DTI User Manual

Version 2.0

Revision History

	Date	Change
VER 2.0.0	2006-4-28	Created this document
VER 2.0.1	2009-2-9	Amended some errors in the document
VER 2.0.2	2010-3-1	Amended some errors in the document
VER 2.0.3	2010-7-1	Amended some errors in the document

For the latest product information, please visit our website at http://www.ehangcom.com

Announcement:

This product may have design defect or wrong, and there maybe some difference to the parameters and description of this document. We are trying to make our product documents more complete and more accurate by the time it published, but because of the constant changes to actual situation, this document can only provide general information of Ehangcom products.

Ehangcom assumes no liability on the developing and marketing any particular functionally products in this document. Nor any mention or imply a commitment of product applications. This document may update anytime without notice. Therefore, it should not be treated as commitments and assurances.

Ehangcom makes no responsibility of any technical or typesetting errors and any of the resulting in this document.

Ehangcom products are not intended for use in medical, life saving, or life sustaining applications.

Brands and Intellectual Property Rights

Example

Ehangcom, iSX, iSX4000, iSX UAP,Ehcomm are trademarks of Ehangcom Corporation。

Microsoft Windows, Microsoft Windows 98, Microsoft Windows 95, Microsoft NT are registered trademarks of Microsoft Corporation.

SUN Solaris is the trademark of SUN MicroSystem Corporation.

Contents

About This Document

Welcome to this document. It is the ISX4000 DTI user manual. The software-related purpose, intended audience, document description and relevant information are as follows:

Purpose

This manual provides information about the DTI functions in SDK of iSX platform.

Intended Audience

1. Distributors
2. System Integrators
3. Toolkit Developers
4. Independent Software Vendors(ISVs)
5. Value Added Resellers(VARs)
6. Original Equipment Manufactures(OEMs)

How to Use This Manual

This manual is concomitant with the software installation. This document mainly includes the following sections:

1. *Overview:* This section describes the general use of this manual.
2. *Open and Close Functions:* This section describes how to use the open and close DTI device functions.
3. *Switching Functions:* This section describes how to use the switching functions.
4. *ABCD Receiving and Sending Functions*

Relevant Information

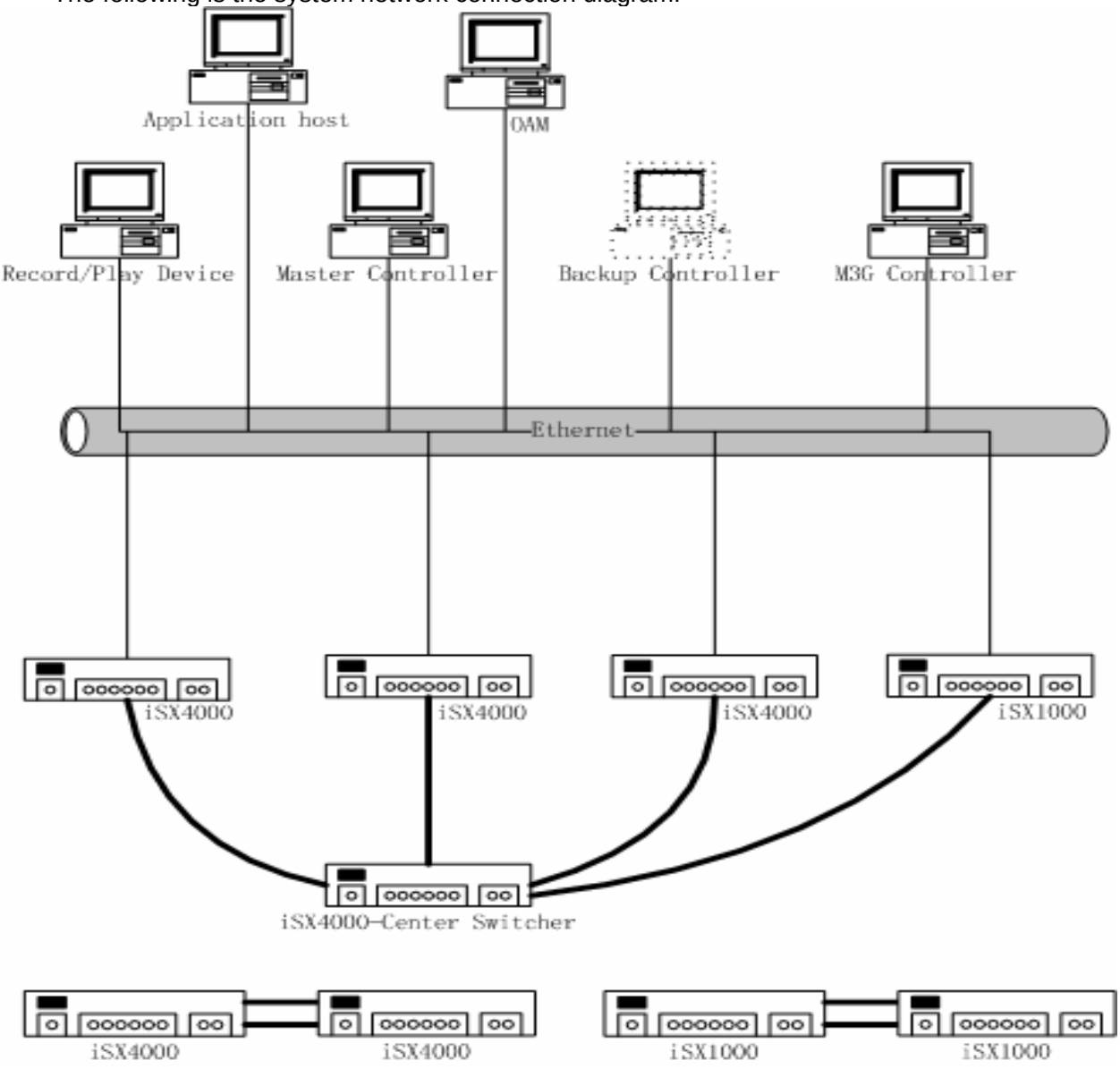
For relevant information of this manual, refer to the following document:

