16 action; /* Action to take (ZEROSTS or NOZEROSTS) */`
	`S16 spare1; /* Alignment */`
	`U32 numMsuRx; /* Number of MSUs received */`
	`U32 numLssuRx; /* Number of LSSUs sent up to host */`
	`U32 numFisuRx; /* Number of FISUs sent up to host */`
	`U32 numRxFiltered; /* Number of messages dropped due to filter configuration */`
	`U32 numSUERMFail; /* Number of times the SUERM has failed on this link */`
	`} MonLinkStats;`

linkNo	Retrieve the statistics of this specific link number.
bReset	Flag that indicates whether or not to set statistics to zero after retrieval.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

MonInitGenCfg

Builds a basic configuration buffer to be passed to **MonSetGenCfg**.

Prototype

MON_STATUS **MonInitGenCfg** (U8 ***board***, MonGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired monitor task resides.
pGenCfg	Pointer to the MonGenCfg structure to be initialized. For more information, refer to Details.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

This function initializes a MonGenCfg structure to all of its default values:

```
typedef struct _MonGenCfg
{
    U8      recvLssu;        /* Send up LSSUs? */
    U8      recvFisu;        /* Send up FISUs? */
    U8      mtp2Hdr;         /* Include the mtp2Hdr? */
    U8      spare;
    U16     numBuffers;      /* Number of buffers in limited pool */
    U16     sizeBuffers;     /* Size of each buffer in limited pool */
    TimerCfg tInd;           /* Maximum time to wait before sending up a data indication
*/
} MonGenCfg;
```

The following table lists the field defaults and possible values:

Field	Default	Range	Description
recvLssu	FALSE	TRUE|FALSE	If TRUE, the LSSUs are passed to the application. If FALSE, they are dropped.
recvFisu	FALSE	TRUE|FALSE	If TRUE, the FSUs are passed to the application. If FALSE, they are dropped.
mtp2Hdr	FALSE	TRUE|FALSE	If TRUE, the MTP2 header on the front of the message is left intact. If FALSE, the monitor task strips it off.
numBuffers	500	50 - 1000	Number of buffers used on the board for sending up data indications.
sizeBuffers	1000	300 - 1900	Size of the buffers used for sending up data indications.
tInd	5	0 = send immediately 0 - 60	The maximum amount of time (in seconds) to wait before sending an SS7 message.

MonInitLinkCfg

Builds a link configuration buffer to be passed to **MonSetLinkCfg**.

Prototype

MON_STATUS **MonInitLinkCFg** (U8 ***board***, MonLinkCfg ****pLinkCfg***, S16 ***linkNo***, U8 ***linkType***)

Argument	Description
board	TX board number on which the desired monitor task resides.
pLinkCfg	Pointer to the MonLinkCfg structure to initialize. For more information, refer to Details.
linkNo	Specifies the link number for which to set this structure.
linkType	CCITT, ITU, or ANSI.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

This function initializes a MonLinkCfg structure to all of its default values:

```
typedef struct
{
    U16     linkNum;              /* Link Number */
    U16     linkType;             /* ANSI/ITU */
#define MON_LINK_ANSI      0
#define MON_LINK_ITU       1
#define MON_LINK_ANSI_EXT  2
#define MON_LINK_ITU_EXT   3
    U16     portNum;              /* Port number (from tdmcfg or txconfig) */
    U16     portType;             /* Port type (Serial no longer supported) */
#define MON_PTYPE_TDM      4
    U16     dataEnc;              /* Data encoding (NRZ/NRZI) */
#define DAT_NRZ            0     /* data format is NRZ */
#define DAT_NRZI           1     /* data format is NRZI */
    U16     maxFrameLen;          /* Maximum SS7 frame length */
    U16     numBuffers;           /* Number of buffers in limited pool */
    U16     sizeBuffers;          /* Size of each buffer in limited pool */
} MonLinkCfg;
```

The following table lists the field defaults and possible values:

