Dialogic
Making Innovation Thrive™

Dialogic® TX Series SS7 Boards

TDM for SS7 Developer's Reference Manual

July 2009 64-0454-01

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.

Any use case(s) shown and/or described herein represent one or more examples of the various ways, scenarios or environments in which Dialogic® products can be used. Such use case(s) are non-limiting and do not represent recommendations of Dialogic as to whether or how to use Dialogic products.

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.

The 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.

Revision history

Revision	Release date	Notes
9000-6475-10		GJG
9000-6475-11	November 2000	GJG, SS7 3.6
9000-6475-12	August 2001	GJG, SS7 3.8
9000-6475-13	November 2003	MCM, SS7 4.0 Beta
9000-6475-14	April 2004	MCM, SS7 4.0
9000-6475-15	August 2004	SRR, SS7 4.1
9000-6475-16	September 2008	SRG/LBG, SS7 5.0
64-0454-01	July 2009	LBG, SS7 5.1
Last modified: July 7, 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® TX Series SS7 Boards TDM for SS7 Developer's Reference Manual* explains how to use the H.100/H.110 switching control library, the T1/E1 library, and the TX SWI library to build applications that control H.100/H.110 switching and the T1/E1 interfaces.

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

2 Overview of the TDM libraries

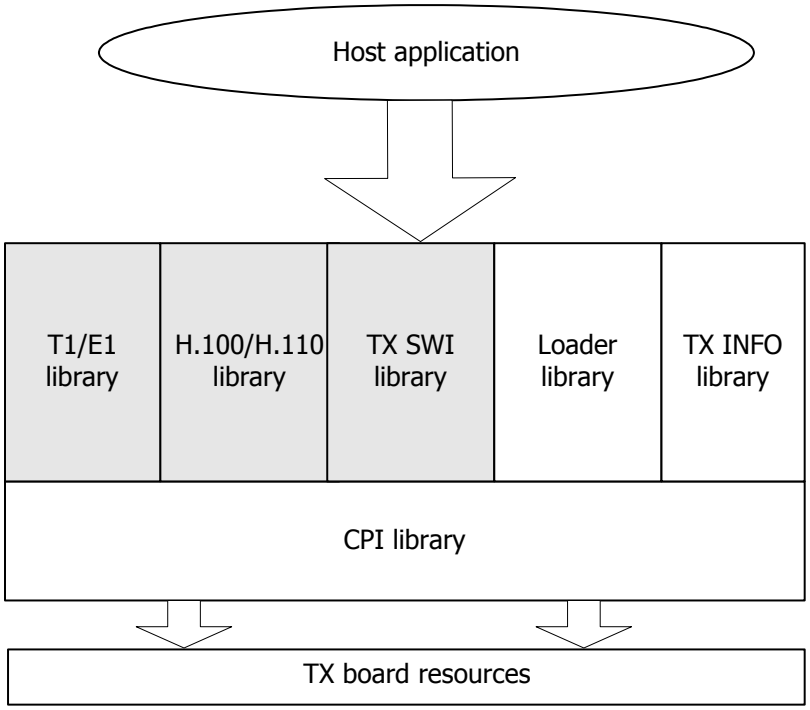
Development environment

The TX host application development environment consists of libraries that enable you to configure and control the protocol engines loaded on the TX board. This manual describes the following TDM libraries:

- H.100/H.110 switching control library
- T1/E1 library
- TX SWI library

The TDM libraries provide high-level interfaces for host applications to control H.100/H.110 switching and the T1/E1 trunks. The H.100/H.110 switching control library and the T1/E1 library provide simple synchronous interfaces to the H.100/H.110 and T1/E1 manager tasks. The functions issue requests and wait for responses from the TX board, blocking the calling application before returning control to the application. The TX SWI interface provides an alternative method for controlling both the H.100/H.110 switch fabric and the T1/E1 trunks on TX boards.

The TDM functions use the services of the H.100/H.110, T1/E1, and SWI manager tasks on the TX board through the standard CPI driver and the CPI channels dedicated to H.100/H.110, T1/E1, and SWI control messages.



PSTN interfaces

The TX board communication processors support the following interfaces for transmitting and receiving digital data, audio, and video signals to and from public switched telephone networks:

Interface	Function
H.100/H.110 bus	Gives the TX board access to the H.100/H.110 bus through a non-blocking digital timeslot switch. The H.100/H.110 bus interface is an enhanced switching-compliant device that makes connections between the H.100/H.110 bus, the T1/E1/J1 interfaces, and the on-board communication controllers.
Quad T1/E1	Standard for PCI and PCI Express TX boards. Provides a flexible interface for terminating four trunk lines that can be configured as T1, E1, or J1. Each trunk line can be configured independently. The T1 (or J1) adapter provides a DSX-1 interface. It can be connected to the public switched telephone network through an external channel service unit (CSU), or it can be directly connected to other local equipment that provides a compatible interface. In the case of a direct local connection, TX board T1 lines can be independently configured as either the loop master (timing source of the circuit) or as the loop slave (deriving clocking from the circuit). The T1 lines can also be configured for D4F4, ESF, and F72 framing formats and NOZCS, B7ZS, or B8ZS line coding formats. E1 lines can be independently configured for standard CCS, CAS, CCSCRC, and CASCRC framing formats and NOZCS or HDB3 line coding formats. Each line can be optionally configured as the timing source (loop master) of the circuit or as the loop slave that derives clocking from the circuit.
Octal T1/E1	Standard for CompactPCI TX boards. Provides a flexible interface for terminating eight trunk lines that can be configured as T1, E1, or J1. Each trunk line can be configured independently. The T1 (or J1) adapter provides a DSX-1 interface. It can be connected to the public switched telephone network through an external channel service unit (CSU), or it can be directly connected to other local equipment that provides a compatible interface. In the case of a direct local connection, TX board T1 lines can be independently configured as either the loop master (timing source of the circuit) or as the loop slave (deriving clocking from the circuit). The T1 lines can also be configured for D4F4, ESF, and F72 framing formats and NOZCS, B7ZS, or B8ZS line coding formats. E1 lines can be independently configured for standard CCS, CAS, CCSCRC, and CASCRC framing formats and NOZCS or HDB3 line coding formats. Each line can be optionally configured as the timing source (loop master) of the circuit or as the loop slave that derives clocking from the circuit.

TDM library services

The TDM libraries provide the following services:

- Circuit switching
- Dedicated data channels

Circuit switching

The circuit switching service enables switching between any pair of timeslots on the T1/E1/J1 and H.100/H.110 interfaces. Circuit switching includes connections between:

- An H.100/H.110 bus timeslot and a T1/E1/J1 timeslot.
- Two H.100/H.110 timeslots.
- Two T1/E1/J1 timeslots on the same or different T1/E1/J1 interfaces without using H.100/H.110 bus timeslots.

Dedicated data channels

The circuit switching service enables timeslots from either T1/E1/J1 interfaces or the H.100/H.110 bus to be nailed up to communication controllers on the TX board to form individual 64, 56, or 48 kbit/s Ds0 channels. Ranges of timeslots can be grouped together to form higher bandwidth hyperchannels.

Maintaining SS7 link data integrity over hyperchannels may not be possible in certain hardware configurations. If SS7 links of greater than standard capacity are desired, then high speed links should be used. All channels of a T1, E1, or J1 trunk can be combined to form a single high speed link.

Circuit switched and nailed up timeslots can be mixed on the same T1/E1/J1 interface and on the H.100/H.110 bus. For example, a single T1/J1 interface can consist of a single timeslot nailed up to a communication controller carrying SS7 (out-of-band) signaling data (processed by the signaling software) and 23 circuit switched voice and data channels under control of an application on the host. The host application instructs the TX board to make connections between the T1/E1/J1 timeslots and the H.100/H.110 bus (or other T1/E1/J1 timeslots) based on signaling messages received on the SS7 data channel.

3 Using the H.100/H.110 library

Accessing TX boards using the H.100/H.110 library

The H.100/H.110 library enables the host application to configure and control the H.100/H.110 switch fabric. It is compatible with the MVIP-95 software standards with some extensions. The H.100/H.110 manager provides the interface from the host to the H.100/H.110 switching control interface on the TX board. The H.100/H.110 manager translates requests from the host through the CPI H.100/H.110 control channel into low-level driver requests. A response is returned over the same CPI channel for each request received.

The *mvipapi.h* include file provides H.100/H.110 function prototypes. Refer to the *t1demo* source code for a working example of using the H.100/H.110 library.

To open the H.100/H.110 library, call **TxMvipOpen** before calling any other functions for a particular board. To close the H.100/H.110 library, call **TxMvipClose**.

Note: Use the TX SWI library instead of the H.100/H.110 library to control switching. Refer to *Accessing TX boards using the TX SWI library* on page 27 for more information.

Initializing switching control

Initialization of the TX board switching control interface consists of:

- Resetting the switch fabric to a known state (disabled with all connections cleared and all outputs tri-stated)
- Setting the clock configuration
- Enabling the switch fabric

Each time you reload a board with *ss7load*, the TX board switching control is reset according to a TDM configuration file. Use the following functions to override the configuration file settings:

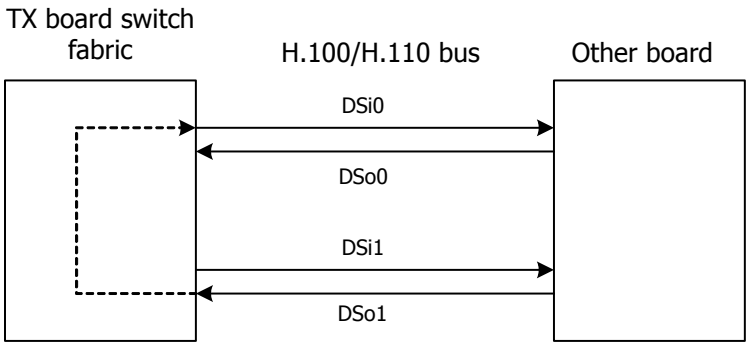
To...	Use this function...
Reset the switch fabric, clear all connections, and disable (tri-state) all outputs	**TxResetSwitch** **Note:** Using **TxResetSwitch** disrupts any configured dedicated data channels.
Set the clock configuration, specify whether or not the TX board drives the main and secondary 8 kHz (SEC8k) clock signals, and specify if the TX board clock is free running or locked to one of its T1/E1/J1 interfaces or the SEC8k signal	**TxConfigClock**
Disable and enable the switch, enabling the TX board to transmit on the H.100/H.110 bus when connections are established	**TxTristateSwitch**

Making and breaking connections

To establish and clear switched connections, use **TxSetOutput** and **TxSetOutputFdx**. These functions accept a destination stream and timeslot, a source stream and timeslot, and a connection mode as input to configure the switch fabric appropriately. The following connection modes are available:

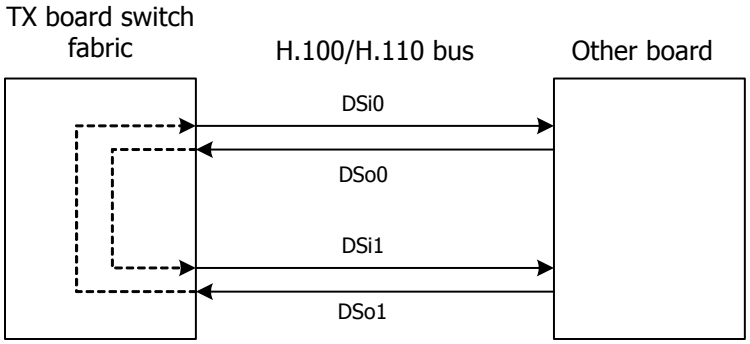
Connection mode	Description
CONNECT_MODE	Establishes a connection between the specified source stream and timeslot and destination stream and timeslot. **TxSetOutput** sets up a half duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot only. **TxSetOutputFdx** sets up a full duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot. Data sampled for the destination stream and timeslot's corresponding input is transmitted on the specified source stream and timeslot's corresponding output.
DISABLE_MODE	Clears a connection by disabling (tri-stating) the output specified by the destination stream and timeslot. **TxSetOutputFdx** also disables the output side corresponding to the source stream and timeslot, effectively clearing both directions of a full duplex connection.
PATTERN_MODE	Forces a specified one-byte pattern to be repeatedly transmitted on the destination stream and timeslot (source stream and timeslot is ignored).

The following illustration shows a half duplex H.100/H.110 connection:



TxSetOutput (0, 16, CONNECT_MODE, 1, 16)

The following illustration shows a full duplex H.100/H.110 connection:



TxSetOutputFdx (0, 16, CONNECT_MODE, 1, 16)

When you create a full duplex connection with **TxSetOutputFdx**, the TX board automatically determines the reverse direction stream numbers. Full duplex connections are always created across a pair of adjacent streams. When the stream number specified as a parameter to **TxSetOutputFdx** (***stream***) is odd, the paired stream number is ***stream*** – 1. When the stream number is even, the paired stream number is ***stream*** + 1. This is done independently for both the source and destination sides of a connection.

