Dialogic® TX Series SS7 Boards

TX INFO Library Reference Manual

July 2009 64-0456-01

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.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. 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
9000-62754-10	September 2008	SRG, SS7 5.0
64-0456-01	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 TX INFO Library Reference Manual* explains how to use the TX INFO library functions to obtain system-level information. The *txinfo* utility provides a tool that uses the TX INFO library to display current TX board status information. Refer to the *Dialogic® TX Series SS7 Boards TX Utilities Manual* for information about the *txinfo* utility. Source code for the *txinfo* utility is also provided as sample code.

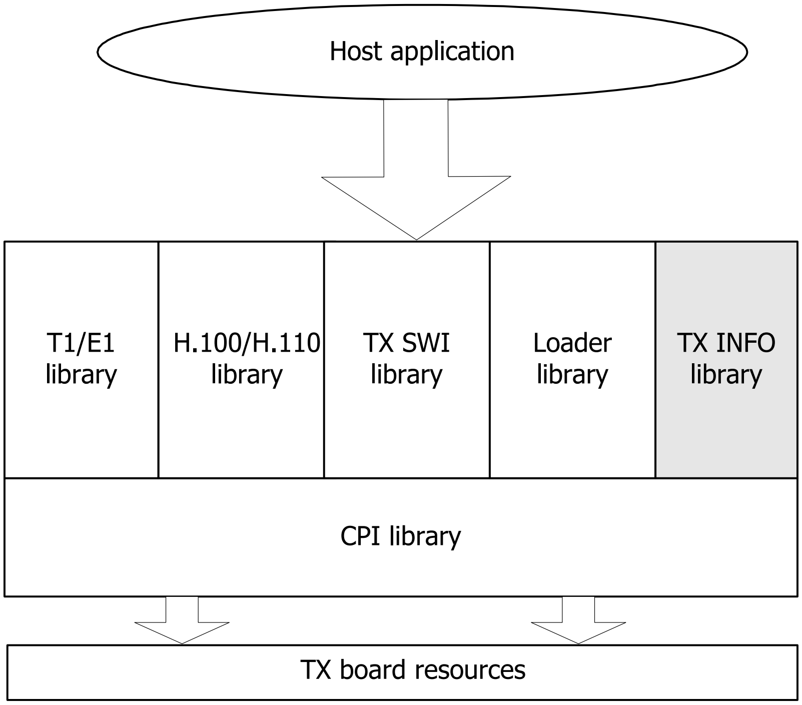
Note: The product(s) to which this document pertains is/are among those sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") in December 2008. Certain terminology relating to the product(s) has been changed, whereas other terminology has been retained for consistency and 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 TX INFO library

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 TX INFO library.



The TX INFO function prototypes are located in the *txinfoapi.h* include file with TX INFO message structures defined in the *txinfomsg.h* include file. The TX INFO functions are provided by the TX INFO library module (*txinfoapi.dll* and *txinfoapi.lib* for Windows systems, *libtxinfoapi.so* for UNIX systems).

Function summary

The TX INFO API functions provide system information such as board serial number, current CPU utilization, and fan state. The following table describes the TX INFO functions:

Function	Description
txinfoAddress	Zero-fills the provided CPI (Communications Processor Interface) packet and then sets addressing information and overall message size in the CPI packet.
txinfoReply	Performs reply message verification and swabbing.
txinfoSystemInfo	Prepares a CPI packet for use as a TX INFO message to retrieve system information.

3 Function reference

Using the function reference

This section provides a reference to the TX INFO library functions. A typical function includes:

Prototype	The prototype is followed by a list of the function's arguments. Dialogic 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.
Return values	The return value for a function is either zero (indicating SUCCESS) or an error code. Since there are no asynchronous functions in the TX INFO library, any return value indicates the final result of the function.

txinfoAddress

Zero-fills the provided CPI (Communications Processor Interface) packet and then sets addressing information and overall message size in the CPI packet.

Prototype

U32 **txinfoAddress** (CPIPKT ***pkt***, U32 ***req_size***, U16 ***board***, U16 ***host_chan***)

Argument	Description
pkt	Pointer to the address of the CPI packet that is to be initialized. The memory area for the packet must be at least ***req_size*** bytes in length.
req_size	Total byte length of the CPI packet (including CPI header size).
board	Destination board number.
host_chan	Source CPI channel number.

Return values

Return value	Description
SUCCESS	
SCCIP_NULL_POINTER	Invalid NULL pointer detected.

Details

txinfoAddress initializes a memory area as a CPI packet that can then be used on a subsequent call to a TX INFO message-specific function (such as **txinfoSystemInfo**). ***req_size*** indicates the total size of the CPI packet memory area. The size of a TX INFO message is provided by calling the appropriate TX INFO message function with a CPI packet address of NULL. The value returned is the total size of the CPI request message.

After calling **txinfoAddress**, the given CPI packet's source indicates the host channel with which the packet is associated. The CPI packet's destination is the TX INFO channel on the target TX board.

