M3G User Manual

Version 2.0

Revision history

	Date	Change
VER 2.0.0	July 13, 2009	Created
VER 2.0.1	October 21, 2009	Corrected the description of the functions record and play
VER 2.0.2	March 9, 2010	Added the eM3G_DATA_DIR of the functions record and play
VER 2.0.3	July 1, 2010	Added the section 2.6 for Video Editing Functions

For the latest products and their related information, please visit our website: http://www.ehangcom.com

Announcement:

This document is protected by intellectual property laws of China and other countries. The activities of reproduction, spreading, or modifying behaviors(whether in the Company's internal and external) are illegal ,unless it gets written agreement ,contract or authorize of Ehangcom Technology.

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

All the names of the products and services in this document are specific vendors trademarks or registered trademarks. The intellectual property rights of those specific equipment and software referred in this document are protected by the relative laws of china and other countries.

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 The Document

Welcome to read the M3G user manual of the iSX univsal application platform. The follows are the purpose of the software, intended audience, document description and other information.

Purpose

This manual is about using the M3G functions include in the SDK of iSX univsal application platform.

Intended audience

1. Distributors
2. System Integrators
3. Toolkit Developers
4. Independent Software Vendor
5. Reseller
6. Original Equipment Manufacturers (OEMs)

How to use this manual

This manual is produced with installing the iSX4000 UAP software, and this document mainly contains the following sections:

1. Overview: Purpose of this manual.
2. Interface Functions Description: The instructions of using the M3G functions.

Glossary

M3G: multimedia, multi-functional 3G services
M3GC: M3G Controller Application
M3G daughter board: abbreviation of VIDEO-3G-64 daughter board
LC: logical channel of the H324M type channel, please refer to **eM3G_CHAN_TYPE**
TCS: terminal capability set
MSD: master-slave determination

References

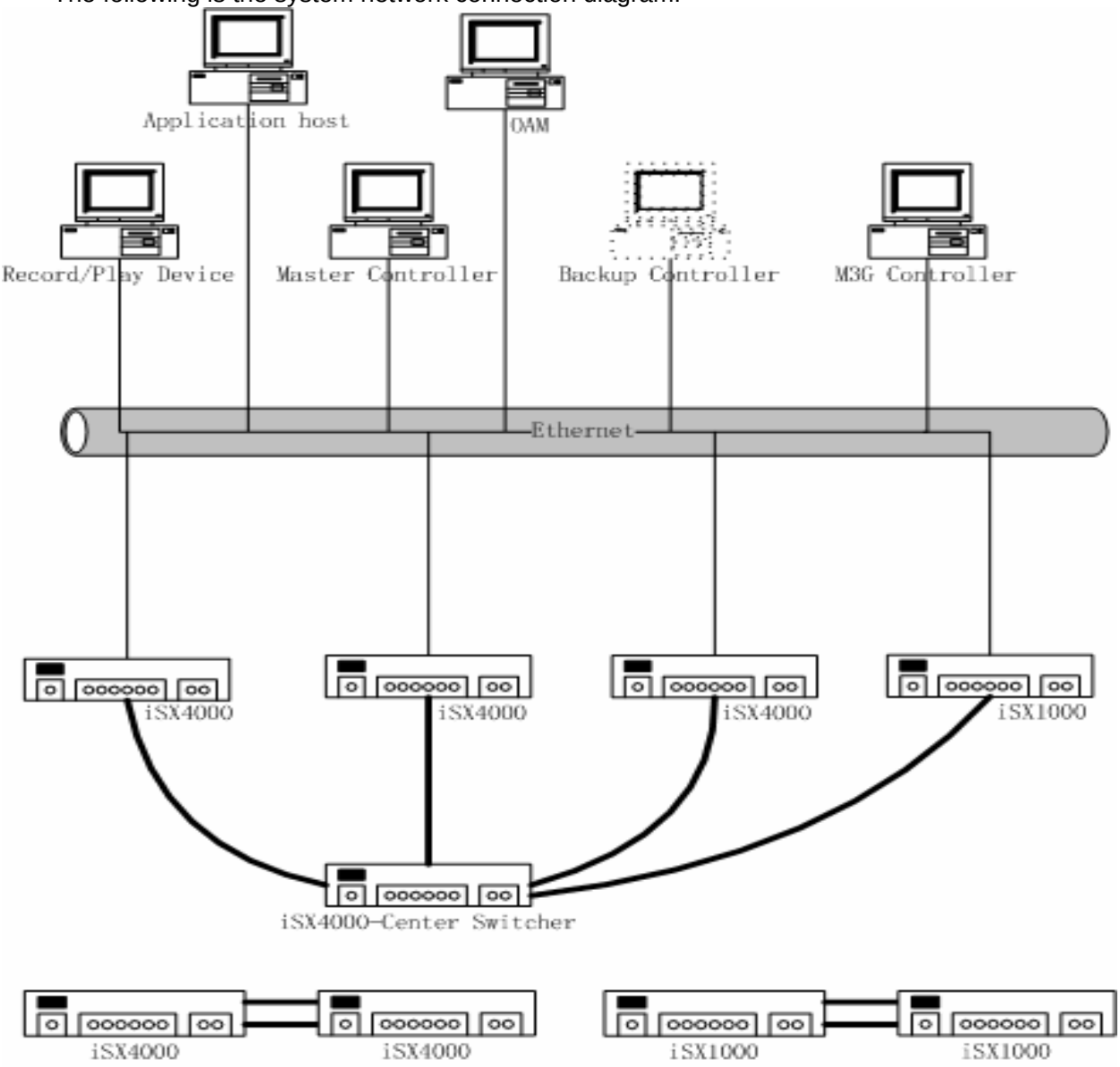
For the relevant information in this manual, please refer to the following documents:

1. ISX4000 DTI User Manual
2. ISX4000 Global Call User Manual
3. ISX4000 ISX CALL User Manual
4. ISX4000 SRL User Manual
5. ISX4000 SDK Programming Guide
6. OAM SDK User Manual
7. PRD SDK User Manual
8. ISX4000 Software Installing 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.