Field	Default	Range	Description
linkNum	linkNo parameter	0 - 15 (TX 4000/20, TX 4000/20C) 0 - 31 (TX 4000, TX 4000C) 0 - 127 (TX 5000 Series)	Link index number to use to refer to this link.
linkType	linkType parameter		Link type can be ANSI or ITU with or without the support for extended sequence numbers. This is used to help the SS7 Monitor task decode parts of the SS7 messages.
portNum	linkNo parameter + 1	1 - 16 (TX 4000/20, TX 4000/20C) 1 - 32 (TX 4000, TX 4000C) 1 - 128 (TX 5000 Series)	Port number from the TDM configuration.
portType	MON_PTYPE_TDM		Only allowed port type is TDM for T1 or E1.
dataEnc	DAT_NRZ, DAT_NRZI|DAT_NRZ		Not applicable.
maxFrameLen	272	0 - 500	Maximum frame length for a received SS7 message.
numBuffers	500	50 - 1000	Number of buffers to use on the board for receiving SS7 messages from the network.
sizeBuffers	400	300 - 500	Size of the buffers used for receiving SS7 messages from the network.

MonMgmtInit

Initializes internal structures and opens communication with the monitor process on the TX board.

Prototype

MON_STATUS **MonMgmtInit** (U8 ***board***, U8 ***srcEnt***, U8 ***srcInst***)

Argument	Description
board	TX board number on which the desired monitor task resides.
srcEnt	Source entity ID.
srcInst	Source instance ID.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

This function must be called before any other management functions can be used. Source entity must be unique for each application accessing the monitor task.

Note: An application that opens both the management and data functions must use different entity IDs for **MonMgmtInit** and data bind request calls.

MonMgmtTerm

Terminates the dual-port RAM channel binding, specified in **MonMgmtInit**, for this application.

Prototype

MON_STATUS **MonMgmtTerm** (U8 ***board***)

Argument	Description
board	TX board number that terminates communication.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

This function frees up resources when an application terminates or finishes communication with the monitor task.

MonSetFilter

Passes a filter configuration to the monitor task.

Prototype

MON_STATUS **MonSetFilter** (U8 ***board***, MonFilterDef ****pfilter***, S16 ***linkNo***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pfilter	Pointer to a new filter definition:
	`typedef struct _MonFilterDef`
	`{`
	`  U8 type;`
	`#define MON_FILTER_SIO 1  /* Filter types */`
	`#define MON_FILTER_DPC 2`
	`#define MON_FILTER_OPC 3`
	`  U8 spare;`
	`  U16 spare2;`
	`  U32 value;`
	`} MonFilterDef;`

linkNo	Link number from the configuration file. If this parameter is set to MON_ALL_LINKS, all filters are set for all links, not for just one specific link.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

This function sets destination point code (DPC), originating point code (OPC), or service information octet (SIO) filters on one or all links.

A filter definition is defined by two values: the type and the value. The type is set to either MON_FILTER_SIO, MON_FILTER_DPC, or MON_FILTER_OPC. When a message comes in, it is compared to the defined filters on the link it came in on. If the message does not match any of the filters, it is dropped. There is a maximum of three filter definitions per link. You cannot have more than one filter of any type active at a time. For example, you cannot have two SIO filters active at a time.

For OPC and DPC filters, the value field is compared to the DPC or OPC in the MTP3 routing label. For the SIO filter, the service indicator octet is retrieved from the received message and compared to the filter value. The service indicator determines for which SS7 upper layer the message is destined (for example, ISUP, SCCP, or TUP).

MonSetGenCfg

Passes a general configuration to the monitor task.