1. *ISX4000 Global Call User Manual*
2. *ISX4000 ISX CALL User Manual*
3. *ISX4000 SRL User Manual*
4. *ISX4000 SDK Programming Guide*
5. *OAM SDK User Manual*
6. *PRD SDK User Manual*
7. *ISX4000 UAP Software Installation Manual*

Chapter 1 Overview

The iSX4000 system is composed of iSX4000/iSX1000 switch nodes, master controller and play/record devices. The maximal system can have 16 iSX4000/iSX1000 switch nodes, 1 center switch node, 1 master controller node and 32 play/record devices. The voice data switching between iSX4000 nodes can be implemented by the ISX4000 center switch through fiber optic couple.

The following is the system network connection diagram:



Note:

The two iSX4000 switch can be coupled by fiber optic, trunk or VoIP.
The two iSX1000 switch can be coupled by trunk or VoIP.
By now, the iSX4000 center switch and backup controller are developing.
The master controller node is a computer that has MC application running.
The Play/Record device nodes are computers that have PRD application running.

An iSX4000 node is composed of 1 motherboard, trunk daughter boards, STM-1 optiic interface daughter board ,DSP daughter boards, XOIP daughter boards, M3G daughter boards and signaling (PRI/SS7/SIP) daughter boards.

The follows are the function description of these boards:

Mother board: mainly responsible for clock management, 16K x 16K switching and daughter board power supply.

Trunk daughter board: provides a trunk interface. The interface can be configured as E1, T1 and J1. A daughter board has 8 trunks. A iSX4000 node maximal has 8 trunk daughter boards and a iSX1000 has only 1 trunk daughter boards.

STM-1 Optic interface board: For simplicity, we mapped all trunks on the optical fiber interface daughter board (STM-1 Optic interface daughter board) as virtual trunk daughter boards and virtual trunks. The virtual trunk board numbered from 8. A virtual trunk daughter board has 8 trunks. These trunks are called optical interface trunk. The usage of the virtual trunk board is as same as the trunk daughter board.

DSP daughter board: It mainly provides functions such as record/play, echo cancellation and voice conference. A daughter board has a maximum of 256 channels

XOIP daughter board: It mainly provides functions such as VOIP and Fax Over IP function.

M3G daughter board: VIDEO-3G-64 daughter board for short. This daughter board mainly provides functions such as audio and video file record and play and 3G-H324M call control. Each daughter board has a maximum of 64 channels.

Signaling daughter boards: include PRI, SS7 and SIP signaling daughter boards. A PRI or SS7 signaling daughter board have capability of up to 100 calls per second. A SIP has capability up to 20 calls per second.