This document describes the API of the M3G functions.

Chapter 2 API Functions Description

A M3G channel is a universal type and can be configure to the constant type such as Voice, Video, H324M, Voice conference, Video mixer and so much. (please refer to **eM3G_CHAN_TYPE**). At present, the types such as Voice, Video, Voice conference, Video mixer and H324M were supported. The channel type can be changed using the ***configuration function***. Normally, the ***media functions*** and ***I/O functions*** can be applicable to all the types if these functions have no special limit, and the ***H324M call functions*** is only applicable to the H324M type.

2.1. Open/Close Functions

- ***ISX_m3g_Open()***

Function name:	INT ISX_m3g_Open (nodenum, brdnum , rfu, pusrattr)	
Parameters:	**CHAR nodenum**	The valid iSX node number that the M3G board installed
	CHAR brdnum	M3G board number
	INT rfu	Reserved
	VOID* pusrattr	The pointer of user attribute
Return value:	>0 board device handle -1 failure	
Header file:	srllib.h m3glib.h	
Category:	Extended functions	
Mode:	Synchronous	

Description

The application use ISX_m3g_Open() function to open the M3G board and get the device handle on successful. One board can only be opened by a application, so if A application has successfully opened the board, then the B application failed when opening the same board. When the A application called ISX_m3g_close() function to close the M3G board, the B application can open the same board now. The device handle returned by the ISX_m3g_Open() function is defined by Ehangcom, and it is not a file handle of the operating system. The application can't use operating system functions such as read(), write() or ioctl() to operate the handle, Unexpected result will occur on doing so.

Parameters	Description
nodenum	The valid iSX node number that the M3G board installed
brdnum	Board number; M3G daughter board number
rfu	Reserved
pusrattr	The pointer of user-defined attributes

Warning

- Please do not use the operating system open() function to open the board, otherwise unexpected results will occur.
- When the parent process of the application generated the child process, the child process can not inherit the device handle of the parent process.
- For normal boards, the valid channel number is 0 to 63.
- The device handle got from the function is used by function **ISX_m3g_DownloadFont()** for downloading the TTF font

Error

The result code -1 for the function call fails. Please use the ISX_sr_getlasterr() function to retrive the failure reasons.