Prototype

MON_STATUS **MonSetGenCfg** (U8 ***board***, MonGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired monitor task resides.

pGenCfg	Pointer to the MonGenCfg structure to send to the board:
	`typedef struct _MonGenCfg`
	`{`
	`  U8  recvLssu;  /* Send up LSSUs? */`
	`  U8  recvFisu;  /* Send up FISUs? */`
	`  U8  mtp2Hdr;  /* Include the mtp2Hdr */`
	`  U8  spare;`
	`  U16  numBuffers;  /* Number of buffers in limited pool */`
	`  U16  sizeBuffers;  /* Size of each buffer in limited pool */`
	`  TimerCfg tInd;  /* Maximum time to wait before sending up a data indication */`
	`} MonGenCfg;`

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonInitGenCfg

MonSetLinkCfg

Passes a link configuration structure to the monitor task.

Prototype

MON_STATUS **MonSetLinkCfg** (U8 ***board***, MonLinkCfg ****plinkCfg***)

Argument	Description
board	TX board number on which the desired monitor task resides.

plinkCfg	Pointer to the MonLinkCfg structure to send to the board:
	`typedef struct`
	`{`
	`  U16  linkNum;  /* Link Number */`
	`  U16  linkType;  /* ANSI/ITU */`
	`#define MON_LINK_ANSI  0`
	`#define MON_LINK_ITU  1`
	`#define MON_LINK_ANSI_EXT  2`
	`#define MON_LINK_ITU_EXT  3`
	`  U16  portNum;  /* Port number (from tdmcfg or txconfig) */`
	`  U16  portType;  /* Port type (tdm v. serial) */`
	`#define MON_PTYPE_SERIAL  0`
	`#define MON_PTYPE_TDM  4`
	`  U16  dataEnc;  /* Data encoding (NRZ/NRZI) */`
	`#define DAT_NRZ  0  /* data format is NRZ */`
	`#define DAT_NRZI  1  /* data format is NRZI */`
	`  U16  maxFrameLen;  /* Maximum SS7 frame length */`
	`  U16  numBuffers;  /* Number of buffers in limited pool */`
	`  U16  sizeBuffers;  /* Size of each buffer in limited pool */`
	`} MonLinkCfg;`

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

See also

MonInitLinkCfg

MonSetTrace

Sets the debug output level for the monitor task.

Prototype

MON_STATUS **MonSetTrace** (U8 ***board***, U16 ***traceLevel***)

Argument	Description
board	TX board number on which the desired monitor task resides.

traceLevel	New trace level that can be set to the following levels:
	`#define MON_TRACE_ERROR 0x01`
	`#define MON_TRACE_WARNING 0x02`
	`#define MON_TRACE_DEBUG 0x10`
	These values can be combined using the OR operator. The default trace level is MON_TRACE_ERROR.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

MonStartLink

Starts the monitoring of a specific link.

Prototype

MON_STATUS **MonStartLink** (U8 ***board***, S16 ***linkNum***)

Argument	Description
board	TX board number on which the desired monitor task resides.
linkNum	Link number for which this applies. If this parameter is set to MON_ALL_LINKS, all links are monitored, not just one specific link.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

When an application binds to the monitor task, the task begins monitoring all links. The link is enabled and traffic received on it is passed to the application. To stop monitoring a specific link, call **MonStopLink**. To restart monitoring a link, use **MonStartLink**.

MonStopLink

Stops the monitoring of a specific link.

Prototype

MON_STATUS **MonStopLink** (U8 ***board***, S16 ***linkNum***)

Argument	Description
board	TX board number on which the desired monitor task resides.
linkNum	Link number for which this applies. If this parameter is set to MON_ALL_LINKS, monitoring is stopped for all links.

Return values

Return value	Description
MON_BOARD	Board number out of range.
MON_DRIVER	CPI driver returned an error.
MON_HANDLE	Board not previously initialized.
MON_LINKTYPE	Invalid link or switch type.
MON_PARAM	Invalid parameter.
MON_RESPONSE	Wrong response from board.
MON_SUCCESS	
MON_TIMEOUT	No response from board.

Details

When an application binds to the monitor task, the task begins monitoring all links. The link is enabled and traffic received on it is passed up to the application. To stop monitoring a specific link, call **MonStopLink**. To restart monitoring a link, use **MonStartLink**.