The iSX4000 system is provided with a PRD (play and record device) for convenience of centralized management of voice files. Files are read from PRD during the voice play, and recorded voice is written to PRD during the record.

iSX1000 switch is similar to iSX4000 switch. To get the details of iSX4000/iSX1000 switch, please refer to i*SX4000 System Introduction Manual,iSX4000 Hardware Produce Datasheet* and other documents.

The trunk daughter board operation includes switching operation and signal handling operation. The API described in the document contains just the trunk daughter board switching operation. For the signal handling operation, refer to the *ISX4000 Global Call User Manual.pdf*.

Chapter 2 Interface Function Information

2.1 Device Managements

▪ *ISX_dt_open()*

Name:	int ISX_dt_open(nodeno, brdno, spanno, channel, pusrattr)	
Input parameters:	**char nodeno**	Node number
	char brdno	Broad number
	char spanno	Trunk number
	short channel	Channel number
	void* pusrattr	Pointer to user-defined attributes
Return value:	>0 to return a device handle -1 failure	
Header file:	srllib.h dtilib.h	
Category:	DTI device management	
Mode:	Synchronous	

Description

The function opens a trunk voice channel (trunk time slot) device, and returns a device handle (DTI Handle) if successful. The handle returned can be used in a switching function, but can not be used in a signaling handling function (**ISX_gc_*xxxx*()**). The trunk voice channel device can only be opened by one process using **ISX_dt_open()**. That is, if process A has opened one trunk voice channel device successfully using **ISX_dt_open()** and process B tries to open the same device using this function**,** failure is returned, except that process A has closed the device using **ISX_dt_close().** For the same trunk voice channel device, if it has been opened by **ISX_dt_open()**, it can also be opened by **ISX_gc_OpenEx()**. This is because their purposes are different, i.e. the purposes of using the first and second functions to open a channel are switching and signaling handling respectively.

The device handle returned by the function is defined by Ehangcom itself, and it's not a standard operating system file handle. Therefore, using operating system commands such as **read()**, **write()** or **ioctl()** to operate the handle may cause unpredictable results.

Parameter	Description
nodeno	Node number: specifies a switch node number.
brdno	Trunk subbroad number: >8 means a optical fiber interface trunk (8 trunks each broad。For example optical fiber interface has 127 E1 ,broad 0~7 are circuit interface trunks, broad 8~15 are optical fiber interface trunks).
spanno	Trunk number
channel	Channel number
pusrattr	Points to buffer where user-defined attributes are stored

Cautions

- Do not use the **open( )** function of an operating system to open a device, otherwise unpredictable results will occur.
- When applications generate a child process from a parent process, the child process cannot inherit the device handle of the parent process.
- For channel E1, there are two modes. One mode is a mapping (a time slot number is mapped into a channel number according to a certain rule), and the other mode is non-mapping (a channel number is a time slot number). The default mode is mapping. You can call the **ISX_sr_TrunkChMapping()** function to change it to non-mapping. If the trunk type is E1, the valid channel numbers are 1-30 and 1-31 in the mapping and no-mapping modes respectively. If the trunk type is T1/J1, only the non-mapping mode is available and the valid channel numbers are 1-24;
  In the mapping mode, an E1 time slot number is mapped into a channel number according to the following rule:
  Time slot number:0 1 2 … 15 16 17 … 30 31
  Channel number : x 1 2 … 15 x 16 … 29 30
- For the ISX1000 switch, only one trunk board is provided and **brdno** is fixed as **0**.

Errors

If the function returns -1, it indicates that the function calling fails. To obtain the failure reason, please use the **ISX_sr_getlasterr()** function.

Example

```
#include <srllib.h>
#include <dtilib.h>
main()
{
    int dtidevh;                    /* dti channel device handle */
     /*   Open dti channel 1 device   */
    if ( ( dtidevh = ISX_dt_open( 0, 0, 0, 1 ) ) == -1 ) {
        printf( "Cannot open dti channel. errno = %d", ISX_sr_getlasterr() );
        exit( 1 );
    }
    ...
}
```

Relevant information

- **ISX_dt_close()**
- **ISX_sr_getlasterr()**

▪ *ISX_dt_close()*

Name:	int ISX_dt_close(dev, oflags)	
Input parameters:	**int dev**	Valid DTI channel device handle
	int ofalgs	Reserved for future use
Return value:	0 success -1 failure	
Header file:	srllib.h dtilib.h	
Category:	DTI device management	

Mode:	Synchronous

Description

The function closes a DTI channel device handle (that is returned after the device is opened by **ISX_dt_open()**). The function does not change any state of the device (such as switch state). It only releases the device handle and breaks the link between the calling process and the device.
Note: After **ISX_dt_close()** runs successfully, all events related to the device are prohibited.