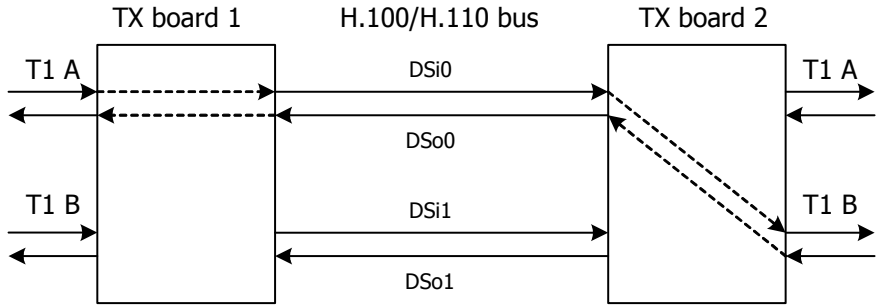
You can configure the TX board to transmit or receive across a given channel for any H.100/H.110 stream and timeslot. There is no restriction to even or odd stream numbers.

The H.100/H.110 library uses pseudo stream numbers to refer to local streams providing access to T1/E1/J1 trunks. These pseudo stream numbers are needed since the H.100/H.110 interface provides for the specification of MVIP bus versus LOCAL bus. The pseudo stream numbers are used as follows:

TX board type	Stream numbers
PCI and PCI Express T1/E1/J1 trunks 1 - 4	80 - 83
CompactPCI T1/E1/J1 trunks 1 - 8	80 - 87

Note: Switch fabric streams that correspond to T1/E1/J1 lines (streams 80 – 83 on PCI and PCI Express TX boards, and streams 80 - 87 on CompactPCI TX boards) do not switch between input and output on a given signal line. There is no pairing of even and odd stream numbers across a T1/E1/J1 connection. For **TxSetOutputFdx**, no T1/E1/J1 stream numbers are altered for the reverse connection.

The following illustration shows an example of setting up a full duplex connection between two T1 lines on different TX boards using the H.100/H.110 bus. This example shows a connection between T1 line A (stream 80), timeslot 2 on board 1 and T1 line B (stream 81), timeslot 7 on board 2. The connection between the two boards is set up over H.100/H.110 bus streams 0 and 1, timeslot 9.



TxSetOutputFdx
(80, 2, CONNECT_MODE, 0, 9)
80(T1/E1 A):2 <=== 0(H.100):9
80:2 ===> 1:9

TxSetOutputFdx
(81, 7, CONNECT_MODE, 1, 9)
0:9 <=== 81(T1/E1 B):7
1:9 ===> 81:7

4 Using the T1/E1 library

Accessing TX boards using the T1/E1 library

The T1/E1 library enables the host application to configure, monitor the status of, and collect performance statistics for the T1/E1/J1 trunks. The T1/E1 manager provides the interface from the host to the T1/E1 device drivers on the TX board. The T1/E1 device driver provides the interface for configuring, conditioning, and monitoring the T1/E1/J1 channels. The T1/E1 manager translates requests from the host application through the CPI T1/E1 control channel into low-level T1/E1 interface control. The T1/E1 manager monitors each line for alarm and out-of-service states and reports changes to the host application through unsolicited messages on the T1/E1/J1 status channel.

Note: The T1/E1 library is also applicable to J1.

The *t1e1api.h* include file provides T1/E1 function prototypes. The *t1e1type.h* include file provides structure definitions. Refer to *TDM demonstration program: t1demo* on page 84 for a working example of using the T1/E1 library.

To open the T1/E1 library, call **TxT1E1Open** before calling any other functions for a particular board. To close the T1/E1 library, call **TxT1E1Close**.

Note: Use the TX SWI library instead of the T1/E1 library to manage T1/E1/J1 trunks. Refer to *Accessing TX boards using the TX SWI library* on page 27 for more information.

Configuring T1/E1 trunks

Before the board can communicate with remote equipment, the T1/E1/J1 trunk configuration options must be set, including the:

- Type of framing employed on the line
- Line coding format
- Clock source

A TDM configuration file automatically performs initial configuration as part of *ss7load*. To override the information in the TDM configuration file, use **TxT1E1ConfigCarrier**. Refer to the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual* for information about *ss7load*.

Note: If you do not download a TDM configuration file to the board (if you have no dedicated data channels), you must call **TxT1E1ConfigCarrier** before attempting to make connections using the T1/E1/J1 trunks.

Monitoring T1/E1 line status

To determine the current line status, call **TxT1E1CarrierStatus** at any time. **TxT1E1CarrierStatus** returns the following information:

- Current state of the line (carrier and frame synchronization status)
- Whether or not an alarm is currently present
- Current configuration parameters (such as framing mode and line coding)
- Current performance statistics

Optionally, the application can request to be notified through an unsolicited message any time the synchronization state of the line changes (carrier or synchronization established or lost) or any time an alarm condition is detected or cleared.

Unsolicited status messages are delivered out-of-band. They come to the application through a separate interface to the TX device driver rather than directly from the library. The application must first call **TxT1E1SuperviseCarrier** to register the events for which it wants to be notified. Refer to *Handling unsolicited T1/E1 notifications* on page 22 for more information.

T1/E1 statistics and performance measures

TxT1E1CarrierStatus returns the statistics for the current 15 minute interval and the current 24 hour interval. To obtain a full performance report for the previous 96 15 minute intervals plus the 24 hour summary totals, call **TxT1E1PerfReport**.

The application can also use **TxT1E1SuperviseCarrier** to request performance reports automatically every 15 minutes or every 24 hours. Performance reports are returned to the application in the form of unsolicited messages using the same TX device driver interface as the unsolicited status and alarm messages.

Event counters and performance measures

The Signaling software maintains event counters and performance measures. Event counters are 16-bit accumulators that are reset only upon request from the host application. Performance measures are counts maintained for:

- The current 15 minute interval
- The previous 96 15 minute intervals (24 hours)
- The sum total for the previous 24 hours

Event counters

The following table lists the event counters:

TR 54016 name	RFC 1406 name	Description
ESF error event	Path code violation	Accumulated frame synchronization or CRC error.
Error event	N/A	Line code violation, accumulated bipolar violation, or excessive zeroes error.
Controlled slip	Controlled slip (CS)	Accumulated replication or deletion of received frames due to timing slips.
E-bit error event	N/A	Number of E-bit errors (CRC errors) reported back by far end through CRC4 multiframe E-bit indicator.

Performance measures

The following table lists the performance measures (all counts are relative to a 15 minute interval):

TR 54016 name	RFC 1406 name	Description
Errored seconds (ES)	Errored seconds (ES)	Number of seconds (0 through 900) with one or more CRC/frame synchronization errors, out-of-frame declarations, or controlled slips.
N/A	Line errored seconds	Number of seconds (0 through 900) with one or more line code violations.
Bursty errored seconds (BES)	Bursty errored seconds (BES)	Number of seconds (0 through 900) with (BES) 1 < path code errors < 320.
Severely errored seconds (SES)	Severely errored seconds (SES)	Number of seconds (0 through 900) with path code errors greater than or equal to 320.
Controlled slip seconds	Controlled slip seconds	Number of seconds (0 through 255) with one or more controlled slips.
Loss of frame count	N/A	Number of out-of-frame defects declared (0 through 255).

The current interval counts and the previous 96 interval counts are available on request from the host application. The following RFC 1406 (DS1/E1 MIB) measures are not explicitly supported:

- Severely errored framing seconds
- Degraded minutes

Handling unsolicited T1/E1 notifications

An application can choose to receive unsolicited notifications of T1/E1/J1 line status changes, alarms, and periodic performance reports. The following events can be requested:

- Establishment and loss of carrier and frame synchronization
- Alarms set and cleared
- 15 minute performance reports
- 24 hour performance reports

Unsolicited notifications are received directly from the TX device driver rather than through the library calls.

This topic presents:

- Registering for unsolicited T1/E1 notifications
- Unsolicited message formats
- Carrier status/alarm change messages
- Performance report messages

Registering for unsolicited T1/E1 notifications

Complete the following steps to register for unsolicited status and performance notifications:

Step	Action
1	Call **cpi_init** to initialize the CPI library.
2	Open the CPI T1/E1 status channel by calling **cpi_open** or **cpia_open**, specifying a channel of PT T1E1S (the T1/E1 status channel).
3	Call **TxT1E1Open** to obtain a valid handle.
4	Call **TxT1E1SuperviseCarrier** to enable unsolicited notifications.
5	Call **TxT1E1Close** to return the configuration channel.
6	When the handle returned by **cpi_open** or **cpia_open** is signaled, call **cpi_get_data** or **cpia_intr** to receive status messages.

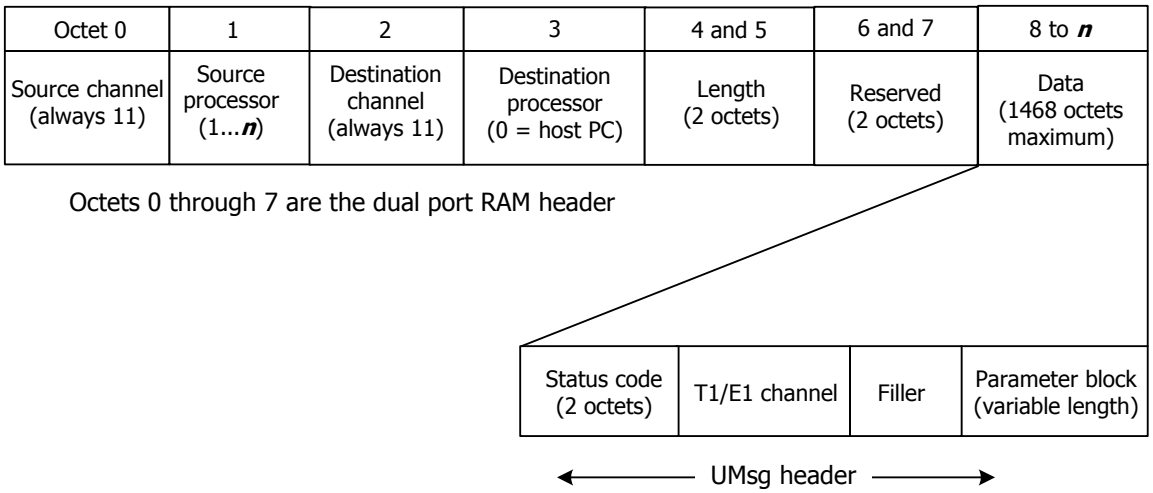
Note: Refer to *T1/E1 line status demonstration program: t1stat* on page 89 for a working example of how to receive unsolicited T1/E1 status messages and performance reports.

Unsolicited message formats

Each message consists of:

- A standard dual port RAM header (DPRH) that interfaces with the device driver. Refer to *Dual port RAM header (DPRH)* on page 23 for more information.
- An unsolicited message header (UMsg) that identifies the notification type and T1/E1 line. Refer to *Unsolicited message header (UMsg)* on page 24 for more information.
- A parameter block that is specific to the notification type.

The following illustration shows the unsolicited status message format:



Byte ordering considerations

All 16-bit and 32-bit fields in the data portion of the CPI packet must be passed to and from the board in the board's native format. When using the TX SWI library, all field conversions are handled by the library functions. No additional conversions are required. Conversions should be performed for the T1/E1 and H.100/H.110 switching control libraries. Use the CPI library functions **cpi_htocp_s** and **cpi_htocp_l** to change from the host's native format to the board's native format. Use **cpi_cptoh_s** and **cpi_cptoh_l** when converting fields received from the TX board. Refer to the *Dialogic® TX Series SS7 Boards CPI Library Developer's Reference Manual* for more information.

Note: The length field in the CPI header is never converted by the host application. The length field is always in the host's native format.

Dual port RAM header (DPRH)

All messages to and from a TX processor contain a dual port RAM header (DPRH). For T1/E1 status messages, use the DPRH to identify the sending board (srcbd field) and the byte length of the message (len field).

The source and destination channel numbers are always PT_T1E1S for the T1/E1 status messages. The following example shows the DPRH layout:

```
typedef struct __dprh
  U8  srcch;  /* Source channel number        */
  U8  srcbd;  /* Source board number (1..8)   */
  U8  dstch;  /* Destination channel number   */
  U8  dstbd;  /* Destination board # (0 = PC) */
  U16 len;    /* Len of msg body (incl hdr)   */

U8  reserved[2];  /* reserved for future use (pad to 32-bit aligned size) */
 } DPRH;
```

Unsolicited message header (UMsg)

The UMsg header identifies the type of notification and the line to which it refers (A, B, C, or D):

```
typedef struct txT1E1UHdr
{
  U16  statusCode; /* Request codes from tdmuser.h */
  U8   carrier;    /* Which T1/E1 line             */
  U8   spare1;     /* Word align what follows      */
} TxT1E1UHdr;
```

Possible values for statusCode are:

Value	Description
STATUS_CARRIER	Line status change.
CARRIER_ALARM	Change in alarm state.
PERFREP_15MIN	15-minute performance report.
PERFREP_24HR	24-hour performance report.