5 Utility programs

Summary of the utility programs

SS7 Monitor provides the following utility programs:

Program	Description
moncfg	Downloads the NA SS7 Monitor Software configuration to the TX board at boot time.
monmgr	Monitors and manages the status of the NA SS7 Monitor Software.
monapp	Partially decodes SS7 messages.

Monitor Configuration utility: moncfg

Scans the NA SS7 Monitor Software configuration text file and downloads the configuration to the monitor task on the TX board.

Usage

```
moncfg options
```

Prerequisites

- A computer with a TX board installed
- Windows or UNIX
- Natural Access

Procedure

Follow this procedure to run *moncfg*:

Step	Action
1	From the command line prompt, navigate to the *\Program Files\Dialogic\tx\bin* directory under Windows or the */opt/dialogic/tx/bin* directory under UNIX.
2	Enter the following command: `moncfg options` where ***options*** include: **Option** / **Description** -b ***boardnum*** — Board number to which the NA SS7 Monitor Software configuration is downloaded. Default = 1. -f ***filename*** — Name and location of the NA SS7 Monitor Software configuration file to be downloaded. The Monitor Configuration utility scans the information in the ASCII file (specified with the -f option) and downloads the information to the monitor task on the TX board.

The NA SS7 Monitor Software configuration file is a text file with the following format:

```
 # Monitoring general configuration section
RECV_LSSU          FALSE      # TRUE/FALSE - Send up LSSUs?
RECV_FISU          FALSE      # TRUE/FALSE - Send up FISUs?
MTP2HDR            FALSE      # TRUE/FALSE - Include MTP2 Header?
TIMER_TIND         5  # Max amount of time to wait before sending data inds
NUM_BUFS           500
SIZE_BUFS          1000

#
#---  Monitoring link parameters
LINK               0
LINK_TYPE          ANSI       # ANSI/ITU/ANSI_EXT/ITU_EXT
PORT_NUM           1
PORT_TYPE          TDM        # TDM
MAX_FRAME          272
NUM_BUFFERS        500
SIZE_BUFFERS       400
END

#
#---  Monitoring link parameters
LINK               1
LINK_TYPE          ANSI       # ANSI/ITU/ANSI_EXT/ITU_EXT
PORT_NUM           2
PORT_TYPE          TDM        # TDM
MAX_FRAME          272
NUM_BUFFERS        500
SIZE_BUFFERS       400
END
```

The first link defined in the NA SS7 Monitor Software configuration file is always half link 0 (zero), the second is always link 1, and so on.

The following table lists general configuration parameters and values:

Parameter	Value	Default	Description
RECV_FISU	TRUE|FALSE	FALSE	When TRUE, FISUs are forwarded to the host application.
RECV_LSSU	TRUE|FALSE	FALSE	When TRUE, LSSUs are forwarded to the host application.
MTP2HDR	TRUE|FALSE	FALSE	When TRUE, MTP2 headers are forwarded to the host application.
NUM_BUFS	50 - 1000	500	Count of the buffers to use to send data indications to the host.
SIZE_BUFS	300 - 1900	1000	Size of the buffers to use to send data indications to the host.
TIMER_TIND	0 - 60	5	Number of seconds to wait until forwarding a data indication to the host. If zero, send one packet per buffer immediately.

The following table lists the monitoring link parameters and values. All parameters are optional (default values are shown) except the LINK port assignment and the terminating END statement.

Parameter	Value	Default	Description
DATA_ENC	NRZ or NRZI	NRZ	Data encoding (NRZ or NRZ inverted).
LINK	T1-T128	None	Link index.
LINK_TYPE	ANSI, ITU, ANSI_EXT, or ITU_EXT	ANSI	Protocol variant. The _EXT values indicate high speed links using extended sequence numbers.