Parameter	Description
dev	Specifies a valid device handle returned by **ISX_dt_open()**.
oflags	This parameter is reserved for future use.

Cautions

- Once a device is closed, the process that opens the device cannot use this handle to operate the device any more.
- Do not use the **close( )** function of an operating system to close a device handle, otherwise unpredictable results will occur.
- **ISX_dt_close( )** clears the event buffer of a closed device.

Errors

If the function returns -1, it indicates that the function calling fails. To obtain the failure reason, please use the **ISX_sr_getlasterr()** function.It returns -1 to indicate the failure, To obtain the failure reason please use **ISX_sr_getlasterr()** function.

Example

```
#include <srllib.h>
#include <dtilib.h>
main()
{
    int dtidevh;            /* dti channel device handle */
    /*
     * Open dti channel 1 device
     */
    if ( ( dtidevh = ISX_dt_open( 0, 0, 0, 1 ) ) == -1 ) {
        printf( "Cannot open dti channel. errno = %d", ISX_sr_getlasterr() );
        exit( 1 );
    }
    /*   Continue processing
     */
    . . .
    /*
     * Done processing - close device.
     */

    if ( ISX_dt_close( dtidevh ) == -1 ) {
        printf( "Cannot close dti channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

Relevant information

- **ISX_dt_open()**
- **ISX_sr_getlasterr()**

2.2 Switching Functions

The switching-related function mentioned in this section is designed only for purpose of compatibility with Dialogic. If you are not transplanting Dialogic program, we strongly recommend you to use **ISX_nr_scroute** and **ISX_nr_scunroute** functions, not the switching functions mentioned in this section, to establish and interrupt switching. For details, refer to the *ISX4000 VOICE User Manual.pdf*

▪ *ISX_dt_listen()*

Name:	int ISX_dt_listen(chdev, sc_tsinfop)	
Input parameters:	**int chdev**	DTI channel device handle
	SC_TSINFO* sc_tsinfop	Pointer to switch matrix time slot information structure
Return value:	0 success -1 failure	
Header file:	srllib.h dtilib.h	
Category:	Time slot switching	
Mode:	Synchronous	

Description

The function establishes switching between the DTI channel device receiving terminal (receiving time slot) and a time slot (transmitting terminal of another device) so that this DTI channel device can receive the data from this time slot and this function sets up a half-duplex based on DTI channel. Both switching devices must be located in the same node. Cross-node device switching must use the Optical DTI channel. For details, refer to relevant documents.

Parameter	Description
chdev	DTI device channel handle returned by **ISX_dt_open( )**
sc_tsinfop	Specifies a pointer to the **SC_TSINFO** structure

Multiple DTI device channel receiving terminals may be simultaneously connected to the same time slot, but one DTI device channel can be connected to only one time slot.

Cautions

- This function will return failure when an invalid device handle or time slot is specified.
- You are strongly recommended to use **ISX_nr_scroute()** instead of **ISX_dt_listen** because **ISX_nr_scroute()** acts more efficiently.

Errors

If the function calling fails, -1 is returned. Use the SRL standard attribute function **ISX_ATDV_LASTERR()** to obtain the error code or use **ISX_ATDV_ERRMSGP()** to obtain a descriptive error message. The following error codes may be returned:

EDX_BADPARM: Parameter error
EDX_TIMEOUT: Function running timeout
EDX_BUSY: Device is busy
EDX_OPERTIMEOUT: Operation times out and no device response is received within the specified time
EDX_SYSTEM: System error (e.g. the network connection to MasterController is

interrupted)

Example