Possible values for carrier are:

Value	Description
NET_T1A	T1/E1 line A
NET_T1B	T1/E1 line B
NET_T1C	T1/E1 line C
NET_T1D	T1/E1 line D
NET_T1E	T1/E1 line E (CompactPCI TX only)
NET_T1F	T1/E1 line F (CompactPCI TX only)
NET_T1G	T1/E1 line G (CompactPCI TX only)
NET_T1H	T1/E1 line H (CompactPCI TX only)

Carrier status/alarm change messages

The unsolicited status structure provides carrier status and alarm notifications information:

```
typedef struct txT1E1UStat
{
  TxT1E1UHdr     uhdr;   /* Unsolicited message header            */
  U8  alarmState;        /* Alarm status (CARRIER_ALARM only)    */
  U8  lastAlmState;      /* Previous alarm status                */
                         /* (CARRIER_ALARM only)                 */
  U8  wordalign;         /* Filler for alignment                 */
  U8  syncState;         /* New synch state (STATUS_CARRIER only) */
} TxT1E1UStat;
```

The alarm state fields are bit maps with a 1 bit indicating that the alarm condition is present and a 0 bit indicating that the alarm condition is cleared. Including both the current and previous alarm states allows the application to determine which alarms changed state and whether the alarm condition was just detected or just cleared.

Possible values for alarmState and lastAlmState are:

- YELLOW_ALARM
- BLUE_ALARM
- TS16SIG_ALARM
- TS16AIS_ALARM
- ISTANTMF_ALARM

Refer to *T1/E1 carrier alarms* on page 26 for alarm conditions.

The syncState field is used only in the STATUS_CARRIER notification message. Possible values are:

Value	Description
CARRIER_SYNC	Carrier signal fully synchronized.
CARRIER_NOSYNC	Carrier signal detected but not synchronized.
CARRIER_LOST	No carrier signal detected.

Performance report messages

The 15 minute performance report contains the performance measures for the previous 15 minute period and the current values for the event accumulators:

```
typedef struct txT1E1Perf15
{
  TxT1E1UHdr   uhdr;       /* Unsolicited message header       */
  TxT1E1Stats  currStats;  /* Current interval statistics      */
  U16          pcvs;       /* Accumulated path code violations */
  U16          lcvs;       /* Accumulated line code violations */
  U16          slips;      /* Accumulated controlled slips     */
  U16          ebits;      /* Accumulated E-bit errors (E1)    */
} TxT1E1Perf15;
```

The 24 hour performance report contains the performance statistics for the preceding 96 15 minute measurement intervals, the sum of each performance measure for the last 24 hours, and the current values for the event accumulators.

If less than 96 intervals are available, the validInts field contains the number of valid intervals actually included in the performance report message.

```
typedef struct txT1E1Perf24
{
  TxT1E1UHdr   uhdr;          /* Unsolicited message header       */
  U16          validInts;     /* Number of intervals included     */
  TxT1E1Stats  intStats[96];  /* Last 96 15-minute intervals      */
  TxT1E1Stats  sumStats;      /* 24-hour summary statistics       */
  U16          pcvs;          /* Accumulated path code violations */
  U16          lcvs;          /* Accumulated line code violations */
  U16          slips;         /* Accumulated controlled slips     */
  U16          ebits;         /* Accumulated E-bit errors (E1)    */
} TxT1E1Perf24;
```

T1/E1 carrier alarms

The TX board monitors carrier alarm conditions. If the application requests it, the setting and clearing of these alarm states can be reported as unsolicited status events or can be returned on request with **TxT1E1CarrierStatus**. The TX board can also report other carrier status changes such as establishment and loss of synchronization and detection or loss of carrier signal.

The following table lists the alarms detected by the TX board:

Alarm state	Entry criteria	Typical usage
Blue alarm (AIS)	Unframed all ones signal received for 3 ms (T1/J1) or two full frames (E1).	Far end cannot transmit normal signal (for example, a repeater lost the incoming carrier on the other side).
Yellow alarm (distant or far end)	D4: Bit 2 in every channel set to 0 for ~ 335 ms. ESF: 16 consecutive patterns of 0x00FF appear in the facility data link (FDL). E1: Bit 3 of timeslot 0 in non-align frames set to 1 for three consecutive occasions.	Far end lost incoming signal from reporting node.
Timeslot 16 alarm indication signal (E1 only)	E1 CAS mode only: Timeslot 16 contains all 1s for 16 consecutive frames (one full multiframe).	Far end lost normal CAS signaling capability (for example, lost incoming CAS signal on other side).
Timeslot 16 signal lost (E1 only)	E1 CAS mode only: Timeslot 16 contains all 0s for 16 consecutive frames (one full multiframe).	Failure of equipment providing CAS signaling.
Distant multiframe alarm (E1 only)	E1 CAS mode only: Bit 6 of timeslot 16 of frame 0 set to 1 for two consecutive multiframes.	Far end lost CAS multiframe alignment.

5 Using the TX SWI library

Accessing TX boards using the TX SWI library

The TX SWI library enables host applications to control H.100/H.110 switching and the T1/E1/J1 trunks by sending requests to TX boards and processing response messages received from the boards. The calling application uses the CPI library directly to send and receive TX SWI messages. Using the CPI asynchronous interface, an application can have many switching commands in progress simultaneously.

The SWI manager provides the interface from the host to all TDM devices on the TX board, including the T1/E1 devices and the H.100/H.110 switching control device. For each SWI request issued, an asynchronous response is returned over the same CPI channel.

The *txswiapi.h* include file provides TX SWI function prototypes. The *swidef.h* include file provides structure and literal definitions.

To begin using the TX SWI library, open a channel to the TX board by calling **cpi_open** or **cpia_open**. The recommended CPI channel range is 32 through 127. After a CPI channel is opened, the application can send requests to the target TX board. Refer to *TX SWI library demonstration program: txsdemo* on page 90 for a working example of using the TX SWI library.

The TX SWI library parallels the Natural Access Switching service that controls TDM switching on other Dialogic. This manual describes the differences between the TX SWI library and the Switching service. Use the function descriptions found in the *Switching Service Developer's Reference Manual* for detailed descriptions of the TX SWI functions.

Sending and receiving requests

To send a switching request to the TX board, the calling application must perform the following steps:

Step	Action
1	Initialize all parameters expected by the TX SWI function. Most TX SWI functions have information blocks associated with the function.
2	Determine the packet size by calling the TX SWI function and providing all parameters except the CPIPKT. Pass the CPIPKT as NULL to indicate that the CPI packet size is needed. The TX SWI function returns the byte length of the CPI packet required to issue the request.
3	Obtain the CPIPKT by either dynamically allocating memory for the CPI packet or pointing to an available global memory space that is large enough to use as the CPI packet.
4	Address the packet by calling **txswiAddress** to zero-fill the packet and to set the addressing information in the CPI packet header.
5	Format the request packet by calling the same TX SWI function used in Step 2 to determine the required packet size, passing all the same parameters, and passing the address of the CPI packet rather than a NULL pointer. This command triggers the TX SWI function to fill in the payload of the request message.
6	Send the request to the board by calling **cpia_send**.
7	Expect a response. The TX board processes the TX SWI request message and issues an asynchronous response message after the request is serviced. These response messages are received like any other message from the TX board, with the receive notification provided to the **cpia_open** call.

To receive TX SWI responses, the application must perform the following steps:

Step	Action
1	Handle the response indication. When a signal is received indicating that the CPI channel requires servicing, call **cpia_intr** to begin handling the notification. **cpia_intr** calls the receive notification function provided to **cpia_open**. Call **cpia_get_data** to get a pointer to the received packet.
2	Decode the response packet. After calling **cpia_get_data**, call **txswiReply** to decode the received message from inside the response notification.
3	Check for errors. Verify that NULL was not returned from **txswiReply** and that the returned completion code is zero.
4	Process successes. Perform any processing that is waiting on the response message.

To stop using the TX SWI library, call **cpi_close** to close the CPI channel.

Function cross-reference

The following table lists the TX SWI functions and their corresponding Switching service functions:

TX SWI function	Switching service function
txswiAddress	No equivalent. Use to zero-fill and address request messages bound for the TX board.
txswiConfigBoardClock	**swiConfigBoardClock**
txswiConfigLocalStream	**swiConfigLocalStream**
txswiConfigLocalTimeslot	**swiConfigLocalTimeslot**
txswiConfigNetrefClock	**swiConfigNetrefClock**
txswiConfigSec8KClock	**swiConfigSec8KClock**
txswiDisableOutput	**swiDisableOutput**
txswiGetBoardClock	**swiGetBoardClock**
txswiGetBoardInfo	**swiGetBoardInfo**
txswiGetLocalStreamInfo	**swiGetLocalStreamInfo**
txswiGetOutputState	**swiGetOutputState**
txswiGetSwitchCaps	**swiGetSwitchCaps**
txswiGetTimingReference	**swiGetTimingReference**
txswiMakeConnection	**swiMakeConnection**
txswiReply	No equivalent. Use to decode any received TX SWI response message.
txswiResetSwitch	**swiResetSwitch**
txswiSampleInput	**swiSampleInput**
txswiSendPattern	**swiSendPattern**

6 H.100/H.110 function reference

Using the function reference

This section provides an alphabetical reference to the H.100/H.110 functions. A typical function includes:

Prototype	The prototype is shown followed by a listing of the function arguments. Data types include: • U8 (8-bit unsigned) • S8 (8-bit signed) • U16 (16-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • S32 (32-bit signed) If a function argument is a data structure, the complete data structure is defined. **Note:** Some parameters are not applicable to both ANSI and ITU-T (CCITT) networks.
Return values	The return value for a function is either SUCCESS or an error code.

Note: Call **TxMvipOpen** before calling any other H.100/H.110 functions for a board.

Function summary

All H.100/H.110 functions are synchronous, suspending the calling application until a response is received from the TX board.

Library access functions

Function	Description
TxMvipOpen	Opens an H.100/H.110 channel for the specified TX board.
TxMvipClose	Closes an H.100/H.110 channel to a particular board and makes it available to other processes.

Initialization functions

Function	Description
TxResetSwitch	Disables the H.100/H.110 interface switch fabric and clears all connections.
TxTristateSwitch	Enables or disables the H.100/H.110 interface switch fabric (enables or tri-states all H.100/H.110 bus output signals).
TxConfigClock	Sets the clocking configuration of the H.100/H.110 interface.

Status functions

Function	Description
TxQuerySwitchCaps	Returns the capabilities of the H.100/H.110 switch fabric interface and the number of T1/E1/J1 interfaces and channels available on the T1/E1/J1 adapter.
TxSwitchStatus	Returns the current clocking configuration and state of the H.100/H.100 switch fabric interface.

Input and output functions

Function	Description
TxSetOutput	Makes or clears half duplex connections or continuously transmits a pattern on a switch output.
TxSetOutputFdx	Makes and clears full duplex connections or continuously transmits a pattern on a switch output.
TxQueryOutput	Queries the current output status of a specified timeslot on a specified stream.
TxSampleInput	Retrieves the most recent input byte received on a particular switch input stream/timeslot.

TxConfigClock

Sets the clocking configuration of the H.100/H.110 interface.

Prototype

S16 **TxConfigClock** (S16 ***handle***, CLOCK_T ***clock***, SEC8K_T ***sec8k***, NETWORK_T ***network***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
clock	Mode for the main clock signals. Refer to Details for valid options.
sec8k	Source of the SEC8K clock signal. Refer to Details for valid options.
network	Clock reference source for the master or SEC8K clock. Refer to Details for valid options.

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_LENGTH	Length of the request packet is invalid.
MVIP_INVALID_STREAM	Output or source stream number is out of range.
MVIP_INVALID_TIMESLOT	Output or source timeslot number is out of range.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

TxConfigClock specifies:

- Whether or not the TX board is the master or slave of the main clock signals
- The source of the TX board clock circuit reference input (internal, T1/E1/J1 interface, or SEC8K clock signal)
- Whether or not the TX board drives the SEC8K clock signal

This request overrides the clocking settings specified in the *txconfig* or *tdmcfg* configuration files.

The following tables provide the valid options for the ***clock*** and ***sec8k*** parameters.

Valid options for the clock parameter

Parameter	Description
CLOCK_REF_MVIP	TX board is a slave.
CLOCK_REF_SEC8K	TX board is a master, referenced from the SEC8K signal.
CLOCK_REF_OSC	TX board is a master, free-running from the internal oscillator.
CLOCK_REF_NET	TX board is a master, referenced from the T1/E1/J1 interface specified in the ***network*** parameter.

Valid options for the sec8k parameter

Parameter	Description
SEC8K_NOT_DRIVEN	SEC8K not driven by the TX board.
SEC8K_DRIVEN_BY_OSC	SEC8K driven by the internal oscillator.
SEC8K_DRIVEN_BY_NET	SEC8K driven by the clock referenced from the T1/E1/J1 interface specified in the ***network*** parameter.

Valid options for the network parameter

Parameter	Description
NET_T1A through NET_T1D	PCI and PCI Express TX boards
NET_T1A through NET_T1H	CompactPCI TX boards

TxMvipClose

Closes the H.100/H.110 channel to a particular board and makes it available to other processes.

Prototype

void **TxMvipClose** (S16 ***handle***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.

Return values

Void.