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g board device
    */
    if ( (devh=ISX_m3g_Open(0, 0))== -1 ) {
       printf( "Cannot open m3g board errno = %d", ISX_sr_getlasterr() );
       exit(1);
    }
    /*
    * Continue processing
    */
    //...
    if (ISX_m3g_Close( devh ) == -1 ) {
        printf( "Cannot close M3G board. errno = %d", ISX_sr_getlasterr() );
    }
}
```

References

- **ISX_m3g_Close()**
- **ISX_sr_getlasterr()**

ISX_m3g_OpenEx()

Function name:	int ISX_m3g_OpenEx(devp, nodenum, brdnum, channel, mode, pusrattr)	
Parameters:	**int *devp**	The pointer for retriving the M3G channel device handle
	char nodenum	iSX node number
	char brdnum	Board number
	short channel	Channel number

	unsigned short mode	Mode
	void* pusrattr	The pointer of user attribute
Return value:	0 success -1 failure	
Header file:	srllib.h m3glib.h	
Category:	Open/close	
Mode:	Synchronous and asynchronous	

Description

The application use ISX_m3g_OpenEx() function to open the M3G channel device of the M3G daughter and got the channel device handle on successful. The channel device handle put in the memory specified by the parameter devp when the function completed. In the asynchronous mode, the event M3GEV_OPEN_CMPLT is for successful and the event M3GEV_OPEN_FAIL for unsuccessful to open the M3G channel device. The channel can only be opened by only one application, so, if A applicaiton has successfully opened a channel, then the B application will fail to open the same channel. Only when A application called ISX_m3g_close() function to close the M3G channel device handle, the B application can open the same channel now. The channel device handle retrived by this function is defined by Ehangcom, it is not a standard file handle of the operating system, so the application can't use operating system functions such as read(), write() or ioctl() to operate the handle. Unexpected result will occur on doing so.

Parameters	Description
devp	The pointer for retriving the M3G channel device handle and can not be empty.
nodenum	iSX node number
brdnum	M3G daughter board number
channel	M3G channel number
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
pusrattr	The pointer of user-defined attributes

Completed Event

M3GEV_OPEN_CMPLT: The channel was successful opened.
M3GEV_OPEN_FAIL: The channel was unsuccessful opened

Warning

- Please do not use the operating system open() function to open the channel, otherwise unexpected results will occur.
- When parent process generates the child process, the child process can not inherit the channel device handle of the parent process.
- For normal daughter boards, the valid channel number is 0 to 63.
- The function does not specify the channel type, the application use the functions such as

ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245() to specify the channel type.

Error

The result code -1 for the function failed, please use ISX_sr_getlasterr() to get failure reasons.

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
    M3GDEV devh;          /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
       printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
       exit(1);
    }
    /*
    * Continue processing
    */
    //...
    if (ISX_m3g_Close( devh ) == -1 ) {
        printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

References

- **ISX_m3g_Close()**
- **ISX_sr_getlasterr()**

ISX_m3g_Close()

Function name:	int ISX_m3g_Close(dev)	
Parameters	**int dev**	Valid M3G channel device handle
Return value:	Success Failure	
Header file:	srllib.h m3glib.h	
Category:	Open/close	
Mode:	Synchronous	

Description

The application use the ISX_m3g_Close() function to close the M3G board device or channel device handle. The function does not change the device state (such as switch), and it only releases the handle and the occupied relation between the call application and the device.

Note: ISX_m3g_Close() forbids all events related to the device to occur.

Parameters	Description
dev	Specify an valid device handle (retrived by **ISX_m3g_OpenEx()**)

Warning

- Once the device is closed, the application of opening the device can not reuse the handle to operate the device.
- The application can not use the operating system close() function to close the device handle, otherwise unexpected results will occur.
- The ISX_m3g_Close() function will empty the event buffer of the closed device

Error

The result code -1 for the function failed, please use ISX_sr_getlasterr() to get failure reasons.

Example

Refer to the example of *ISX_m3g_OpenEx()*
Refer to the example of *ISX_m3g_Open()*

References

- **ISX_m3g_OpenEx()**
- **ISX_m3g_Open()**
- **ISX_sr_getlasterr()**

2.2. Configuration Functions

- ***ISX_m3g_SetParm()***

Function name:	int ISX_m3g_SetParm(dev, M3G_CHAN_PARAM *pParmInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_CHAN_PARAM *pParmInfo	The pointer of the M3G channel parameter information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	Asynchronous operation number
Return value:	0: Success -1: Failure	
Header file:	srllib.h m3glib.h	
Category:	Configuration functions	
Mode:	Synchronous and asynchronous	

Description

The application use the **ISX_m3g_SetParm()** function to set the M3G channel type and relevant parameters. The **ISX_m3g_OpenEx()** function only open the M3G channel device, and don't change the channel type. Once the channel type is specified, the channel type can not be changed. If the channel type indeed needs to be changed, the application must ensure that the channel is in the idle state.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pParmInfo	The pointer of the M3G channel parameter information structure; for the detailed description of parameters, please refer to the data structure **M3G_CHAN_PARAM**
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number. When the operation completed and the event M3GEV_SET_PARM_CMPLT or M3GEV_SET_PARM_FAIL occured, the application can use the ISX_sr_getevtoperindex() function to get the operation number.

Stop Event

M3GEV_SET_PARM_CMPLT: The function call completed successfully.
M3GEV_SET_PARM_FAIL: The function call completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fail.
- The function must be called on the channel is in the idle state, otherwise the function will fail.

Error

The result code -1 for the function fail. Please use the SRL function **ISX_ATDV_LASTERR()** to retrive the error code, or use the function **ISX_ATDV_ERRMSGP()** to retrive the error description. The function may return the following error codes:

EM3G_BADPARM: parameter error
EM3G_TIMEOUT: timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
        printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /*
    * Set m3g channel device param
     */
    M3G_CHAN_PARAM ParmInfo;
    ISX_m3g_Set2Default(M3G_CHAN_TYPE_H324M, &ParmInfo);
    ParmInfo.h324m.H223Params.ucValid = 1;
    ParmInfo.h324m.H223Params.ulAl2WithSeqNumberAudio = 0;
    armInfo.h324m.H223Params.ulAl2WitchSeqNumberVideo = 1;
    ParmInfo.h324m.H223Params.ulMaxPdu = 160;

    if ( ISX_m3g_SetParm(devh, &ParmInfo) == -1 ) {
        printf( "Cannot set m3g channel param. errmsg = %s",
                ISX_ATDV_ERRMSGP(devh) );
        exit(1);
    }
    /*
     .
     Continue processing
     .
    */


    if (ISX_m3g_Close( devh ) == -1 ) {
        printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

References

- **ISX_m3g_GetParm( )**
- **ISX_m3g_Set2Default( )**

▪ *ISX_m3g_GetParm()*

Function name:	int ISX_m3g_GetParm(dev, M3G_CHAN_PARAM *pParmInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_CHAN_PARAM *pParmInfo	The pointer of the M3G channel parameter information structure buffer
	unsigned short mode	Mode
	ULONG ulOperIndex	Asynchronous operation number
Return value:	0: Success -1: Failure	
Header file:	srllib.h m3glib.h	
Category:	Configuration function	
Mode:	Synchronous and asynchronous	

Description

The application use the **ISX_m3g_GetParm()** function to get the M3G channel type and relevant parameters. The retrived M3G channel parameter information will be written into the local buffer specified by **pParmInfo**.

Synchronous Operation

The function is in synchronous mode default. In synchronous mode, if the function is successfully completed, the function returns 0.

Asynchronous Operation

If the application set the **mode** to EV_ASYNC, then the function worked on asynchronous mode. On asynchronous mode, if the function call succeeds, the function completed immediately and the result code is 0. When the channel paramters are retrived, the event **M3GEV_GET_PARM_CMPLT** occured. The application can use the funciotn **ISX_sr_getevtdatap() to** obtain the event data and map the event data to the structure **M3G_CHAN_PARAM.** But if there are some errors in the retriving process, the **M3GEV_GET_PARM_FAIL** event will occur.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pParmInfo	The pointer of the M3G channel parameter information structure; please refer to **M3G_CHAN_PARAM** for detailed description.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number. When the operation completed and the event M3GEV_SET_PARM_CMPLT or M3GEV_SET_PARM_FAIL occured, the application can use the ISX_sr_getevtoperindex() function to get the operation number..

Stop Event

M3GEV_GET_PARM_CMPLT: The function call completed successfully.

M3GEV_GET_PARM_FAIL: The function call completed unsuccessfully.

Warning

- If the channel device handle is invalid, the function will fail.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description. The function may return the following error codes:

EM3G_BADPARM: parameter error
EM3G_TIMEOUT: timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_GetParm() function