```
#include <srllib.h>
#include <dtilib.h>
main( )
{
    int voxh;                       /* Voice channel device handle */
    int dtih;                       /* Digital channel (time slot) device handle */
    SC_TSINFO   sc_tsinfo;     /* Time slot information structure */
    long scts;                      /* Switch Matrix time slot */
    /* Open voice channel device */
    if ((voxh = ISX_dx_open(DT_DSP_CH, 0, 0, 0)) == -1) {
        printf("Cannot open voice channel. errno = %s", ISX_sr_getlasterr() );
        exit(1);
    }
    /* Fill in the Switch Matrix time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get Switch Matrix time slot connected to transmit of voice channel */
    if (ISX_dx_getxmitslot(voxh, &sc_tsinfo) == -1) {
        printf("Error message = %s", ISX_ATDV_ERRMSGP(voxh));
        exit(1);
    }
    /*Open node0 board0 span0 time slot 1 on Digital network interface device*/
    if ((dtih = ISX_dt_open(0, 0, 0, 1)) == -1) {
        printf("Cannot open time slot 1.    errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /* Connect the receive of digital channel (time slot) 1 on board 0 to
      Switch Matrix transmit time slot of voice channel*/
    if (ISX_dt_listen(dtih, &sc_tsinfo) == -1) {
        printf("Error message = %s", ISX_ATDV_ERRMSGP(dtih));
        exit(1);
    }
}
```

Relevant information

- **ISX_dt_unlisten()**
- **ISX_dt_getxmitslot()**
- **ISX_dx_getxmitslot( )**

▪ *ISX_dt_unlisten()*

Name:	int ISX_dt_unlisten(chdev)	
Input parameters:	**int chdev**	DTI channel device handle
Return value:	0 success -1 failure	
Header file:	srllib.h dtilib.h	
Category:	Time slot switching	
Mode:	Synchronous	

Description

The function interrupts the switching between the DTI channel device receiving terminal and the connected transmitting time slot**.** If **ISX_dt_listen()** is successfully called, the switching between the DTI channel device receiving terminal and the connected transmitting time slot is automatically interrupted. Therefore, when changing connections, you need not call the **ISX_dt_unlisten( )** function.

Parameter	Description
chdev	DTI channel device handle returned by **ISX_dt_open( )**

Cautions

- This function will return failure when an invalid device handle or time slot is specified.
- You are strongly recommended to use **ISX_nr_scroute()** instead of **ISX_dt_listen**, because **ISX_nr_scunroute()** acts more efficiently.

Errors

If the function calling fails, -1 is returned. Use the SRL standard attribute function **ISX_ATDV_LASTERR( )** to obtain the error code or use **ISX_ATDV_ERRMSGP( )** to obtain a descriptive error message. The following error codes may be returned:
EDX_BADPARM: Parameter error
EDX_TIMEOUT: Function running timeout
EDX_BUSY: Device is busy
EDX_OPERTIMEOUT: Operation times out and no device response is received within the specified time
EDX_SYSTEM: System error (e.g. the network connection to MasterController is interrupted)

Example

```
#include <srllib.h>
#include <dtilib.h>
main( )
{
    int devh;                   /* Digital channel (time slot) device handle */
    /* Open node 0 board 0 time slot 1 device */
    if ((devh = ISX_dt_open(0, 0, 0, 1)) == -1) {
        printf("Cannot open time slot. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /* Disconnect receive of time slot 1 from all Switch Matrix time slots */
    if (ISX_dt_unlisten(devh) == -1) {
        printf("Error message = %s", ISX_ATDV_ERRMSGP(devh));
        exit(1);
    }
}
```

Relevant information

- **ISX_dt_listen()**

▪ *ISX_dt_getxmitslot()*

Name:	int ISX_dt_getxmitslot(chdev, sc_tsinfop)	
Input parameters:	**int chdev**	Valid DTI channel device handle
	SC_TSINFO *sc_tsinfop	Pointer to a time slot information structure

Return value:	0 success -1 failure
Header file:	srllib.h dtilib.h
Category:	Time slot switching
Mode:	Synchronous

Description

The **ISX_dt_getxmitslot( )** function returns the time slot of a specified DTI channel device transmitting terminal. Time slot information is stored in the **SC_TSINFO** structure.

Parameter	Description
chdev	Valid DTI channel device handle
sc_tsinfop	Pointer to the **SC_TSINFO** structure to obtain time slot information

Cautions

- The function calling will fail when an invalid channel device handle is specified.

Errors

If the function calling fails, -1 is returned. Use the SRL standard attribute function **ISX_ATDV_LASTERR( )** to obtain the error code or use **ISX_ATDV_ERRMSGP( )** to obtain a descriptive error message. The following error codes may be returned:
EDX_BADPARM: Parameter error
EDX_BUSY: Device is busy

Example