See also

txinfoReply

Example

```
req_size = txinfoSystemInfo( NULL, 0 );
if (req_size == 0)
{
    printf( "Error sizing SYSTEM INFO.\n" );
    exit( 1 );
}

/* ----- create the request msg */
pkt = (CPIPKT *)malloc( req_size );
txinfoAddress( pkt, req_size, board, PT_MGR );
status = txinfoSystemInfo( pkt, 0 );
if (status != 0)
{
    printf( "Error [0x%08X] building SYSTEM INFO.\n", status );
    free( pkt );
    exit( 1 );
}

/* ----- send the request msg */
cpistatus = cpia_send( handle, pkt, NULL );
```

txinfoReply

Performs reply message verification and swabbing. **txinfoReply** is called each time a TX INFO message is received. The function verifies the received message and then converts all fields in the message to the native representation of fields that span more than 1 byte (for example, 16-bit or 32-bit values).

Prototype

TXINFO_MSG ***txinfoReply** (CPIPKT ***pkt***, S16 ***len***, U32 ***ccode***)

Argument	Description
pkt	Pointer to CPI packet to be verified and converted to native representation.
len	Byte length of packet to be converted.
ccode	Address where the verification or conversion result code is placed.

Return values

Return value	Description
SUCCESS	
SCCIP_INVALID_TYPE	Invalid TX INFO message type encountered.
SCCIP_NULL_POINTER	Invalid NULL pointer detected.
SCCIP_TOO_SMALL	Provided buffer size is too small.
SCCIP_UNKNOWN_COMMAND	Unknown command (TX INFO operation) encountered.

Details

Call **txinfoReply** for each message that is received in response to any other TX INFO API request (such as a response to a **txinfoSystemInfo** request). Call **txinfoReply** before performing additional processing of the TX INFO response. Once **txinfoReply** has returned successfully, all fields of the received message hold values represented in the host system’s native orientation.

See also

txinfoAddress

Example

```
void CpiaRxNotif( TX_HANDLE handle, void *chkey )
{
    CPIPKT              rcvBuf;
    S16                 len;
    CPI_ERR_TYPE        cpi_error;
    TXINFO_MSG         *msg;
    U32                 ccode;

    /* ----- read the received message ----- */
    len = sizeof( CPIPKT );
    if (cpia_get_data( handle, &rcvBuf, &len ) == CPI_ERROR)
    {
        cpi_error = cpi_get_last_error( );
        fprintf( stderr, "\nERROR: Receive error (%s)\n",
                 cpi_get_error_str( cpi_error ) );
        return;
    }

    /* ----- process (and verify) received system information message ----- */
    msg = txinfoReply( &rcvBuf, len, &ccode );
    if ( (msg == NULL) || (ccode != 0) )
    {   /* invalid message received */
        fprintf( stderr, "\nERROR: Reply message (%p) ccode=0x%08X\n",
                 msg, ccode );
        return;
    }

    /* ----- entire message has been verified ----- */
    sysInfo = &msg->response.info.system;
    temp = sysInfo->status.cpuTemp;
    printf( "CPU Temp: %dC [%dF] degrees\n", temp, (U32)(((temp*9)/5)+32));

    temp = sysInfo->status.boardTemp;
    printf( "Board Temp: %dC [%dF] degrees\n", temp, (U32)(((temp*9)/5)+32));

    temp = sysInfo->status.cpuTemp;
    printf( "Fan State: %u\n", sysInfo->status.fanState );
```

txinfoSystemInfo

Prepares a CPI packet for use as a TX INFO message to retrieve system information.

Prototype

U32 **txinfoSystemInfo** (CPIPKT ***pkt***, U32 ***msgkey***)

Argument	Description
pkt	Pointer to a CPI packet to be filled in as a request message or NULL when using the function to determine the size of a CPI packet needed to perform a system information request.
msgkey	User-controlled key sent in request and provided in response.

Return values

Return value	Description
SUCCESS	
SCCIP_INVALID_SIZE	Invalid buffer size.

Details

Call **txinfoSystemInfo** with ***pkt*** = NULL to determine the required byte length of a CPI packet needed to represent a system information request. Upon successful return from this function, allocate the amount of memory indicated and pass that packet address to **txinfoAddress**. Upon successful return from **txinfoAddress**, call **txinfoSystemInfo** again while providing the address of the memory to be initialized as a system information request message. After the second call to **txinfoSystemInfo**, the packet can be sent to the TX board using **cpi_send** or **cpia_send**. Refer to the *CPI Library Developer's Reference Manual* for information about **cpi_send** or **cpia_send**.

See also

txinfoReply

Example

```
    /* ----- determine the size of the request packet */
    req_size = txinfoSystemInfo( NULL, 0 );
    if (req_size == 0)
    {
        printf( "Error sizing SYSTEM INFO.\n" );

        exit( 1 );
    }

    /* ----- create the request msg */
    pkt = (CPIPKT *)malloc( req_size );
    txinfoAddress( pkt, req_size, board, PT_MGR );
    status = txinfoSystemInfo( pkt, 0 );
    if (status != 0)
    {
        printf( "Error [0x%08X] building SYSTEM INFO.\n", status );
        free( pkt );

        exit( 1 );
    }

    /* ----- send the request msg */
    cpistatus = cpia_send( handle, pkt, NULL );
```