```
#include <srllib.h>
#include <m3glib.h>
main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
        printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /*
    * Get m3g channel device param
     */
    M3G_CHAN_PARAM ParmInfo;
    if ( ISX_m3g_GetParm(devh, &ParmInfo) == -1 ) {
        printf( "Cannot get m3g channel param. errmsg = %s",
        ISX_ATDV_ERRMSGP(devh) );
        exit(1);
    }

    /*
     * Continue processing
     */
    //...


    if (ISX_m3g_Close( devh ) == -1 ) {
         printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

B> This example shows how to use the asynchronous mode of ISX_m3g_GetParm() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>
```

```
int getparm_handler();
M3G_CHAN_PARAM ParmInfo;

main()
{
     M3GDEV devh;           /* channel device handle */
     /*
     * Open m3g channel device
     */
     if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
          printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
          exit(1);
     }
     /*
     * Get m3g channel device param
      */

     if ( ISX_m3g_GetParm(devh, &ParmInfo, EV_ASYNC) == -1 ) {
          printf( "Cannot get m3g channel param. errmsg = %s",
          ISX_ATDV_ERRMSGP(devh) );
          exit(1);
     }
     ISX_sr_waitevt(-1);
     getparm_handler();

     /*
      .
      . Continue processing
      .
     */

}
int getparm_handler()
{
   int cnt, numdigs;

    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    M3G_CHAN_PARAM *pParmInfo = &ParmInfo;
    /*
    or M3G_CHAN_PARAM *pParmInfo = (M3G_CHAN_PARAM *)ISX_sr_getevtdatap();
    */

    switch(nEventType)
    {
     /*
      .
      . List of expected events
      .
     */
     //Expected reply to ISX_m3g_GetParm()
     case M3GEV_GET_PARM_CMPLT:
        printf("Received M3GEV_GET_PARM_CMPLT for the
               device,ulOperIndex=%u\n", ulOperIndex);
        /*
        .
        Process pParmInfo data;
        .
        */
        break;
```

```
    case M3GEV_GET_PARM_FAIL:
       printf("Received M3GEV_GET_PARM_FAIL for the device,
            ulOperIndex=%u\n", ulOperIndex);
       break;
    default:
       break;
    }

   /* Kick off next function in the state machine model. */
   /*
    .
    Continue processing
    .
    */
   return 0;
}
```

References

- **ISX_m3g_SetParm( )**

▪ *ISX_m3g_Set2Default()*

Function name:	int ISX_m3g_Set2Default(ucChanType, pParmInfo)	
Parameters:	**eM3G_CHAN_TYPE ucChanType**	M3G channel type
	M3G_CHAN_PARAM *pParmInfo	The pointer of the M3G channel parameter information structure
Return value	0: Success -1: Failure	
Header file	srllib.h m3glib.h	
Category:	Configuration function	
Mode：	Synchronous	

Description

The application use the **ISX_m3g_Set2Default( )** function to set each domain of **M3G_CHAN_PARAM** structure into the default value of SDK.

Parameters	Description
ucChanType	M3G channel type, please refer to **eM3G_CHAN_TYPE** for detailed information
pParmInfo	The pointer of the M3G channel parameter information structure; for the detailed description of parameters, please refer to the data structure **M3G_CHAN_PARAM**

Warning

None.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

Example

Please refer to the example of ISX_m3g_SetParm() function

References

- **ISX_m3g_SetParm()**

ISX_m3g_SetEvtMsk()

Function name:	int ISX_m3g_SetEvtMsk(dev, mask, action)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned long mask	Event bitmask
	int action	Action mark
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Configuration	
Mode：	Synchronous	

Description

The application use the function **ISX_m3g_SetEvtMsk()** to set the channel event mask. If the mask of an event is cleared, the event will not be sent to the application. Please refer to Table 1 for the mask type and the default value of the event.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
mask	Please refer to Table 1 for the event mask
action	Action marks: M3GACT_SETMSK: allows receiving the specified event, but forbids the unspecified events. M3GACT_ADDMSK: allows receiving the specified events. M3GACT_SUBMSK: does not allow receiving the specified events.

Table 1 Parameter Mask

Type	Description	Default value

M3GMM_DIGITS_RECEIVED	Set whether to receive the **M3GEV_DIGITS_RECEIVED** event	Disable
M3GMM_CST	Set whether to receive the **M3GEV_CST** event	Disable

Warning

- If the channel device handle is invalid, the function will fail.
- To change the **M3GEV_DIGITS_RECEIVED** or **M3GEV_CST** mask through this function will not influence the application gather digit in using the **ISX_m3g_GetDigits()** function.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The possible reasons are as follows:
EM3G_BADPARM: invalid parameters
EM3G_BUSY: device busy

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV dev;
   if ((m3gdev = ISX_m3g_OpenEx(&dev, 0, 0, 0)) == -1) {
      /* process error */
   }
    /* Set event mask to collect digits */
   if (ISX_m3g_SetEvtMsk(dev,M3GMM_DIGITS_RECEIVED) == -1) {
     /* error setting event */
   }
}
```

References

None.