```
#include <srllib.h>
#include <dtilib.h>
main( )
{
    int devh;                    /* Time slot device handle */
    SC_TSINFO sc_tsinfo;         /* Time slot information structure */
    long scts;                   /* Switch Matrix time slot */
    /* Open node 0 board 0 time slot 1 for Digital network interface device */
    if ((devh = ISX_dt_open(0, 0, 0, 1)) == -1) {
        printf("Cannot open time slot. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /* Fill in the Switch Matrix time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get Switch Matrix time slot connected to transmit of time slot (digital
    channel) 1 on node 0 board 0    */
    if (ISX_dt_getxmitslot(devh, &sc_tsinfo) == -1) {
        printf("Error message = %s", ISX_ATDV_ERRMSGP(devh));
        exit(1);
    }
    printf("%d is transmitting on Switch Matrix time slot %d", devh, scts);
}
```

Relevant information

- **ISX_dt_listen()**
- **ISX_dx_listen( )**

3. ABCD Receiving and Sending Functions

▪ *ISXE_dt_settssig()*

Name:	int ISXE_dt_settssig(dev, ucABCD, mode, ulOperIndex)	
Input parameters:	**int chdev**	Valid DTI channel device handle
	unsigned char ucABCD	ABCD code to be sent
	unsigned short mode	Synchronous or asynchronous mode
	unsigned long ulOperIndex	Operation number
Return value:	0 success -1 failure	
Header file:	srllib.h dtilib.h	
Category:	ABCD receiving and sending	
Mode:	Synchronous or asynchronous	

Description

The **ISXE_dt_settssig( )** function sends ABCD bits.

Parameter	Description
dev	Valid DTI channel device handle
ucABCD	ABCD bits to be sent. ABCD bits are only 4BIT long. Therefore, ucABCD that is less than 4BIT is valid, and ucABCD that is more than -4BIT makes no sense.
mode	Synchronous or asynchronous: EV_ASYNC: Asynchronous mode EV_SYNC: Synchronous mode(default)
ulOperIndex	Operation number. When the operation is completed and a **DTEV_SETTSSIG** event is generated in the asynchronous mode, the number can be obtained through the **ISX_sr_getevtoperindex()** function.

Cautions

- The function calling will fail when an invalid channel device handle is specified.
- It can receive and send ABCD bits only when the trunk signaling mode is set to **CAS**. For signaling mode setting, refer to the *ISX4000 OAM User Manual*.

Errors

If the function calling fails, -1 is returned. Use the SRL standard attribute function **ISX_ATDV_LASTERR()** to obtain the error code or use **ISX_ATDV_ERRMSGP()** to obtain a descriptive error message. The following error codes may be returned:
EDX_BADPARM: Parameter error
EDX_BUSY: Device is busy

Example

```
#include <srllib.h>
#include <dtilib.h>
main( )
```

```
{
    int devh;                       /* Time slot device handle */

    /* Open node 0 board 0 time slot 1 for Digital network interface device */
    if ((devh = ISX_dt_open(0, 0, 0, 1)) == -1) {
        printf("Cannot open time slot. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }

    unsigned char ucABCD = 0x0a;
    if (ISXE_dt_settssig(devh, ucABCD) == -1) {
        printf("settssig fail.\n");
        exit(1);
    }

}
```

Relevant information

- **ISX_dt_open()**
- **ISXE_dt_gettssig()**

ISXE_dt_gettssig()

Name:	int ISXE_dt_gettssig(dev, pucABCD, mode, ulOperIndex)	
Input parameters:	**int chdev**	Valid DTI channel device handle
	unsigned char* pucABCD	Pointer to the address where an ABCD code is stored
	unsigned short mode	Synchronous or asynchronous mode
	unsigned long ulOperIndex	Operation number
Return value:	0 success -1 failure	
Header file:	srllib.h dtilib.h	
Category	ABCD receiving and sending	
Mode:	Synchronous or asynchronous	

Description

The **ISXE_dt_gettssig( )** function inquires about ABCD bits.

Parameter	Description
dev	Valid DTI channel device handle
pucABCD	Pointer to the address where an ABCD code is stored
mode	Synchronous or Asynchronous: EV_ASYNC: Asynchronous mode EV_SYNC: Synchronous mode(default)

ulOperIndex	Operation number. When the operation is completed and a **DTEV_SETTSSIG** event is generated in the asynchronous mode, the number can be obtained through the **ISX_sr_getevtoperindex()** function.

Cautions

- The function calling will fail when an invalid channel device handle is specified.
- It can receive and send ABCD bits only when the trunk signaling mode is set to **CAS**. For signaling mode setting, refer to the *ISX4000 OAM User Manual*.

Errors

If the function calling fails, -1 is returned. Use the SRL standard attribute function **ISX_ATDV_LASTERR( )** to obtain the error code or use **ISX_ATDV_ERRMSGP( )** to obtain a descriptive error message. The following error codes may be returned:
EDX_BADPARM: Parameter error
EDX_BUSY: Device is busy

Example