Parameter	Value	Default	Description
MAX_FRAME	64 - 1024	272	Maximum frame length for MSU.
NUM_BUFFERS	50 - 1000	500	Number of buffers in pool. (Dialogic does not recommend changing this parameter.)
PORT_NUM	1 - 16 (TX 4000/20 and TX 4000/20C) 1 - 32 (TX 4000 and TX 4000C) 1 – 128 (TX 5000 Series)	None	Maps to TDM port parameter. **Note:** The maximum port number for TX 5000 Series boards is determined by licensing options with a maximum of 128 ports available.
PORT_TYPE	TDM	TDM	Port type. Serial ports are no longer supported.
SIZE_BUFFER	300 - 500	400	Size of buffer in pool (Dialogic does not recommend changing this parameter.)

Monitor Manager program: monmgr

After downloading the Dialogic configuration to the TX board with *moncfg*, run the Monitor Manager (*monmgr*) program to monitor the status of the monitor task. The Monitor Manager provides a command line interface that can be used to view statistics.

Usage

```
monmgr -b boardnum
```

Prerequisites

- A computer with a TX board installed
- Windows or UNIX
- Natural Access

Procedure

Follow this procedure to run *monmgr*:

Step	Action
1	From the command line prompt, navigate to the *\Program Files\Dialogic\tx\bin* directory in Windows or the */opt/dialogic/tx/bin* directory in UNIX.
2	Enter the following command: `monmgr -b boardnum` where ***boardnum*** is the number of the TX board to open.

The *monmgr* program supports the following commands:

Command	Description
LINK ***link number*** START|STOP	Starts or stops monitoring a specific link.
TRACE trace_level	Changes the debug level of information sent to *txalarm.*
GET GEN	Displays general configuration.
GET LINK ***link number*** | *	Displays configuration of a specific link or all defined links.
GET FILTERS ***link number*** | *	Displays the filter settings of a specific link or all defined links.
STATS GEN	Displays general statistics.
STATS LINK ***link number*** [RESET]	Displays a link's statistics.
FILTER ***link number*** DPC|OPC|SIO|CLEAR ***filter value***	Sets the filter for a specific link.
? [***command name***]	Displays Help for all commands or for a specific command.
BOARD ***board number***	Switches to a new board.
Q	Quits *monmgr*.

Sample monitoring application: monapp

Partially decodes SS7 messages, but does not support the decoding of MTP2 extended sequence numbers (used for high speed links).

Usage

```
monapp -b boardnum
```

Prerequisites

- A computer with a TX board installed
- Windows or UNIX
- Natural Access

Procedures

Follow this procedure to run *monapp*:

Step	Action
1	From the command line prompt, navigate to the *\Program Files\Dialogic\tx\bin* directory under Windows or the */opt/dialogic/tx/bin* directory under UNIX.
2	Enter the following command: `monapp -b boardnum` where ***boardnum*** is the number of the TX board to open. At startup, the sample application attempts to bind to the monitor task and starts printing to the screen whenever a data indication is received.

Follow this procedure to enter command mode:

Step	Action
1	Press **Enter** to display the following prompt: `Command?>`
2	Enter one of the following commands from this prompt:

Command	Description
Q	Quit *monapp*
I	Make tracing to screen invisible
V	Make tracing to screen visible
DATA pattern	set a specific pattern to match on
STATS ISUP|GEN [***link_number***]	Get statistics for one link or all links
RESET [***link_number***]	Reset statistics for one link or all links
MTP2 [0|1]	0 = No MTP2 header in the data 1 = MTP2 header in the data
SHOW [0|1]	0 = Do not display data 1 = Display data
ISUP [0|1]	0 = No ISUP messages 1 = Decode ISUP message types
?	Display *monapp* command help

SS7 message header

The *monapp* application partially decodes SS7 messages. *monapp* decodes the following information from the SS7 message header:

- SS7 message length
- Link number
- Number of spares
- Time stamp of each SS7 message.