2.3. Extended Functions

▪ *ISX_ATM3G_TERMMSK()*

Function name:	long ISX_ATM3G_TERMMSK(dev, OperType)	
Parameters:	**int dev**	Valid M3G channel device handle
	int OperType	Operation type
Return value	The last terminated reason (bitmask) or AT_FAILURE for function call fail	
Header file	srllib.h m3glib.h	
Category:	Extended function	
Mode：	Synchronous	

Description

The application use the **ISX_ATM3G_TERMMSK()** function to retrive the last I/O operation terminated reason.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
OperType	Operation type

Terminated reasons are as follows:

TM_DIGIT: receive the specific digits
TM_EOD: End of data (for example, the files playing finished, or the total voice data numbers were received during recording)
TM_ERROR: I/O device error
TM_IDDTIME: key-pressing interval timeout
TM_MAXDTMF: receive a specified number of DTMF
TM_MAXTIME: exceed the operation time specified by the function
TM_USRSTOP: the user terminate the operation
TM_BARGEIN: playing stopped for VAD
TM_MAXDATA: receive the data with specified length
TM_TIMEOUT: timeout

Warning

- If terminated conditions are met at the same time, the reason has more bits set at the same time

Error

The result code AT_FAILURE for the invalid device handle is specified.

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
```

```
    int chdev;
    ...
    /* Examine bitmap to determine if digits caused termination */
    if((term = ISX_ATM3G_TERMMSK(chdev)) == AT_FAILURE) {
      /* Process error */
    }
    if(term & TM_MAXDTMF) {
      printf("Terminated on digits\n");
       ...
    }
}
```

References

- **DV_TPT**

ISX_ATM3G_BUFDIGS()

Function name:	long ISX_ATM3G_BUFDIGS(dev)	
Parameters:	**int dev**	Valid M3G channel device handle
Return value	AT_FAILURE: failure Other: number of digits	
Header file	srllib.h m3glib.h	
Category:	Extended functions	
Mode：	Synchronous	

Description

The application use the **ISX_ATM3G_BUFDIGS()** function to get the number of digits in the digit buffer of VOICE channel. The digit buffer has at most 31 digits and 1 NULL terminator.

Parameters	Description
chdev	Valid channel device handle

Warning

Error

The result code AT_FAILURE for the invalid channel device handle is specified.

Example

```
#include <srllib.h>
#include <voclib.h>
main()
{
   int chdev; //VOICE channel type device handle
   int bufdigs;
   //...
   /* Check # of digits collected and continue processing. */
   if((bufdigs=ISX_ATM3G_BUFDIGS(chdev))==AT_FAILURE)  {
```

```
        /* process error */
      }
      ...
    }
```

References

- **ISX_dx_getdig()**

▪ *ISX_m3g_SetUsrAttr()*

Function name:	int ISX_m3g_SetUsrAttr(dev, usrattr)	
Parameters:	**int dev**	Valid M3G channel device handle
	void* usrattr	The pointer of user attribute
Return value	0 Success <0 Failure	
Header file	srllib.h m3glib.h	
Category:	Extended functions	
Mode:	Synchronous	

Description

The application use the function **ISX_m3g_SetUsrAttr()** to set the channel attribute (user-defined). The user attribute set by the user can be retrived through the **ISX_m3g_GetUsrAttr()** function.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
usrattr	User-defined attribute; the application can retrive the attribute through the **ISX_m3g_GetUsrAttr()** function.

Warning

None.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code.

Example

```
#include <srllib.h>
#include <m3glib.h>
#define MAXCH   120              /* max. number of channels in system */

/*
 * Data structure which stores all information for ipm channel
 */
struct chbag {
   M3GDEV   dev;                /* m3g channel handle */
   UCHAR    ucIsxNo;            /* ISX node no. */
```

```
   UCHAR   ucBrdNo;            /* m3g Board no. */
   USHORT  usCh;               /* m3g channel no. */
} m3gport[MAXCH+1];

int OpenM3gCh()
{
   for(int j=0; j<MAXCH; j++){
       m3gport[j].ucIsxNo = 0;
       m3gport[j].ucBrdNo = 0;
       m3gport[j].usCh = j;
       ISX_m3g_OpenEx(&m3gport[j].dev, 0, 0, j, EV_SYNC, NULL);
       if(m3gport[j].dev == -1){
          printf("open m3g channel fail.reason:%x\n",
                        ISX_sr_getlasterr());
          return(-1);
       }
       ISX_m3g_SetUsrAttr(m3gport[j].dev, (void*)&m3gport[j]);
    }
}
```

References

- **ISX_m3g_GetUsrAttr()**
- **ISX_m3g_OpenEx()**

ISX_m3g_GetUsrAttr()

Function name:	int ISX_m3g_GetUsrAttr(dev, usr_attrp)	
Parameters:	**int dev**	Valid M3G channel device handle
	void usr_attrp**	The pointer for saving the user attribute address
Return value	0 Success <0 Failure	
Header file	srllib.h m3glib.h	
Category:	Extended functions	
Mode：	Synchronous	

Description

The application use the function **ISX_m3g_GetUsrAttr()** to retrive the user-defined attributes (the user attributes set previously through the function **ISX_m3g_OpenEx()** and **ISX_m3g_SetUsrAttr()**).

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
usr_attrp	The pointer for saving the user attribute address

Warning

None.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code.

Example