```
#include <srllib.h>
#include <dtilib.h>
main( )
{
    int devh;                    /* Time slot device handle */

    /* Open node 0 board 0 time slot 1 for Digital network interface device */
    if ((devh = ISX_dt_open(0, 0, 0, 1)) == -1) {
        printf("Cannot open time slot. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }

    unsigned char* ucABCD;
    if (ISXE_dt_gettssig(devh, &ucABCD) == -1) {
        printf("settssig fail.\n");
        exit(1);
    }
    else{
        printf("The ABCD status of the dev(%d) is: %x\n", ucABCD);
    }

}
```

Relevant information

- **ISX_dt_open()**
- **ISXE_dt_settssig()**

4. Extended Attribute Functions

▪ *ISX_dt_SetUsrAttr()*

Name:	int ISX_dt_SetUsrAttr(nDeviceHandle, usrattr)	
Return value:	**int nDeviceHandle**	DTI device handle
	void* usrattr	User attribute
Return value:	0 success <0 failure	
Header file	srllib.h dtilib.h	
Category:	System control	
Mode:	Synchronous	

Description

The **ISX_dt_SetUsrAttr( )** function sets channel attributes (user-defined). User attributes that are set by the user can be obtained through the **ISX_dt_GetUsrAttr()** function.

Parameter	Description
nDeviceHandle	DTI device handle
usrattr	User defined attributes that the user can obtain through the **ISX_dt_GetUsrAttr()** function

Suspension event

None

Cautions

None

Errors

If the function calling fails, the return value is less than 0. Fault causes can be obtained by calling the **ISX_ATDV_LASTERR( )** function.

Example

```
#include <srllib.h>
#include <dtilib.h>
#define MAXCH     30                    /* max. number of channels in system */

/*
 * Data structure which stores all information for dti channel
 */
struct chbag {
    int         dev;                    /* dti channel handle */
    UCHAR       ucIsxNo;                /* ISX node no. */
    UCHAR       ucBrdNo;                /* DTI Board no. */
    UCHAR       ucSpanNo;               /* DTI span no. */
    UCHAR       ucChannel;              /* DTI channel no. */
} dtiport[MAXCH+1];

int OpenDtiCh()
{
    for(int j=1; j<MAXCH; j++){
       dtiport[j].ucIsxNo = 0;
       dtiport[j].ucBrdNo = 0;
```

```
            dtiport[j].ucSpanNo = 0;

            dtiport[j].ucChannel = j;
            dtiport[j].dev = ISX_dt_Open(0, 0, 0, j, NULL);
            if(dtiport[j].dev == -1){
               printf("open dti channel fail.reason:%x\n", ISX_sr_getlasterr());
               return(-1);
            }
            ISX_dt_SetUsrAttr(dtiport[j].dev, (void*)&dtiport[j]);
         }
      }
```

Relevant information

- **ISX_dt_GetUsrAttr()**
- **ISX_dt_open()**

▪ *ISX_dt_GetUsrAttr()*

Name:	int ISX_dt_GetUsrAttr(nDeviceHandle, usr_attrp)	
Input parameters	**int nDeviceHandle**	Valid DTI device handle
	void usr_attrp**	Pointer to the address where the user attributes are stored
Return value	0 success <0 failure	
Header file	srllib.h dtilib.h	
Category:	System control	
Mode:	Synchronous	

Description

The **ISX_dt_GetUsrAttr( )** function obtains the user-defined attributes (that are set through **ISX_dt_open()** or **ISX_dt_SetUsrAttr()).**

Parameter	Description
nDeviceHandle	DTI device handle
usr_attrp	Pointer to the address where the user attributes are obtained

Cautions

None

Errors

If the function calling fails, the return value is less than 0. Fault causes can be obtained by calling the **ISX_ATDV_LASTERR( )** function.

Example