TxMvipOpen

Opens an H.100/H.110 channel for a specified TX board.

Prototype

S16 **TxMvipOpen** (TXBOARD_T ***board***, U16 ***swint***)

Argument	Description
board	TX board number.
swint	Not used.

Return values

Return value	Description
MVIP_BAD_BOARD	Invalid board number.
MVIP_NO_DRIVER	TX driver not found.
MVIP_OPEN_FAIL	Opening of MVIP manager failed.

Details

TxMvipOpen returns the specified TX board handle to use in subsequent requests. Call this function before calling any other functions for a particular board.

TxQueryOutput

Queries the current output status of a specified timeslot on a specified stream.

Prototype

S16 **TxQueryOutput** (S16 ***handle***, STREAM_T ***oStream***, TIMESLOT_T ***oTimeslot***, MODE_T ***mode**, STREAM_T ***iStream***, TIMESLOT_T ***iTimeslot***, DATA_T ***message***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
oStream	Output stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1/J1 trunks 1 - 4 84 - 87 for T1/E1/J1 trunks 5 - 8 (CompactPCI TX only)
oTimeslot	Output timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 (or J1) 0 through 31 for E1
mode	Pointer to where the current mode of the specified output stream and timeslot is returned to the caller.
iStream	Pointer to where the input stream is returned to the caller when the mode is CONNECT_MODE (not used for other modes).
iTimeslot	Pointer to where the input timeslot is returned to the caller when the mode is CONNECT_MODE (not used for other modes).
message	Pointer to where the current value transmitted on the stream and timeslot is returned to the caller when the mode is PATTERN_MODE (not used for other modes).

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_STREAM	Output or source stream number is out of range.
MVIP_INVALID_TIMESLOT	Output or source timeslot number is out of range.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

The following table lists the valid values for the ***mode*** parameter:

Mode	Description
DISABLE_MODE	Output is disabled (not connected).
PATTERN_MODE	A fixed pattern is continuously transmitted on this channel.
CONNECT_MODE	Output is connected to the source channel specified in the ***iStream*** and ***iTimeslot*** parameters.

TxQuerySwitchCaps

Returns the capabilities of the H.100/H.110 switch fabric interface and the number of T1/E1/J1 interfaces and channels available on the T1/E1 adapter.

Note: Because a CompactPCI TX board provides more than two T1/E1/J1 interface lines, the T1/E1/J1 information provided by this function is not complete for these board types. Use the TX SWI function **txswiGetSwitchCaps** to determine CompactPCI TX switching capabilities.

Prototype

S16 **TxQuerySwitchCaps** (S16 ***handle***, int ***size***, int ****revision***, U16 ****domain***, U16 ****routing***, U16 ****blocking***, U16 ****networks***, U16 ****channels***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
size	Not used.
revision	Pointer to the revision number of the interface software.
domain	Pointer to the streams that are connected to the switch fabric.
routing	Pointer to the half duplex routing capabilities of the switch fabric.
blocking	Pointer to the possible blocking conditions in the switch fabric.
networks	Pointer to the number of T1/E1 lines for legacy devices. This value is not accurate for CompactPCI TX boards. PCI and PCI Express TX boards provide four T1/E1 lines. CompactPCI TX boards provide eight T1/E1 lines.
channels	Pointer to an array of two S16s where the number of channel timeslots available on each network interface (for example, channel[0] => T1/E1 A, channel[1] => T1/E1 B) is returned for legacy devices. This value is not accurate for CompactPCI TX boards. PCI and PCI Express TX boards provide four T1/E1 lines and CompactPCI TX boards provide eight T1/E1 lines. These lines can be configured as T1 or J1 (24 channels) or as E1 (32 channels).

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

Refer to the *MVIP-95 Device Driver Standard release 1.2* for a detailed description of the domain, routing, and blocking capability bit masks returned by this request.

The number of T1/E1/J1 lines is four for PCI and PCI Express TX boards and eight for CompactPCI TX boards. The number of available timeslots per T1/E1/J1 line is dependent on whether the given line is configured for T1 or J1 (24 channels) or E1 (32 channels).

TxResetSwitch

Disables the H.100/H.110 switch fabric interface and clears all connections.

Prototype

S16 **TxResetSwitch** (S16 ***handle***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

TxResetSwitch is usually issued before any other request. All nailed-up timeslots and dynamically defined connections are reset by this function. The H.100/H.110 switch fabric interface is also reset each time you reload a TX board using *ss7load*.

TxSampleInput

Retrieves the most recent input byte received on a particular switch input stream and timeslot.

Prototype

S16 **TxSampleInput** (S16 ***handle***, STREAM_T ***iStream***, TIMESLOT_T ***iTimeslot***, DATA_T ***iSample***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
iStream	Input stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only)
iTimeslot	Input timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
iSample	Pointer to where the current value received from the stream and timeslot is returned to the caller.

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_STREAM	Output or source stream number is out of range.
MVIP_INVALID_TIMESLOT	Output or source timeslot number is out of range.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

The data returned by **TxSampleInput** can change every 125 microseconds, and may not be accurate by the time the application receives the data. Sampling is most useful when testing a channel where a constant pattern is being transmitted.

TxSetOutput

Makes and clears half duplex connections or continuously transmits a pattern on a switch output.

Prototype

S16 **TxSetOutput** (S16 ***handle***, STREAM_T ***oStream***, TIMESLOT_T ***oTimeslot***, MODE_T ***mode***, STREAM_T ***iStream***, TIMESLOT_T ***iTimeslot***, DATA_T ***message***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
oStream	Output stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only)
oTimeslot	Output timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
mode	Mode of operation. Refer to Details for valid options.
iStream	Input stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only)
iTimeslot	Input timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
message	Value transmitted on the stream and timeslot when the mode is PATTERN_MODE (not used for other modes).

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_MODE	Invalid mode. Refer to Details for valid options.
MVIP_INVALID_STREAM	Output or input stream number is out of range.
MVIP_INVALID_TIMESLOT	Output or input timeslot number is out of range.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

Use **TxSetOutput** with **TxSampleInput** to test the ability of two TX boards to switch a digital path across the H.100/H.110 bus or their T1/E1 interfaces.

The result of this function is determined by the ***mode*** parameter. The following table lists acceptable values for the ***mode*** parameter and the resulting actions:

Connection mode	Description
CONNECT_MODE	Establishes a connection between the specified source stream and timeslot and destination stream and timeslot. **TxSetOutput** sets up a half duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot only. **TxSetOutputFdx** sets up a full duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot. Data sampled for the destination stream and timeslot's corresponding input is transmitted on the specified source stream and timeslot's corresponding output.
DISABLE_MODE	Clears a connection by disabling (tri-stating) the output specified by the destination stream and timeslot. **TxSetOutputFdx** also disables the output side corresponding to the source stream and timeslot, effectively clearing both directions of a full duplex connection.
PATTERN_MODE	Forces a specified one-byte pattern to be repeatedly transmitted on the destination stream and timeslot (source stream and timeslot is ignored).

Refer to *Making and breaking connections* on page 16 for more information.

TxSetOutputFdx

Makes and clears full duplex connections or continuously transmits a pattern on a switch output.

Prototype

S16 **TxSetOutputFdx** (S16 ***handle***, STREAM_T ***oStream***, TIMESLOT_T ***oTimeslot***, MODE_T ***mode***, STREAM_T ***iStream***, TIMESLOT_T ***iTimeslot***, DATA_T ***message***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
oStream	Output stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only)
oTimeslot	Output timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
mode	Mode of operation. Refer to Details for valid values.
iStream	Input stream number. Must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only)
iTimeslot	Input timeslot number. Must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
message	Value transmitted on the stream and timeslot when the mode is PATTERN_MODE (not used for other modes).

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_MODE	Invalid mode. Refer to Details for valid values.
MVIP_INVALID_STREAM	Output or input stream number is out of range.
MVIP_INVALID_TIMESLOT	Output or input timeslot number is out of range.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

The result of **TxSetOutputFdx** is determined by the ***mode*** parameter. The following table lists acceptable values for the ***mode*** parameter and the resulting actions:

Connection mode	Description
CONNECT_MODE	Establishes a connection between the specified source stream and timeslot and destination stream and timeslot. **TxSetOutput** sets up a half duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot only. **TxSetOutputFdx** sets up a full duplex connection in which data sampled from the specified source stream and timeslot is transmitted on the specified destination stream timeslot. Data sampled for the destination stream and timeslot's corresponding input is transmitted on the specified source stream and timeslot's corresponding output.
DISABLE_MODE	Clears a connection by disabling (tri-stating) the output specified by the destination stream and timeslot. **TxSetOutputFdx** also disables the output side corresponding to the source stream and timeslot, effectively clearing both directions of a full duplex connection.
PATTERN_MODE	Forces a specified one-byte pattern to be repeatedly transmitted on the destination stream and timeslot (source stream and timeslot is ignored).

Refer to *Making and breaking connections* on page 16 for more information.

TxSwitchStatus

Returns the current clocking configuration and state of the H.100/H.100 switch fabric interface.

Prototype

S16 **TxSwitchStatus** (S16 ***handle***, CLOCK_T ***clock***, SEC8K_T ***sec8k***, NETWORK_T ***network***, TRISTATE_T ***state***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
clock	Pointer to the mode for the main clock signals. Refer to *TxConfigClock* on page 33 for valid modes.
sec8k	Pointer to the source of the SEC8K clock signal. Refer to **TxConfigClock** for valid sources.
network	Pointer to the clock reference source for the master or SEC8K clock. Refer to **TxConfigClock** for valid sources.
state	Pointer to the current state of switch fabric returned to the caller. Valid values: ENABLE_OUTPUT TRISTATE_OUTPUT

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Specified handle is invalid.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

TxTristateSwitch

Enables or disables the H.100/H.110 switch fabric interface (enables or tri-states all H.100/H.110 bus output signals).

Prototype

S16 **TxTristateSwitch** (S16 ***handle***, TRISTATE_T ***state***)

Argument	Description
handle	Handle returned from a previous call to **TxMvipOpen** for the desired board.
state	Current state of the switch fabric returned to the caller. Must be either ENABLE_OUTPUT or TRISTATE_OUTPUT.

Return values

Return value	Description
SUCCESS	
MVIP_INVALID_HANDLE	Invalid handle.
MVIP_INVALID_PARAMETER	Invalid state parameter.
MVIP_NO_RESOURCES	MVIP adapter is not present on the communications processor.

Details

The H.100/H.110 switch fabric interface is also reset each time you reload a TX board using *ss7load*.

7 T1/E1 function reference

Using the function reference

This section provides an alphabetical reference to the T1/E1 functions. A typical function includes:

Prototype	The prototype is shown followed by a listing of the function arguments. Data types include: • U8 (8-bit unsigned) • S8 (8-bit signed) • U16 (16-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • S32 (32-bit signed) If a function argument is a data structure, the complete data structure is defined. **Note:** Some parameters are not applicable to both ANSI and ITU-T (CCITT) networks.
Return values	The return value for a function is either T1E1_SUCCESS or an error code.

Note: Call **TxT1E1Open** before calling any other T1/E1 functions for a board.

Function summary

All T1/E1 functions are synchronous, suspending the calling application until a response is received from the TX board.

Note: The T1/E1 library also applies to J1.

Library access functions

Function	Description
TxT1E1Open	Opens the T1/E1 manager on the specified TX board.
TxT1E1Close	Closes the T1/E1 manager for a TX board and makes it available to other processes.

Trunk configuration functions

Function	Description
TxT1E1ConfigCarrier	Configures the basic operating parameters (framing type, line coding, and line build out) for a T1/E1 carrier.
TxT1E1CtrlCarrier	Puts the carrier into loopback mode or forces it to transmit an alarm indication.

Channel configuration functions

Function	Description
TxT1E1ConditionChan	Selects a specific bit pattern to continuously transmit on one or all channels of a carrier.
TxT1E1ConfigChan	Forces a specific channel in or out of transparent mode.

Status functions

Function	Description
TxT1E1CarrierStatus	Queries the current status of a T1 or E1 carrier.
TxT1E1PerfReport	Requests an on-demand performance report.
TxT1E1ChannelStatus	Queries the current status of all channels on a carrier.
TxT1E1SuperviseCarrier	Registers the application to receive unsolicited notification of carrier alarm conditions, carrier synchronization state changes, and/or automatic 15 minute/24 hour performance statistics.

TxT1E1CarrierStatus

Queries the current status of a T1 or E1 carrier.