```
#include <srllib.h>
#include <m3glib.h>
#define MAXCH   120              /* max. number of channels in system */
/*
 * Data structure which stores all information for each m3g channel
 */
struct chbag {
   M3GDEV   dev;                /* m3g channel handle */
   UCHAR    ucIsxNo;            /* ISX node no. */
   UCHAR    ucBrdNo;            /* m3g Board no. */
   USHORT   usCh;               /* m3g channel no. */
} m3gport[MAXCH+1];

int OpenM3gCh()
{
   for(int j=0; j<MAXCH; j++){
      m3gport[j].ucIsxNo = 0;
      m3gport[j].ucBrdNo = 0;
      m3gport[j].usCh = j;
      ISX_m3g_OpenEx(&m3gport[j].dev, 0, 0, j, EV_SYNC, (void*)&m3gport[j]);
      if(m3gport[j].dev == -1){
         printf("open m3g channel fail.reason:%x\n",
                     ISX_sr_getlasterr());
         return(-1);
      }
   }
}

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

    if ( ISX_m3g_GetUsrAttr( dev, (void**)&vattrp) != 0) {
       printf ("ISX_m3g_GetUsrAttr() fail.reason:%d\n”,
                    ISX_ATDV_LASTERR());
       return (-1);
    }
    printf("dev(%d)'s attr is %d:%d:%d\n",
          vattrp->ucIsxNo, vattrp->ucBrdNo, vattrp->usCh);
    return (0);
}
```

References

- **ISX_m3g_SetUsrAttr()**
- **ISX_m3g_OpenEx()**

2.4. Media Functions

- ***ISX_m3g_StartMedia()***

Function name:	int ISX_m3g_StartMedia(dev, pParmInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_CHAN_PARAM *pParmInfo	The pointer of the M3G channel parameter structure
	unsigned short mode	Mode
	ULONG ulOperIndex	Asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Media functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StartMedia()** to start the media of the specified M3G channel. The function is only applicable to the VOICE and VIDEO channels.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**()
pParmInfo	The pointer of the M3G channel parameter structure. Please refer to the structure **M3G_CHAN_PARAM** for detail.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number. When the operation is completed and the M3GEV_START_MEDIA_CMPLT or M3GEV_START_MEDIA_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_START_MEDIA_CMPLT: The function completed successfully.
M3GEV_START_MEDIA_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fail.
- The function set the channel type to VOICE or VIDEO type. the other functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>

int startmedia_handler();
main()
{
   ULONG ulOperIndex = 0;
   M3GDEV devh;           /* channel device handle */
   /*
   * Open m3g channel device
  */
   if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
            printf( "Cannot open m3g channel. errno = %d",
            ISX_sr_getlasterr() );
            exit(1);
   }
   /*
   * Set m3g channel device param
  */
   M3G_CHAN_PARAM ParmInfo;
   ISX_m3g_Set2Default(M3G_CHAN_TYPE_VOICE, &ParmInfo);
   ParmInfo.voice.SrcIpAddr.Valid = 1;
   strcpy(ParmInfo.voice.SrcIpAddr.IpAddr,"1.2.3.4");
   ParmInfo.voice.SrcRtpPort.Valid =1 ;
   ParmInfo.voice.SrcRtpPort.UdpPort = 9001;
   ParmInfo.voice.DstIpAddr.Valid = 1;
   strcpy(ParmInfo.voice.DstIpAddr.IpAddr,"5.6.7.8");
   ParmInfo.voice.DstRtpPort.Valid = 1 ;
   ParmInfo.voice.DstRtpPort.UdpPort = 9002;
   ParmInfo.voice.SrcRTCPPort.Valid = 1;
   ParmInfo.voice.SrcRTCPPort.UdpPort = 9003;
   ParmInfo.voice.DstRTCPPort.Valid = 1;
   ParmInfo.voice.DstRTCPPort.UdpPort = 9004;
   ParmInfo.voice.ucConnMode = 0;
   ParmInfo.voice.ucVoiceMode = 0;
   //...
   if( ISX_m3g_StartMedia(devh, &ParmInfo, EV_ASYNC, ulOperIndex)==-1 ) {
      printf( "Cannot StartMedia() for m3g channel. errmsg = %s\n",
             ISX_ATDV_ERRMSGP(devh) );
      /*
       .
       Perform Error Processing
       .
      */
   }
   ISX_sr_waitevt(-1);
   startmedia_handler();

   /*
   .
```

```
    Continue processing
  .
   */
}
int startmedia_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
     /*
      .
     . List of expected events
      .
     */
     //Expected reply to ISX_m3g_StartMedia()
     case M3GEV_START_MEDIA_CMPLT:
          printf("Received M3GEV_START_MEDIA_CMPLT for the
               device,ulOperIndex=%u\n", ulOperIndex);
          break;
      case M3GEV_START_MEDIA_FAIL:
           printf("Received M3GEV_START_MEDIA_FAIL for the device,
                       ulOperIndex=%u\n", ulOperIndex);
          break;
      default:
          break;
      }
}
```