```
#include <srllib.h>
#include <dtilib.h>
#define MAXCH     30                    /* max. number of channels in system */
/*
 * Data structure which stores all information for each dti channel
 */
struct chbag {
    int         dev;                    /* dti channel handle */
    UCHAR       ucIsxNo;                /* ISX node no. */
```

```
    UCHAR       ucBrdNo;                    /* DTI Board no. */
    UCHAR       ucSpanNo;                    /* DTI span no. */
    UCHAR       ucChannel;                  /* DTI channel no. */
} dtiport[MAXCH+1];

int OpenDtiCh()
{
    for(int j=1; j<MAXCH; j++){
        dtiport[j].ucIsxNo = 0;
        dtiport[j].ucBrdNo = 0;
        dtiport[j].ucSpanNo = 0;
        dtiport[j].ucChannel = j;
        dtiport[j].dev = ISX_dt_open(0, 0, 0, j, (void*)&dtiport[j]);
        if(dtiport[j].dev == -1){
            printf("open dt channel fail.reason:%x\n", ISX_sr_getlasterr());
            return(-1);
        }
    }
}

int get_usrattr(int dev)
{
     chbag *vattrp;              /* to retrieve the attribute */

     if ( ISX_dt_GetUsrAttr( dev, (void**)&vattrp) != 0) {
         printf ("ISX_dt_GetUsrAttr() fail.reason:%d\n", ISX_ATDV_LASTERR());
         return (-1);
     }
     printf("dev(%d)'s attr is %d:%d:%d:%d\n",
              vattrp->ucIsxNo, vattrp->ucBrdNo,
               vattrp->ucSpanNo, vattrp->ucChannel);
     return (0);
}
```

Relevant information

- **ISX_dt_SetUsrAttr()**
- **ISX_dt_Open()**

5. Events

▪ *NR_SCROUTE*

Switching end event. Switching operation: **ISX_nr_scroute()**. For details, refer to the description of the ***ISX4000 Voice User Manual.pdf***.

▪ *DTEV_SETTSSIG*

Sets an ABCD function end event. ABCD setting operation: **ISXE_dt_settssig()**. The event can be received only in the asynchronous operation mode. The event contains data of SETTSSIG_EVT_DATA type to indicate whether the setting operation is successful. For example:

```
case DTEV_SETTSSIG:
{
    SETTSSIG_EVT_DATA* psed = (SETTSSIG_EVT_DATA*)ISX_sr_getevtdatap();
    int iDev = ISX_sr_getevtdev();
    if(psed->sRetVal == 0)
        printf("dev:%d, ISXE_dt_settssig ok\n", iDev);
    else
        printf("dev:%d, ISXE_dt_settssig fail. ret=0x%x\n", iDev, (USHORT) psed->sRetVal);
    break;
}
```

▪ *DTEV_GETTSSIG*

Query about an ABCD function end event. ABCD query operation: **ISXE_dt_gettssig()**. The event can be received only in the asynchronous operation mode. The event contains data of SETTSSIG_EVT_DATA type to indicate the following: whether the query operation is successful; queried ABCD bits. For example:

```
case DTEV_GETTSSIG:
{
    GETTSSIG_EVT_DATA* pged = (GETTSSIG_EVT_DATA*)ISX_sr_getevtdatap();
    int iDev = ISX_sr_getevtdev();
    if(pged->sRetVal == 0)
        printf("dev:%d, ISXE_dt_gettssig ok, ABCD=0x%02x\n", iDev, pged->ucABCD);
    else
        printf("dev:%d, ISXE_dt_gettssig fail. ret=0x%x\n", iDev, (USHORT) pged->sRetVal);
    break;
}
```

▪ *DTEV_ABCD*

The event is a system-active event. When the embedded system on the motherboard detects that the ABCD bits of a channel are changed , the system will promptly notify the SDK that will then notify the application in the form of **DTEV_ABCD** event. This event can not be shielded and contains data of ABCD_EVT_DATA type to save changed ABCD bits. For example:

```
case DTEV_ABCD:
{
    ABCD_EVT_DATA* paed = (ABCD_EVT_DATA*)ISX_sr_getevtdatap();
    int iDev = ISX_sr_getevtdev();
    printf("dev:%d, ABCD changed, ABCD=0x%02x\n", iDev, paed->ucABCD);
    break;
}
```