Prototype

S16 **TxT1E1CarrierStatus** (S16 ***handle***, U8 ***carrier***, U8 ***flags***, U8 ****pAlarmState***, U8 ****pLastAlmState***, U8 ****pSyncState***, TxT1E1Config ****pConfig***, U16 ***pCurrInterval***, TxT1E1Stats ***pCurrStats***, U16 ***pValidInterval***, TxT1E1Stats ****pSummaryStats***, U16 ***pPvcs***, U16 ***pLvcs***, U16 ***pSlips***, U16 ***pEbits***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
flags	Status request options flags. The only available option is T1E1_RESET_COUNTS, which resets statistics accumulators after returning values.
pAlarmState	Pointer to where the current alarm state is returned to the caller.
pLastAlmState	Pointer to where the previous alarm state is returned to the caller.
pSyncState	Pointer to where the current synchronization state is returned to the caller.
pConfig	Pointer to the configuration block for this carrier.
pCurrInterval	Number of seconds in the current statistics interval returned to the caller.
pCurrStats	Current 15 minute interval performance statistics returned to the caller.
pValidInterval	Number of 15 minute intervals collected in the summary statistics (always 96, except during first 24 hours after a boot, when it can be less than 96).
pSummaryStats	Pointer to the caller memory where the 24 hour summary statistics are returned.
pPcvs	Current value of the path code violations counter (accumulated since last reset) returned to the caller.
pLcvs	Current value of the line code violations counter (accumulated since last reset) returned to the caller.
pSlips	Current value of the controlled slips counter (accumulated since last reset) returned to the caller.
pEbits	Current value of the E-bit (E1) counter (accumulated since last reset) returned to the caller.

Return values

Return value	Description
T1E1_DEVICE_ERROR	Invalid adapter found.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

TxT1E1CarrierStatus displays the configuration, state, and statistics for the specified T1/E1 carrier. Refer to *T1/E1 statistics and performance measures* on page 20 for more information.

TxT1E1ChannelStatus

Queries the current signaling status of all channels on a carrier.

Prototype

S16 **TxT1E1ChannelStatus** (S16 ***handle***, U8 ***carrier***, U32 ****pTransparent***, U32 ****pRxABCD***, U32 ****pTxABCD***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
pTransparent	Pointer to U32 where the transparency bit map is returned to the caller. A bit set to 1 indicates the corresponding channel is transparent; LSB = channel 1 (only 24 least significant bits are used). This field has meaning only for T1/J1 carriers.

pRxABCD	Pointer to the array of four U32s for the current values of the received ABCD signaling bits for each channel (only 24 least significant bits are used for T1/J1):
	`RxABCD[0] = A bits (LSB = chan 1,MSB = chan 32)`
	`RxABCD[1] = B bits ""`
	`RxABCD[2] = C bits ""`
	`RxABCD[3] = D bits ""`

pTxABCD	Pointer to the array of four U32s for the current values of the transmitted ABCD signaling bits for each channel (only 24 least significant bits are used for T1/J1):
	`TxABCD[0] = A bits (LSB = chan 1,MSB = chan 32)`
	`TxABCD[1] = B bits  ""`
	`TxABCD[2] = C bits  ""`
	`TxABCD[3] = D bits  ""`

Return values

Return value	Description
T1E1_SUCCESS	
T1E1_DEVICE_ERROR	Invalid adapter found.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

TxT1E1ChannelStatus displays the signaling bits and transparency attributes of all channels on the specified T1/E1 carrier.

TxT1E1Close

Closes the T1/E1 manager for a TX board and makes it available to other processes.

Prototype

void **TxT1E1Close** (S16 ***handle***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.

Return values

Void.

TxT1E1ConditionChan

Selects a specific bit pattern to continuously transmit on one or all channels of a carrier.

Prototype

S16 **TxT1E1ConditionChan** (S16 ***handle***, U8 ***carrier***, U8 ***channel***, U8 ***control***, U8 ***pattern***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
channel	Channel number to be conditioned. Valid values: 0 through 31 0xFF for all
control	Whether to start or stop conditioning. Valid values: T1E1_COND_OFF T1E1_COND_ON
pattern	Bit pattern to transmit.

Return values

Return value	Description
T1E1_SUCCESS	
T1E1_DEVICE_ERROR	Invalid adapter found.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_INVALID_TIMESLOT	Invalid channel number.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

TxT1E1ConfigCarrier

Configures the basic operating parameters for a T1/E1 carrier, including framing type, line coding, and line build out.

Prototype

S16 **TxT1E1ConfigCarrier** (S16 ***handle***, U8 ***carrier***, TxT1E1Config ****pConfig***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
pConfig	Pointer to the configuration block for this carrier.

Return values

Return value	Description
T1E1_SUCCESS	
T1E1_DEVICE_ERROR	Invalid adapter found.
T1E1_INVALID_BLDOUT	Line buildout is out of range.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_CODING	Invalid encoding scheme.
T1E1_INVALID_FRAMING	Invalid framing value.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_INVALID_TIMESLOT	Invalid channel number.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

A TDM configuration file automatically performs initial configuration as part of *ss7load*. Use **TxT1E1ConfigCarrier** to override the information in the TDM configuration file.

TxT1E1ConfigChan

Forces a specific channel in or out of transparent mode.

Prototype

S16 **TxT1E1ConfigChan** (S16 ***handle***, U8 ***carrier***, U8 ***channel***, U8 ***transparent***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
channel	Channel number to be conditioned. Valid values: 0 through 31 0xFF for all
transparent	Whether to enable or disable transparency. Valid values: T1E1_CHAN_NORMAL T1E1_CHAN_TRANSPARENT

Return values

Return value	Description
T1E1_SUCCESS	
T1E1_DEVICE_ERROR	Invalid adapter found.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_INVALID_TIMESLOT	Invalid channel number.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

TxT1E1ConfigChan overrides the robbed bit signaling or bit 7 stuffing enable for the specified carrier. This function creates one or more clear channels with no robbed bit signaling or bit 7 stuffing when these features are enabled on the carrier.

TxT1E1CtrlCarrier

Puts the carrier into loopback mode or forces it to transmit an alarm indication.

Prototype

S16 **TxT1E1CtrlCarrier** (S16 ***handle***, U8 ***carrier***, U8 ***loopback***, U8 ***alarmState***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
loopback	Loopback mode. Refer to Details for valid values.
alarmState	Forces the TX board to start generating or stop generating an alarm on this carrier. Refer to Details for valid values.

Return values

Return value	Description
T1E1_DEVICE_ERROR	Invalid adapter.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_INVALID_LOOPBACK	Invalid loopback mode.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

The following table lists the valid values for ***loopback***:

Value	Description
LOOPBACK_NONE	Normal operating mode (no loopback).
LOOPBACK_LOCAL	Local data transmitted is looped back to the receive side.
LOOPBACK_REMOTE	Data received from the line is transmitted back onto the line.

The following table lists the valid values for ***alarmState***:

Value	Description
NO_ALARM	Stop generating all alarms.
YELLOW_ALARM	Generate yellow (or distant/far end) alarm.
BLUE_ALARM	Generate blue (also called AIS) alarm.
TS16AIS_ALARM	Generate timeslot 16 alarm indication signal (E1 only).
DISTANTMF_ALARM	Generate distant multiframe alarm (E1 only).

Refer to *T1/E1 carrier alarms* on page 26 for more information.

TxT1E1Open

Opens the T1/E1 manager on the specified TX board.

Prototype

S16 **TxT1E1Open** (TXBOARD_T ***board***, U16 ***swint***)

Argument	Description
board	TX board number.
swint	Not used.

Return values

Return value	Description
T1E1_BAD_BOARD	Invalid board number.
T1E1_NO_DRIVER	TX driver not found.
T1E1_OPEN_FAIL	Attempt to open the T1/E1 manager failed.

Details

TxT1E1Open returns the specified TX board handle to be used in subsequent T1/E1 function requests. This function must be called before any other T1/E1 functions are called for a particular board.

TxT1E1PerfReport

Generates an on-demand performance report.

Prototype

S16 **TxT1E1PerfReport** (S16 ***handle***, U8 ***carrier***, U8 ***flags***, U16 ***pCurrInterval***, TxT1E1Stats ***pCurrStats***, U16 ***pValidInterval***, TxT1E1Stats ****pIntervalStats***, TxT1E1Stats ****pSummaryStats***, U16 ***pPvcs***, U16 ***pLvcs***, U16 ***pSlips***, U16 ***pEbits***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
flags	Status request options flags. The only option is T1E1_RESET_COUNTS, which resets statistics accumulators after returning values.
pCurrInterval	Number of seconds in the current statistics interval returned to the caller.
pCurrStats	Current 15 minute interval performance statistics returned to the caller.
pValidInterval	Number of 15 minute intervals collected in summary statistics (always 96 except during first 24 hours after a boot, when it can be less than 96).
pIntervalStats	Pointer to the caller memory where interval statistics are returned.
pSummaryStats	Pointer to the caller memory where 24 hour summary statistics are returned.
pPcvs	Current value of the path code violations counter (accumulated since last reset) returned to the caller.
pLcvs	Current value of the line code violations counter (accumulated since last reset) returned to the caller.
pSlips	Current value of the controlled slips counter (accumulated since last reset) returned to the caller.
pEbits	Current value of the E-bit (E1) counter (accumulated since last reset) returned to the caller.

Return values

Return value	Description
T1E1_DEVICE_ERROR	Invalid adapter.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

TxT1E1PerfReport generates a report that includes performance statistics for the current 15 minute interval, the last 24 hours worth of 15 minute intervals, and the 24 hour summary. Refer to *T1/E1 statistics and performance measures* on page 20 for more information.

TxT1E1SuperviseCarrier

Registers the application to receive unsolicited notifications of carrier alarm conditions, carrier synchronization state changes, and automatic 15 minute and 24 hour performance statistics.

Prototype

S16 **TxT1E1SuperviseCarrier** (S16 ***handle***, U8 ***carrier***, U8 ***notifMask***)

Argument	Description
handle	Handle returned from a previous call to **TxT1E1Open** for the desired board.
carrier	Network interface (carrier trunk number). Valid values: NET_T1A through NET_T1D for PCI and PCI Express TX boards NET_T1A through NET_T1H for CompactPCI TX boards
notifMask	Specifies the events that are notified. Refer to Details for a list of acceptable values.

Return values

Return value	Description
T1E1_SUCCESS	
T1E1_DEVICE_ERROR	Invalid adapter.
T1E1_INVALID_CARRIER	Invalid network interface.
T1E1_INVALID_HANDLE	Invalid handle.
T1E1_NO_RESOURCES	T1/E1 adapter is not present on the communications processor.

Details

The following table lists the acceptable values for the ***notifMask*** parameter. Use OR with these values to request multiple events.

Value	Registers application to receive...
T1E1_REG_STATUS	Status changes.
T1E1_REG_ALARMS	Alarm notifications.
T1E1_REG_15MIN_REPORT	15 minute report.
T1E1_REG_24HR_REPORT	24 hour report.

Refer to *Handling unsolicited T1/E1 notifications* on page 22 for more information.

8 TX SWI function reference

Using the function reference

This section provides an alphabetical reference to the TX SWI functions. A typical function includes:

Prototype	The prototype is shown followed by a listing of the function arguments. Data types include: • U8 (8-bit unsigned) • S8 (8-bit signed) • U16 (16-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • S32 (32-bit signed) If a function argument is a data structure, the complete data structure is defined. **Note:** Some parameters are not applicable to both ANSI and ITU-T (CCITT) networks.
Return values	The return value for a function is either SUCCESS or an error code. SUCCESS (zero) indicates the function was initiated. Subsequent events indicate the status of the operation.

Function summary

All TX SWI functions are asynchronous.

Outbound and inbound message functions

Function	Description
txswiAddress	Zero-fills a CPI packet to be used as a TX SWI request message and sets all addressing information.
txswiReply	Verifies the reply message and decodes the message if required.

Initialization functions

Function	Description
txswiResetSwitch	Builds a **swiResetSwitch** message to reset the switch block to the idle state.

Clocking control functions

Function	Description
txswiConfigBoardClock	Builds a **swiConfigBoardClock** message to establish the clock source for a board.
txswiConfigNetrefClock	Builds a **swiConfigNetrefClock** message to define the source of the NETREF clocks.
txswiConfigSec8KClock	Builds a **swiConfigSec8KClock** message to define the source of the secondary 8 KHz clock on the bus.
txswiGetBoardClock	Builds a **swiGetBoardClock** message to retrieve the board clocking configuration and the current status of the clocks.

Board functions

Function	Description
txswiConfigLocalStream	Builds a **swiConfigLocalStream** message to control the board's T1/E1 trunks.
txswiConfigLocalTimeslot	Builds a **swiConfigLocalTimeslot** message to configure local stream and timeslot-specific attributes.
txswiGetBoardInfo	Builds a **swiGetBoardInfo** message to return board description information.
txswiGetLocalStreamInfo	Builds a **swiGetLocalStreamInfo** message to return information about the specified local stream.

Status functions

Function	Description
txswiGetSwitchCaps	Builds a **swiGetSwitchCaps** message to return the capabilities of the device driver and the switch controlled by it.
txswiGetTimingReference	Builds a **swiGetTimingReference** message to retrieve the status of a potential timing reference.

Switched connection control functions

Function	Description
txswiMakeConnection	Builds a **swiMakeConnection** message to connect inputs to outputs.
txswiSendPattern	Builds a **swiSendPattern** message to send a fixed pattern on the specified switch block outputs.
txswiDisableOutput	Builds a **swiDisableOutput** message to reset the specified switch block outputs to their idle state.
txswiSampleInput	Builds a **swiSampleInput** message to retrieve the current data values present on the specified switch block inputs.
txswiGetOutputState	Builds a **swiGetOutputState** message to retrieve the state of the specified switch block outputs.

txswiAddress

Zero-fills a CPI packet to be used as a TX SWI request message and sets all addressing information.