References

- **ISX_m3g_Set2Default( )**
- **ISX_m3g_StopMedia()**

ISX_m3g_StopMedia()

Function name:	int ISX_m3g_StopMedia(dev, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned short mode	Mode
	ULONG ulOperIndex	Asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Media functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StopMedia()** to stop the media of the specified M3G channel. The function is only applicable to the VOICE and VIDEO M3G channels.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_STOP_MEDIA_CMPLT or M3GEV_STOP_MEDIA_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_STOP_MEDIA_CMPLT: The function completed successfully.
M3GEV_STOP_MEDIA_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE or VIDEO channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>

int stopmedia_handler();
void main()
{
    M3GDEV dev;
    /*
    .
    Main Processing
    .
    */
    /*
    Application needs to stop a current session on M3G device handle, dev
    ASSUMPTION: A valid dev was obtained from prior call to ISX_m3g_OpenEx().
    */
     ULONG ulOperIndex = 1;
    if(ISX_m3g_StopMedia(dev, EV_ASYNC, ulOperIndex) == -1)
```

```
    {
        printf("ISX_m3g_StopMedia failed for device\n");
        /*
        .
        Perform Error Processing
        .
        */
    }
    ISX_sr_waitevt(-1);
    stopmedia_handler();

    /*
    .
    . Continue Processing
    .
    */
}
int stopmedia_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
        /*
        .
        . List of expected events
        .
        */
        //Expected reply to ISX_ipm_Stop()
        case M3GEV_STOP_MEDIA_CMPLT:
            printf("Received M3GEV_STOP_MEDIA_CMPLT for the device,
                  ulOperIndex=%u\n", ulOperIndex);
            break;
        case M3GEV_STOP_MEDIA_FAIL:
            printf("Received M3GEV_STOP_MEDIA_FAIL for the device,
                  ulOperIndex=%u\n", ulOperIndex);
            break;
        default:
            break;
     }
}
```

References

- **ISX_m3g_StartMedia()**

2.5. I/O Functions

- ***ISX_m3g_Play()***

Function name:	int ISX_m3g_Play(dev, iottp, tptp, xpbp, eDirection, mode, pExChanInfo, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	DX_IOTT *iottp	pointer to I/O transfer table
	DV_TPT *tptp	pointer to the Termination Parameter Table
	M3G_XPB * xpbp	pointer to I/O transfer parameter block
	eM3G_DATA_DIR eDirection	Stream direction
	unsigned short mode	Play mode
	M3G_CHAN_EXINFO *pExChanInfo	Extension information
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_play()** to play audio and video files. The FileType domain of **M3G_XPB** specify the file formats of the file, and other domains of **M3G_XPB** specify the data format of the file. When the audio and video files need to be played at the same time or the video/audio transcoding needs to be carried out, please use **M3G_CHAN_EXINFO** structure to indicate the channel extension information. The audio and video files played by the function are stored in the M3G control server.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
iottp	Points to the I/O transfer table structure. I/O transfer tables composed of one or more **DX_IOTT** structures. One DX_IOTT structure identifies a source or destination for audio and video data. During the files playing, I/O transfer tables must be global or static. Please refer to *ISX4000 VOICE User Manual* for the detail of **DX_IOTT**. Note: It must point to a global or static variable, otherwise unexpected errors will occur.
tptp	Points to Termination Parameter Table structure; the Termination Parameter Table composed of one or more **DV_TPT** structures. The **DV_TPT** structure is used to specify all conditions of play terminated. Please refer to *ISX4000 VOICE User Manual* for the detail of **DV_TPT**. Note: It must point to a global or static variable, otherwise unexpected errors will occur.

xpbp	Points to I/O parameter block structure. The parameter specifies the file format and data formats, please refer to **M3G_XPB** for detail.
eDirection	Please refer to **eM3G_DATA_DIR** for stream direction Note: The stream direction of the H324M channel always is TDM direction (please use M3G_DATA_DIR_TDM instead).
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
pExChanInfo	Additional extension information is used only when the audio and video files need to be played at the same time or the video/audio transcoding needs to be carried out. Please refer to **M3G_CHAN_EXINFO** for the detail.
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_PLAY_CMPLT or M3GEV_PLAY_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_PLAY_BEGIN: The function completed successfully and the files begin to play.
M3GEV_PLAY_CMPLT: All files have played, so the playing are finished.
M3GEV_PLAY_FAIL: The function call completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE, VIDEO or H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.
- Iottp and tptp structures must be a global or static variable, otherwise unexpected errors will occur.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:

EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_play() function

```
/* Play a voice file. Terminate on receiving 4 digits or at end of file*/
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV   chdev;
   DX_IOTT  iott;
   DV_TPT   tpt;
```

```
   M3G_XPB  xpb;
   DV_DIGIT dig;
   char szPrompt[] = {“Prompt.vox”};