For each SS7 message, a generic trace is displayed. For example:

```
=======================================================
Length: --          Link #:--          NumSpare: --
Number seconds: -- msecs: --
```

The following table describes the SS7 message data:

If the message length is...	Then...
Less than 3 bytes	The trace, Bad message length --, displays.
Equal to 3 bytes	A FISU is decoded and the following trace displays: FISU: BSN BSN:BIB FSN FSN:FIB
Equal to 4 or 5 bytes	A LSSU is decoded. One of the following traces displays, depending on the LSSU type: • SIO: BSN BSN:BIB FSN FSN:FIB • SIN: BSN BSN:BIB FSN FSN:FIB • SIE: BSN BSN:BIB FSN FSN:FIB • SIOS: BSN BSN:BIB FSN FSN:FIB • SIPO: BSN BSN:BIB FSN FSN:FIB • If some other type of LSSU:LSSU: BSN BSN:BIB FSN FSN:FIB
Greater than 5 bytes	MSU and generic trace display. MSU: BSN BSN:BIB FSN FSN:FIB Additional traces display: • To specify the type of the upper protocol of the MSU (such as MTP3, TCAP, SCCP, and ISUP) Protocol: ***protocol_name*** • In case of ISUP, additional traces are available to specify the ISUP message type (such as ACM, ANM, REL, and RLC) Protocol: ISUP Message: ***message_name***

The following table describes the mnemonics that the SS7 message data displays:

Mnemonic	Description
ACM	Address complete message
ANM	Answer message
BIB	Backward indicator bit
BSN	Backward sequence number
FIB	Forward indicator bit
FISU	Fill in signal unit
FSN	Forward sequence number
LSSU	Link status signal unit
MSU	Message signal unit
REL	Release
RLC	Release complete
SIE	Status indicator emergency alignment
SIN	Status indicator normal alignment
SIO	Status indicator out of alignment
SIOS	Status indicator out of service
SIPO	Status indicator processor outage

6 NA SS7 Monitor Software message reference

Using the message reference

This section specifies the layout of messages between the monitor task and the application. The message layouts differ depending on whether the message is read or written directly from or to the TX device driver or whether it is read from the IPC queue.

The following topics define the structure of each message beginning with the NA SS7 Monitor message type and including all parameters of the message:

- Bind Request
- Unbind Request
- Data Indication

Bind Request

Establishes communication between the application and the monitor task. An application must send this message only once at initialization time.

Direction

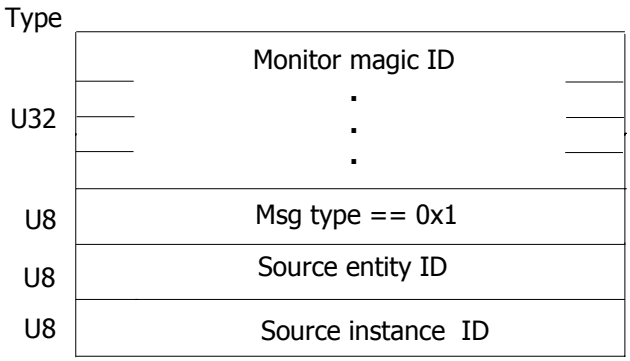
Application -> Monitor

Parameters

Parameter	Description
Monitor Magic ID	Set to 0x214d4f4e ("!MON").
Msg Type	Set to 0x1 for a Bind Request.
Source Entity ID	Set to a value above 0x20. For more information, refer to *Entity and instance IDs* on page 13.
Source Instance ID	Set to 0 for host computers.

Format

The following illustration shows the format of the data [] array:



Unbind Request

Tears down communication between the application and the monitor task. An application must send this message only once when it is shutting down.

Direction

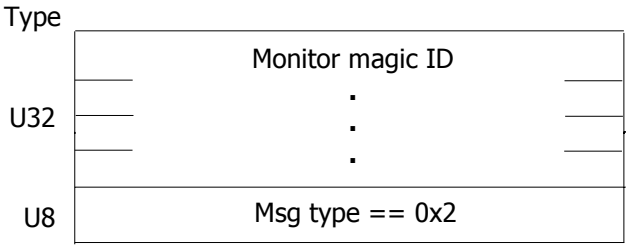
Application --> Monitor

Parameters

Parameter	Description
Monitor magic ID	Set to 0x214d4f4e ("!MON").
Msg type	Set to 0x2 for an Unbind Request.