Prototype

void **txswiAddress** (CPIPKT ***pkt**, U32 ***req_size***, U16 ***board***, U16 ***host_chan***)

Argument	Description
pkt	Pointer to the CPI packet (of ***req_size***) to be formatted into a TX SWI request message.
req_size	Size of the CPI packet.
board	Destination board number.
host_chan	Source CPI channel number.

Details

Call **txswiAddress** after determining the required size of the TX SWI request message to be passed to the TX board. All other TX SWI functions except **txswiReply** return the required request packet size when a NULL CPI packet pointer is provided.

Refer to *Sending and receiving requests* on page 28 for more information.

txswiConfigBoardClock

Builds a **swiConfigBoardClock** message to establish the clock source for a board. For more information, refer to **swiConfigBoardClock** in the *Switching Service Developer's Reference Manual*. Refer to the *NMS OAM System User's Manual* for information about clock fallback configurations.

Prototype

U32 **txswiConfigBoardClock** (CPIPKT ***pkt***, SWI_CLOCK_ARGS ***args***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
args	Pointer to the clock arguments.

Return value

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiConfigLocalStream

Builds a **swiConfigLocalStream** message to control the board's T1/E1 trunks.

Prototype

U32 **txswiConfigLocalStream** (CPIPKT ***pkt***, SWI_LOCALSTREAM_ARGS ***args***, void ***buffer***, unsigned ***size***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
args	Pointer to local stream arguments. Refer to Details for more information.
buffer	Pointer to stream-specific information.
size	Size of buffer (in bytes).

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

Details

The following table lists the settings for the ***args*** parameter used to trigger the T1/E1 trunk commands:

deviceid (MVIP95_...)	Parameterid (TX_X_...)	Buffer	Description
E1_TRUNK_DEVICE T1_TRUNK_DEVICE X_J1_TRUNK_DEVICE	CONFIG_CARRIER	TxT1E1ConfigEx	Configure and enable a trunk line (as type indicated by ***deviceid***).
X_FRAMER_DEVICE	CMD_CARRIER	TxT1E1Cmd	Perform a command on the T1/E1 carrier. Refer to the following command table.

The following table lists the CMD_CARRIER commands:

Command	Use this command to...
T1E1_X_CMD_ENABLE	Enable the framer.
T1E1_X_CMD_DISABLE	Disable the framer.
T1E1_X_CMD_RESYNC	Force framer resynchronization.
T1E1_X_CMD_AIS	Transmit or stop transmitting an alarm indication signal. Flags are: T1E1_X_AIS_LINE T1E1_X_AIS_SYS
T1E1_X_CMD_RA	Transmit or stop transmitting remote alarms. Flags are: T1E1_X_RA_SEND
T1E1_X_CMD_LLBDOWN	Transmit or stop transmitting a line loopback deactivate code. Flags are: T1E1_X_LLB_SEND T1E1_X_LLB_FIXED T1E1_X_LLB_UNFRAMED llb_code llb_bits
T1E1_X_CMD_LLBUP	Transmit or stop transmitting a line loopback activate code. Flags are: T1E1_X_LLB_SEND T1E1_X_LLB_FIXED T1E1_X_LLB_UNFRAMED llb_code llb_bits
T1E1_X_CMD_LOOP	Place the entire framer in loop mode or exit loop mode. Flags are: T1E1_X_LOOP_JITTER T1E1_X_LOOP_AIS T1E1_X_LOOP_TRANS loop_mode
T1E1_X_CMD_CHANLOOP	Place a single framer channel in loop mode or disable channel loop mode. Flags are: T1E1_X_CHANLOOP_OFF chanloop_channel
T1E1_X_CMD_PRBS	Transmit a pseudo random bit sequence. Flags are: T1E1_X_PRBS_SEND T1E1_X_PRBS_INVERT T1E1_X_PRBS_GEN20 T1E1_X_PRBS_FRAMED T1E1_X_PRBS_MONITOR
T1E1_X_CMD_DEFECT	Insert single bit defects. Flags are: T1E1_X_DEFECT_FAS T1E1_X_DEFECT_MULTI T1E1_X_DEFECT_CRC T1E1_X_DEFECT_CAS T1E1_X_DEFECT_PRBS T1E1_X_DEFECT_BIPOLAR
T1E1_X_CMD_SIM	Initiate or stop alarm simulation, or advance to the next test. Flag is: T1E1_X_STOP_SIM

For more information, refer to **swiConfigLocalStream** in the *Switching Service Developer's Reference Manual*.

txswiConfigLocalTimeslot

Builds a **swiConfigLocalTimeslot** message to configure local stream and timeslot-specific attributes.

Prototype

U32 **txswiConfigLocalTimeslot** (CPIPKT ***pkt**, SWI_LOCALTIMESLOT_ARGS ***args**, void ***buffer**, unsigned ***size**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
args	Pointer to local stream arguments. Refer to Details for more information.
buffer	Pointer to timeslot-specific information.
size	Size of buffer (in bytes).

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

Details

The following table lists the settings for the ***args*** parameter used to trigger the commands:

deviceid (MVIP95_...)	Parameterid (TX_X_...)	Buffer	Description
X_COMMPROC_DEVICE	CONFIG_PORT	SWIX_LOCAL_TIMESLOT_CONFIG	Define a port.
X_COMMPROC_DEVICE	CONFIG_SPEED	SWIX_LOCAL_TIMESLOT_CONFIG	Set 56 Kb and 48 Kb.

For more information, refer to **swiConfigLocalTimeslot** in the *Switching Service Developer's Reference Manual*.

txswiConfigNetrefClock

Builds a **swiConfigNetrefClock** message to define the source of the NETREF clocks. For more information, refer to **swiConfigNetrefClock** in the *Switching Service Developer's Reference Manual*.

Prototype

U32 **txswiConfigNetrefClock** (CPIPKT ***pkt***, SWI_NETREF_CLOCK_ARGS ***args***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
args	Pointer to clock arguments.

Return value

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiConfigSec8KClock

Builds a **swiConfigSec8KClock** message to define the source of the secondary 8 KHz clock on the bus. For more information, refer to **swiConfigSec8KClock** in the *Switching Service Developer's Reference Manual*.

Prototype

U32 **txswiConfigSec8KClock** (CPIPKT ***pkt***, DWORD ***source***, DWORD ***network***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
source	Source of the clock signal (MVIP95_SOURCE_***xxx***).
network	Network number when the source = MVIP95_SOURCE_NETWORK.

Return value

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiDisableOutput

Builds a **swiDisableOutput** message to reset the specified switch block outputs to their idle state. For more information, refer to **swiDisableOutput** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiDisableOutput** (CPIPKT ***pkt***, SWI_TERMINUS ***output[]***, unsigned ***count***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
output	Array of output terminus descriptors.
count	Number of output connections to disable.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

txswiGetBoardClock

Builds a **swiGetBoardClock** message to retrieve the board clocking configuration and the current status of the clocks. For more information, refer to **swiGetBoardClock** in the *Switching Service Developer's Reference Manual*.

Prototype

U32 **txswiGetBoardClock** (CPIPKT ***pkt**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.

Return value

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiGetBoardInfo

Builds a **swiGetBoardInfo** message to return board description information. For more information, refer to **swiGetBoardInfo** in the *Switching Service Developer's Reference Manual*.

Prototype

U32 **txswiGetBoardInfo** (CPIPKT ***pkt**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.

Return value

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiGetLocalStreamInfo

Builds a **swiGetLocalStreamInfo** message to return information about the specified local stream.

Prototype

U32 **txswiGetLocalStreamInfo** (CPIPKT ***pkt**, SWI_LOCALSTREAM_ARGS ***args**, void ***buffer**, unsigned ***size**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
args	Pointer to local stream arguments. Refer to Details for more information.
buffer	Pointer to stream-specific information.
size	Size of buffer (in bytes).

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

Details

The following table lists the settings for the ***args*** parameter used to trigger the commands:

deviceid (MVIP95_...)	Parameterid (TX_X_...)	Buffer	Description
X_FRAMER_DEVICE	CARRIER_STATUS	TxT1E1_CarrierStatus	Get status of the T1/E1 trunk.
X_FRAMER_DEVICE	PERF_REPORT	TxT1E1_PerfReport	Get a performance report.
X_FRAMER_DEVICE	CHANNEL_STATUS	TxT1E1_ChannelStatus	Get channel status.
X_COMMPROC_DEVICE	PORT_COMM_CONFIG	SWIX_COMM_PORT_ALL	Get the configuration of all ports.

For more information, refer to **swiGetLocalStreamInfo** in the *Switching Service Developer's Reference Manual*.

txswiGetOutputState

Builds a **swiGetOutputState** message to retrieve the state of the specified switch block outputs. For more information, refer to **swiGetOutputState** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiGetOutputState** (CPIPKT ***pkt***, SWI_TERMINUS ***output[]***, unsigned ***count***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
output	Array of output terminus descriptors.
count	Number of output connections for which to return state information.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

txswiGetSwitchCaps

Builds a **swiGetSwitchCaps** message to return the capabilities of the device driver and the switch controlled by it. For more information, refer to **swiGetSwitchCaps** in the *Switching Service Developer's Reference Manual*.

Prototype

U32 **txswiGetSwitchCaps** (CPIPKT ***pkt***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiGetTimingReference

Builds a **swiGetTimingReference** message to retrieve the status of a potential timing reference. For more information, refer to **swiGetTimingReference** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiGetTimingReference** (CPIPKT ***pkt**, DWORD ***referencesource***, DWORD ***network***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
referencesource	Clock reference to query.
network	Device source for the CT bus clock signals.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

txswiMakeConnection

Builds a **swiMakeConnection** message to connect inputs to outputs. For more information, refer to **swiMakeConnection** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiMakeConnection** (CPIPKT ***pkt**, SWI_TERMINUS ***input[]**, SWI_TERMINUS ***output[]**, unsigned ***count**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
input	Array of input terminus descriptors.
output	Array of output terminus descriptors.
count	Number of connections to make.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

txswiResetSwitch

Builds a **swiResetSwitch** message to reset the switch block to the idle state. For more information, refer to **swiResetSwitch** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiResetSwitch** (CPIPKT ***pkt**)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.

txswiReply

Verifies the reply message and decodes the message if required.

Prototype

TXSWI_MSG **txswiReply** (CPIPKT ***pkt**, S16 ***len**, U32 ***ccode**)

Argument	Description
pkt	Pointer to the CPI packet (of ***len***) to be formatted into a TX SWI request message.
len	Byte length of the CPI packet to be decoded.
ccode	Pointer to the location where the completion code is stored.

Return value

Return value	Description
Message pointer	Pointer to the received and decoded TX SWI message. NULL if invalid ***pkt***.

Details

All TX SWI request messages complete asynchronously. A separate reply message is received when the TX board completes processing the corresponding request message. When a TX SWI reply message is received, the application calls **txswiReply** to have the reply verified and converted to the host native format. After calling **txswiReply**, the returned message pointer holds the information provided by the TX board. The TXSWI_MSG structure is a union of all message types provided by the TX SWI library. To obtain the correct reply information, reference the proper response sub-structure. For example, to obtain the response from a **txswiGetSwitchCaps** request, reference the message as:

```
rxmsg->response.get_switch_caps…
```

Refer to *Sending and receiving requests* on page 28 for more information.

txswiSampleInput

Builds a **swiSampleInput** message to retrieve the current data values present on the specified switch block inputs. For more information, refer to **swiSampleInput** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiSampleInput** (CPIPKT ***pkt***, SWI_TERMINUS ***input[]***, unsigned ***count***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
input	Array of input terminus descriptors.
count	Number of inputs to be sampled.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

txswiSendPattern

Builds a **swiSendPattern** message to send a fixed pattern on the specified switch block outputs. For more information, refer to **swiSendPattern** in the *Switching Service Developer's Reference Manual.*

Prototype

U32 **txswiSendPattern** (CPIPKT ***pkt***, BYTE ***pattern[]***, SWI_TERMINUS ***output[]***, unsigned ***count***)

Argument	Description
pkt	Pointer to the CPI packet to be formatted into a TX SWI request message. NULL = Return the byte length of the CPI packet required to send the request.
pattern	Array of patterns to output.
output	Array of output terminus descriptors.
count	Number of output patterns to define.

Return values

Return value	Description
SUCCESS	If ***pkt*** is NULL, the size of the required CPI packet is returned. If ***pkt*** is not NULL, 0 is returned to indicate a successful request, or an error code is returned if the request failed.
MVIP95_ERR_INVALID_PARAMETER	Invalid count.

9 Demonstration programs

Demonstration programs summary

The following programs are provided in source and executable form to demonstrate how to use the TDM libraries:

Program	Demonstrates how to...
t1demo	Use the T1/E1 and H.100/H.110 library functions and test the effect of the functions on other TX boards in a system.
t1stat	Receive unsolicited T1/E1/J1 status messages and performance reports.
txdynamic	Dynamically switch SS7 links across TDM channels without rebooting the TX boards.
txsdemo	Use the TX SWI library. Use this program as a starting point to control switching on a TX board.

TDM demonstration program: t1demo

Provides a command line interface to call the T1/E1 and H.100/H.110 library functions and test the effect of the functions on other TX boards in a system.

Usage

```
t1demo -b board
```

where ***board*** specifies the TX board number to connect to. The default is board 1.

Description

t1demo provides the following commands:

Command	Description
CL ***net frame coding bldout robbed timing***	Configures the T1/E1 line. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***frame*** must be: NONE D4 ESF CCS CAS CCSCRC4 CASCRC4 ***coding*** must be: NONE NOZCS B7S B8ZS HDB3 ***bldout*** must be from 0 through 4. ***robbed*** must be: DIS (disabled) ENA (enabled) ***timing*** must be: SLAVE MASTER
LS ***net flags***	Displays the T1/E1 line status. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***flags*** must be: 0 1 (reset the counts)

Command	Description
PR ***net flags***	Displays a T1/E1 performance report. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***flags*** must be: 0 1 (reset the counts)
LC ***net loopback alarm***	Sets the T1/E1 line control. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***loopback*** must be: NONE LOCAL REM (remote) ***alarm*** must be: NONE YELLOW BLUE TS16AIS DISTMF
CD ***net channel state pattern***	Conditions a T1/E1 channel. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***channel*** must be 0 through 31. ***state*** must be: DIS (disabled) ENA (enabled) ***pattern*** must be: 0 through 255 0x00 through 0xFF
CC ***net channel trans***	Configures a T1/E1 channel. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***channel*** must be 0 through 31. ***trans*** must be: 0 (normal) 1 (transparent)
CS ***net***	Displays the T1/E1 channel status. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards

Command	Description
SV ***net mask***	Starts carrier supervision. ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards ***mask*** can be any combination of the following: 1 (status) 2 (alarms) 3 (15 minute report) 8 (24 hour report)
CK ***clock sclock net***	Configures the H.100/H.110 clock. ***clock*** must be: SEC8K OSC NET ***sclock*** must be: SLAVE OSC NET ***net*** must be: A, B, C, or D for PCI and PCI Express TX boards A, B, C, D, E, F, G, or H for CompactPCI TX boards
QO ***stream timeslot***	Queries the timeslot output configuration. ***stream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***timeslot*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces
QS	Displays the switch capabilities.
RS	Resets the switch fabric.
SI ***stream timeslot***	Displays the last input sample on a stream and timeslot. ***stream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***timeslot*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces

Command	Description
SO ***ostream otime mode istream itime pat***	Makes or clears the switch simplex connection. ***ostream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***otime*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces ***mode*** must be: CONN (connect) DIS (disable) PAT (pattern) ***istream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***itime*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces ***pat*** must be: 0 through 255 0x00 through 0xFF
SF ***ostream otime mode istream itime pat***	Makes or clears the switch duplex connection. ***ostream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***otime*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces ***mode*** must be: CONN (connect) DIS (disable) PAT (pattern) ***istream*** must be: 0 through 31 for H.100/H.110 80 - 83 for T1/E1 trunks 1 - 4 84 - 87 for T1/E1 trunks 5 - 8 (CompactPCI TX only) ***itime*** must be: 0 through 127 for H.100/H.110 0 through 23 for T1 or J1 interfaces 0 through 31 for E1 interfaces ***pat*** must be: 0 through 255 0x00 through 0xFF

Command	Description
TR ***mode***	Tristates or enables the switch. ***mode*** must be: DIS (disabled) ENA (enabled)
Q	Ends the test application.

T1/E1 line status demonstration program: t1stat

Registers with the TX device driver and displays any T1/E1 status messages or performance reports received.

Usage

```
t1stat [options]
```

where ***options*** are:

Option	Description
-c ***chan***	Specifies an alternate host-side CPI channel.
-r	Registers for T1/E1 monitoring (all boards, full monitoring).
-s	Displays last reported state banner after each standard message.
-q	Quiets line status messages, including banner information.

Description

t1stat waits indefinitely and displays any T1/E1 status messages or performance reports received.

TX SWI library demonstration program: txsdemo

Demonstrates how to use the TX SWI library to control switching on a TX board. *txsdemo* tests communications between two TX boards that are connected with the H.100/H.110 bus or with T1/E1/J1 cables.

Usage

```
txsdemo [options]
```

where ***options*** are:

Option	Description
-m	Sets the TX board number of the board that starts as clock master. Default: CP 1.
-s	Sets the TX board number of the board that starts as clock slave. Default: CP 2.
-p	Sets the number of test passes. Default: Infinite.
-c	Sets the host-side channel number. Default: 3.
-t	Performs a T1 test between the two TX boards. Default: Test H.100/H.110 connectivity.
-e	Performs an E1 test between the two TX boards. Default: Test H.100/H.110 connectivity.
-j	Performs a J1 test between the two TX boards. Default: Test H.100/H.110 connectivity.

Description

txsdemo tests connectivity between two TX boards identified as the initial master and slave boards. The master board begins the test as the board mastering the clock (H.100/H.110 A_CLOCK, or loop master for T1/E1/J1 test). After a test pass completes with the initial master/slave clocking, the master becomes the clock slave for the next pass.

The following table lists the stages of the connectivity test:

Stage	Description
1	CAPS. Retrieving the switching capabilities of the TX boards.
2	CLOCK. Configuring clocking on the TX boards.
3	CONFIG. Configuring T1/E1/J1 trunks. This step is skipped if testing over the H.100/H.110 bus.
4	CONNECT. Making connections on the receiver board.
5	PATTERN. Configuring output patterns on the transmitter board.
6	SAMPLE. Collecting samples on the receiver board.
7	BREAK. Breaking all connections defined for the given test pass.
8	ADVANCE. Testing advance stream or trunk. If testing T1/E1/J1 connections, go to CLOCK phase. If testing H.100/H.110 connections, go to CONNECT phase.

For the H.100/H.110 bus, the test begins with the master board transmitting patterns to the odd stream numbers. The slave board connects odd stream inputs to even stream outputs. After all connections are verified, the master board is changed to transmit patterns to the even stream numbers (with the slave board connecting even stream inputs to odd stream outputs).

Although *txsdemo* tests TX board connectivity, its main purpose is to demonstrate the use of the TX SWI library. Use the example source code as a starting point for other TDM switching control applications.

When *txsdemo* executes, a progress bar displays the current test phase:

```
----- H.100/H.110 Streams ------ | T1/E1/J1
          1         2         3  | Framers
01234567890123456789012345678901 | 12345678
================================ | ========
################################ | ########
```

where **#** is the testing phase for the given H.100/H.110 stream or T1/E1/J1 trunk and is one of the following values:

Value	Description
_	Stream or trunk has not been tested.
+	Making connections across a stream or trunk.
*	Sampling for expected patterns across a stream or trunk.
x	Breaking connections across a stream or trunk.
.	Pattern test PASSED for a stream or trunk.
F	Pattern test FAILED for a stream or trunk.

txsdemo also supports the following single-character user commands, which can be entered from the keyboard:

Command	Description
k	Displays the progress bar key.
s	Shows all pattern checking statistics.
r	Resets all pattern checking statistics (zeros all statistics).
?	Shows the set of supported keyboard commands.
q	Quits *txsdemo.*

Dynamic SS7 link switching demonstration program: txdynamic

Demonstrates how to use the TX SWI library to dynamically switch an SS7 link between two TX boards across the H.100/H.110 bus, and optionally across T1/E1/J1 connections.

txdynamic also registers for alarms from the two TX boards and uses the receipt of the MTP LINK_UP alarm to indicate when a link is successfully switched to a new channel.

Usage

```
txdynamic options[]
```

where ***options*** are:

Option	Description
-m	Sets the master TX board (the first board to switch). Default: CP 1.
-s	Sets the slave TX board (the second board to switch). Default: CP 2.
-p	Sets the number of test passes. Default: Infinite.
-c	Sets the host side channel number. Default: Channel 3.
-a	Shows all received alarms. Default: Only show LINK_UP and LINK_DOWN.
-v	Shows alarm severity. Default: Not shown.
-t	Performs a T1 test between the TX boards. Default: No T1 test.
-e	Performs an E1 test between the TX boards. Default: No E1 test.
-j	Performs a J1 test between the TX boards. Default: No J1 test.

Description

txdynamic dynamically switches an SS7 link across TDM channels without rebooting the TX boards. The two TX boards are provided as slave and master boards. The master board link connection is switched first. The slave board is switched after both boards report a link failure.

Before running *txdynamic*, perform an *ss7load* of the master and slave TX boards using the configuration files (*txcfg**x**.txt*) provided with this demonstration program. For more information on running *ss7load*, refer to the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual* .

Note: To test switching over T1/E1/J1 connections in addition to testing the H.100/H.110 bus, uncomment the appropriate trunk configuration entries from the configuration files before loading the TX boards. Then specify the trunk type by entering the -t, -e, or -j option when you invoke *txdynamic*.

The test assumes that each trunk of the master board is connected to the same trunk number of the slave board.

The following table lists the stages of the dynamic switching demonstration:

Stage	Description
1	CAPS. Retrieving the switching capabilities of both TX boards.
2	REG_ALARMS. Registering to receive alarms from both boards.
3	WAIT_CONNECT. Waiting for alarms indicating link up (initially connected).
4	MOVE_MASTER. Moving master board's TDM connection.
5	WAIT_DOWN. Waiting for both boards to send an alarm that the link is down.
6	MOVE_SLAVE. Moving slave board's TDM connection.
7	WAIT_UP. Waiting for both boards to alarm link up.
8	ADVANCE. Advancing stream or trunk under test.

For the H.100/H.110 bus, the test uses timeslot 1 on each H.100/H.110 stream, advancing the stream number by two with each pass to maintain the standard even/odd convention. For T1/E1/J1 trunks, the test traverses all available trunks using timeslot 1.

The following example shows the output generated by *txdynamic*:

```
************************************
* DYNAMIC SS7 SWITCHING DEMONSTRATION *
************************************
GETTING TX board switching capabilities...
WAITING for initial connection alarms...
ALARM: <03/30/04 09:41:01> mtp       2  18179 MTP3 Link 0 Up
ALARM: <03/30/04 09:41:02> mtp       1  18179 MTP3 Link 0 Up
SWITCHING board 1's port 1 to H.100/H.110 streams 2 and 3...
ALARM: <03/30/04 09:41:21> mtp       2  18180 MTP3 Link 0 Down
ALARM: <03/30/04 09:41:21> mtp       1  18180 MTP3 Link 0 Down
SWITCHING board 2's port 1 to H.100/H.110 streams 2 and 3...
ALARM: <03/30/04 09:41:22> mtp       1  18179 MTP3 Link 0 Up
ALARM: <03/30/04 09:41:22> mtp       2  18179 MTP3 Link 0 Up
```

txdynamic retrieves the switching capabilities of both TX boards. It then waits for the initial LINK_UP alarms from the boards. Once both boards report that the SS7 MTP link is up, the master board's TDM definition for the link is moved to the next stream pair.

txdynamic then waits for the boards to send an alarm that the link is down (since one half of the link is no longer defined as the same channel in use by the other half). Next, *txdynamic* changes the TDM definition for the slave board's link and waits until the boards send an alarm that the link is up. This is the final confirmation that the link is successfully switched. This procedure continues until each H.100/H.110 stream is visited. After the link is successfully switched to streams 30 and 31, *txdynamic* reports that the test is complete (if a number of passes was specified), returns to testing the H.100/H.110 stream 0 and 1 pair (if no -t|e|j is specified on the command line), or switches the link over each T1|E1|J1 trunk.

10 TX board switching

Switch blocking

TX boards control TDM switching using a Time Slot Interchanger (TSI) providing local bus to local bus switching in a full non-blocking way.