Format

The following illustration shows the format of the data [] array:



Data Indication

Contains the raw SS7 message.

Direction

Monitor --> Application

Details

The data[] array can contain multiple SS7 messages. The following table describes the content of each message:

Message content	Description
link number	Specifies the link on which the SS7 message was received.
spare indicator	Indicates the number of spare bytes that follow the raw SS7 data portion of the message. The spare indicator aligns the fields so that the data length field of the next message starts on a 32-bit boundary.
timestamp	Indicates the time the message was received. The time is expressed as the number of seconds in epoch time format.
data	Contains the SS7 message.
length field	Sets the length field to the data length of the SS7 message. The length field contains only bytes of message data, and does not include the link number or timestamp.

After a message is decoded, the application looks at the next byte available after the spares. This next byte becomes the length field for the next message. If the length field is set to 0, the application is at the last message in the data[] array.

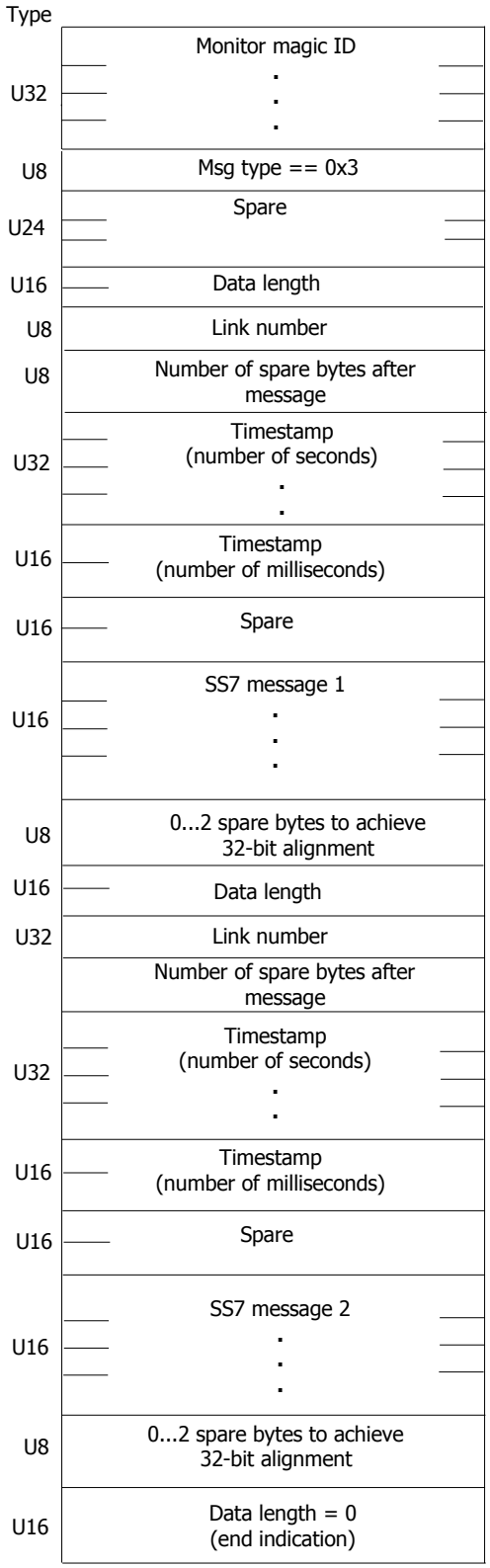
The timer tInd in the general configuration defines how long to hold a buffer before sending a Data Indication message. If the buffer fills, the Data Indication message is sent before the timer expires. If the timer is set to 0, each Data Indication contains only one message.

Parameters

Parameter	Description
Monitor magic ID	Set to 0x214d4f4e ("!MON").
Msg type	Set to 0x3 for a Data Indication.

Format

The following illustration shows the format of a Data Indication with two SS7 messages:



Index