Basic TX switching capacity

All TX boards that provide TDM connectivity offer a large number of concurrent connections. The number of H.100/H.110 connections is limited to a maximum of 128 full duplex or 256 simplex (or half duplex) connections, in any combination, from either:

- H.100/H.110 bus to the local bus
- H.100/H.110 bus to H.100/H.110 bus

Extended TX switching capabilities

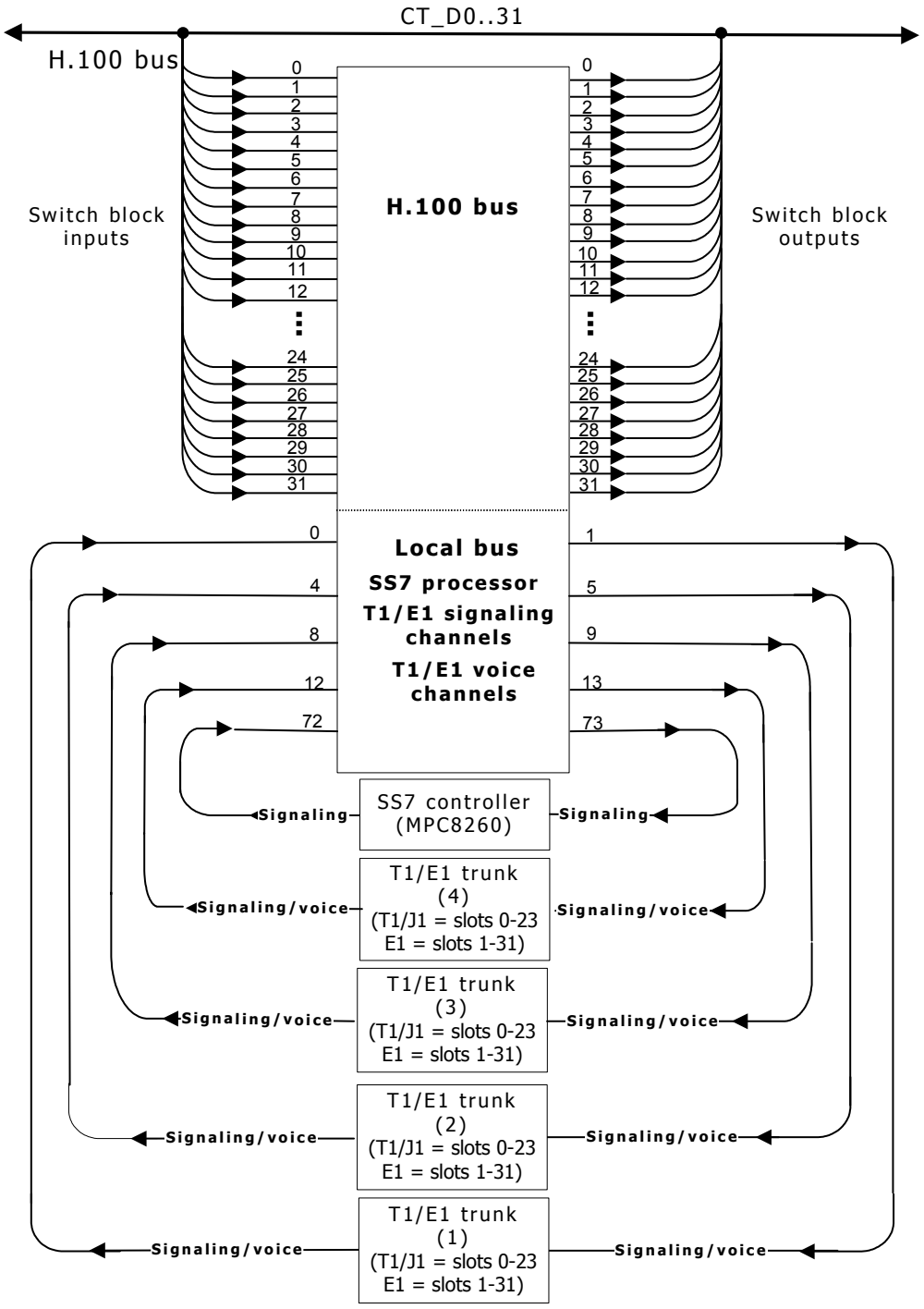
PCI Express and CompactPCI TX boards can maintain additional bus switching connections.

The number of H.110 connections is limited to a maximum of 256 full duplex or 512 simplex (or half duplex) connections, in any combination, from either:

- H.100/H.110 bus to the local bus
- H.100/H.110 bus to H.110 bus

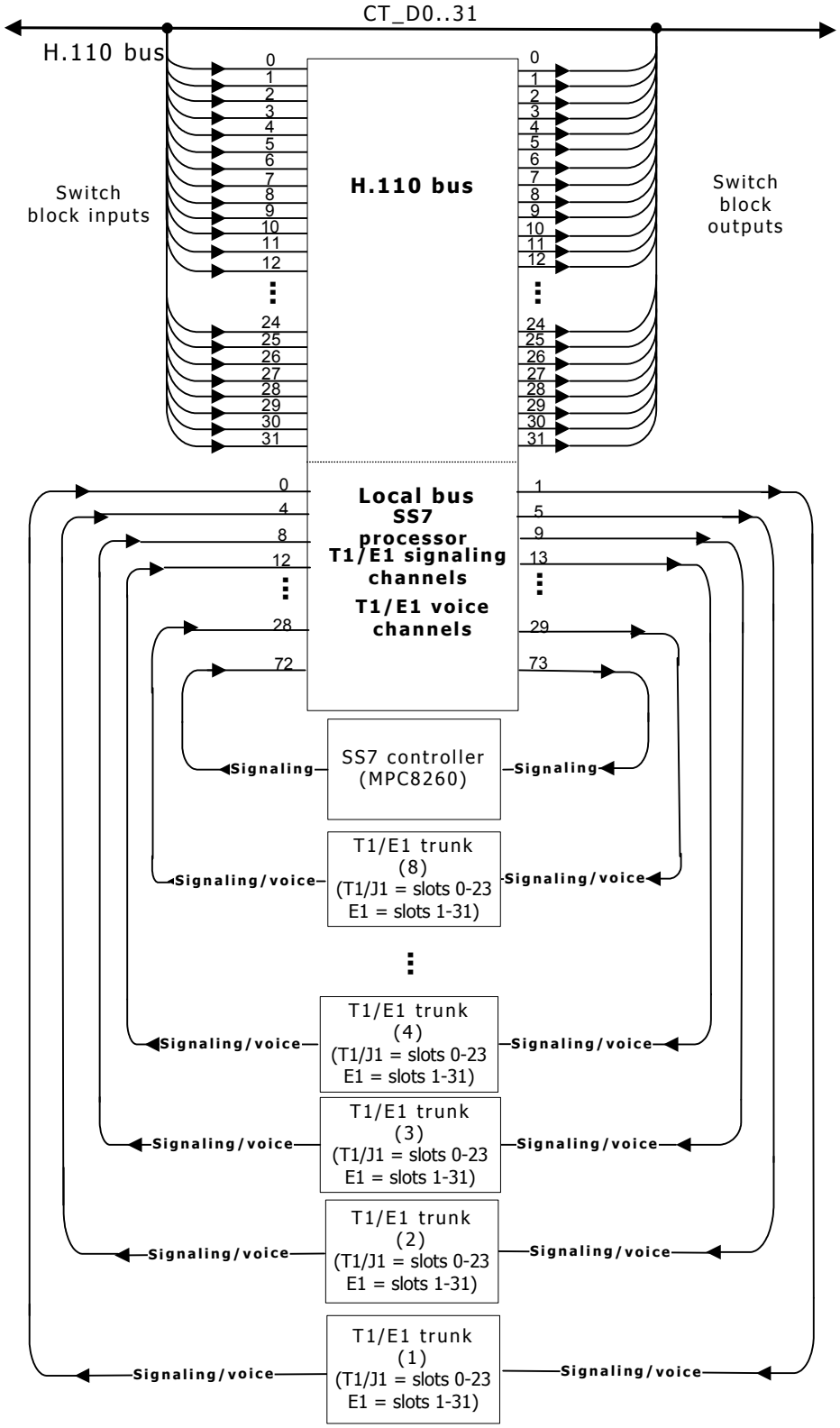
PCI and PCI Express TX switch model

The following illustration shows the switch model for PCI and PCI Express TX boards:



CompactPCI TX switch model

The following illustration shows the switch model for CompactPCI TX boards:



H.100/H.110 and local streams

The following tables list the specific use of each stream in the TX switching model:

H.100/H.110 streams

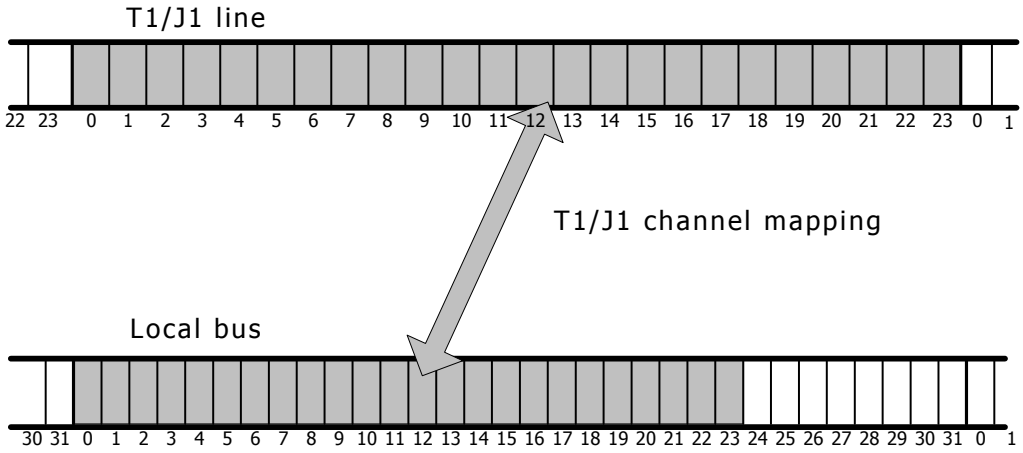
H.100/H.110 bus	Streams 0 through 31, timeslots 0 through 127 (Streams clocked at 8 MHz)

Local streams

Trunk connections (T1 or J1 trunks)	Trunk 1: Streams 0 and 1, Trunk 2: Streams 4 and 5, Trunk 3: Streams 8 and 9, Trunk 4: Streams 12 and 13, Trunk 5: Streams 16 and 17, Trunk 6: Streams 20 and 21, Trunk 7: Streams 24 and 25, Trunk 8: Streams 28 and 29,	timeslots 0 through 23 timeslots 0 through 23 timeslots 0 through 23 timeslots 0 through 23 timeslots 0 through 23 (CompactPCI TX only) timeslots 0 through 23 (CompactPCI TX only) timeslots 0 through 23 (CompactPCI TX only) timeslots 0 through 23 (CompactPCI TX only)
Trunk connections (E1 trunks)	Trunk 1: Streams 0 and 1, Trunk 2: Streams 4 and 5, Trunk 3: Streams 8 and 9, Trunk 4: Streams 12 and 13, Trunk 5: Streams 16 and 17, Trunk 6: Streams 20 and 21, Trunk 7: Streams 24 and 25, Trunk 8: Streams 28 and 29,	timeslots 1 through 31 timeslots 1 through 31 timeslots 1 through 31 timeslots 1 through 31 timeslots 1 through 31 (CompactPCI TX only) timeslots 1 through 31 (CompactPCI TX only) timeslots 1 through 31 (CompactPCI TX only) timeslots 1 through 31 (CompactPCI TX only)
SS7 communication controller	Streams 72 and 73, timeslots 0 through 31	

T1 trunk channel routing

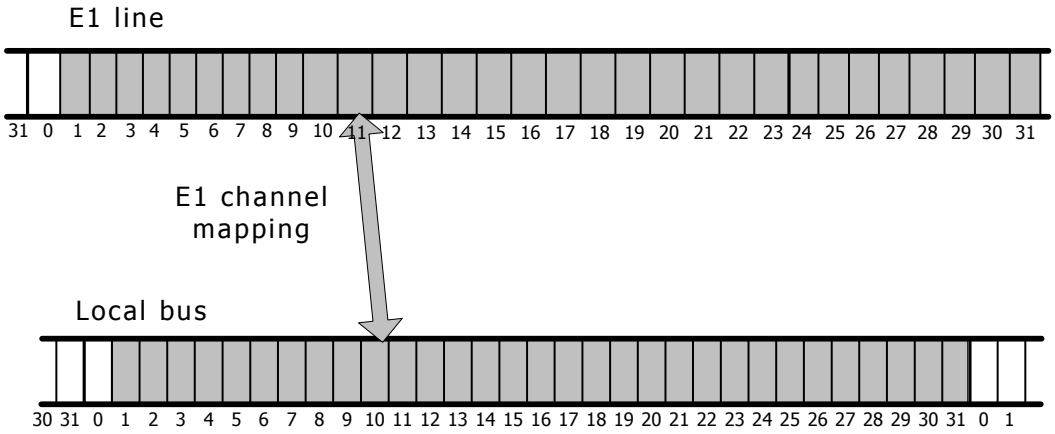
Control access is provided for 24 T1/J1 channels (0 through 23). Each channel on the T1/J1 trunk is placed in a corresponding timeslot on the local bus in the following streams:



Trunk 1: Streams 0, 1
Trunk 2: Streams 4, 5
Trunk 3: Streams 8, 9
Trunk 4: Streams 12, 13
Trunk 5: Streams 16, 17 (CompactPCI TX only)
Trunk 6: Streams 20, 21 (CompactPCI TX only)
Trunk 7: Streams 24, 25 (CompactPCI TX only)
Trunk 8: Streams 28, 29 (CompactPCI TX only)

E1 trunk channel routing

Control access is provided for 31 E1 channels (1 through 31). Timeslot 0 on the E1 line carries framing data and is not switched by the standard TX switching model. Each channel on the E1 trunk is placed in a corresponding timeslot on the local bus in the following streams:



Trunk 1: Streams 0, 1
Trunk 2: Streams 4, 5
Trunk 3: Streams 8, 9
Trunk 4: Streams 12, 13
Trunk 5: Streams 16, 17 (CompactPCI TX only)
Trunk 6: Streams 20, 21 (CompactPCI TX only)
Trunk 7: Streams 24, 25 (CompactPCI TX only)
Trunk 8: Streams 28, 29 (CompactPCI TX only)

Index

U