   /* Get M3G channel device handle in chdev.*/
   if (ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
     /* process error */
   }
   /*
   .
   Set M3G channel type to VOICE
   .
   */
   /* set up DX_IOTT */
   iott.io_type = IO_EOT;
   iott.io_bufp = szPrompt;
   iott.io_offset = 0;
   iott.io_length = -1;  /* play till end of file */
   iott.io_fhandle = 0;
   /* set up DV_TPT */
   ISX_dx_clrtpt(&tpt,1);
   tpt.tp_type   = IO_EOT;         /* only entry in the table */
   tpt.tp_termno = DX_MAXDTMF;      /* Maximum digits */
   tpt.tp_length = 4;             /* terminate on four digits */
   tpt.tp_flags  = TF_MAXDTMF;      /* Use the default flags */
   /* set up M3G_XPB */
   xpb.ucFileType=M3G_FILE_TYPE_WAV;
   xpb.wav.ulCoder = 0;
   xpb.wav.ulBlockSize = 5;
   /* Now play the file */
   if (ISX_m3g_Play(chdev,&iott,&tpt,&xpb, EV_SYNC) == -1) {
     /* process error */
   }
   /*
   .
   continue processing
   .
   */
}
```

B> This example shows how to use the asynchronous mode of ISX_dx_play() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>
#define MAXCHAN 128

int play_handler();
DX_IOTT prompt[MAXCHAN];
char    szPrompt[] = {“Prompt.vox”};
DV_TPT tpt;
DV_DIGIT dig;
main()
{
   M3GDEV chdev[MAXCHAN];
   int i,index, index1;

   /* initialize all the DX_IOTT structures for each individual prompt */
   //...
   /* For each channel, open the device using ISX_dx_open(), set up a DX_IOTT
```

```
   * structure for each channel,and issue ISX_dx_play()in asynchronous mode.
   */
   for (i=0; i<MAXCHAN; i++) {
      /* Open the device. chdev[i] has channel device handle*/
      if ( ISX_m3g_OpenEx(&chdev[i], 0, 0, i) == -1)   {
         /* process error */
      }
      /*
       .
        Set M3G channel type to VOICE;
       .
      */

      /* Set the DV_TPT structures up for MAXDTMF. Play until one digit is
          pressed or the file is played
      */
      tpt.tp_type   = IO_EOT;          /* only entry in the table */
      tpt.tp_termno = DX_MAXDTMF;       /* Maximum digits */
      tpt.tp_length = 1;               /* terminate on the first digit */
      tpt.tp_flags  = TF_MAXDTMF;       /* Use the default flags */
      prompt[i].io_type = IO_EOT; /* play from file */
      prompt[i].io_bufp = szPrompt;
      prompt[i].io_offset = 0;
      prompt[i].io_length = -1;         /* play till end of file */
      prompt[i].io_nextp = NULL;
      prompt[i].io_fhandle = 0;
      /* set up M3G_XPB */
      xpb.ucFileType=M3G_FILE_TYPE_WAV;
      xpb.wav.ulCoder = 0;
      xpb.wav.ulBlockSize = 5;
      /* play the data */
      if (ISX_m3g_Play(chdev,&iott,&tpt,&xpb, EV_ASYNC) == -1) {
          /* process error */
       }
   }
   ISX_sr_waitevt(-1);
   play_handler();
   /*
   .
   continue processing
   .
   */
}

int play_handler()
{
    long term;
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
        /*
        .
        . List of expected events
        .
        */
        //Expected reply to ISX_m3g_Play()
        case M3GEV_PLAY_BEGIN:
        printf("Received M3GEV_PLAY_BEGIN for the device,ulOperIndex=%u\n",
             ulOperIndex);
```

```
        break;
        case M3GEV_PLAY_CMPLT:
        printf("Received M3GEV_PLAY_CMPLT for the device,ulOperIndex=%u\n",
             ulOperIndex);

        /* Use ISX_ATM3G_TERMMSK() to get the reason for termination. */
        term = ISX_ATM3G_TERMMSK(chdev);
        if (term & TM_MAXDTMF) {
           printf("play terminated on receiving DTMF digit(s)\n");
        } else if (term & TM_EOD) {
           printf("play terminated on reaching end of data\n");
        } else {
           printf("Unknown termination reason: %x\n", term);
        }

        break;
        case M3GEV_PLAY_FAIL:
        printf("Received M3GEV_PLAY_FAIL for the device,ulOperIndex=%u\n",
             ulOperIndex);
        break;
        default:
        break;
     }

    /* Kick off next function in the state machine model. */
    ...
    return 0;
}
```

References

- **ISX_m3g_Record( )**
- **ISX_m3g_StopCh( )**
- **ISX_m3g_PauseCh( )**
- **ISX_m3g_ResumeCh( )**

ISX_m3g_Record()

Function name:	int ISX_m3g_Record(dev, iottp, tptp, xpbp, eDirection,mode, pExChanInfo, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	DX_IOTT *iottp	pointer to I/O transfer table
	DV_TPT *tptp	pointer to the Termination Parameter Table
	M3G_XPB * xpbp	pointer to I/O transfer parameter block
	eM3G_DATA_DIR eDirection	stream direction
	unsigned short mode	record mode
	M3G_CHAN_EXINFO *pExChanInfo	Extension information
	ULONG ulOperIndex	asynchronous operation number

Return value:	0 Success -1 Failure
Header file:	srllib.h m3glib.h
Category:	I/O function
Mode:	Synchronous and asynchronous

Description

The application use the function **ISX_m3g_ Record()** to record audio and video files. The FileType domain of **M3G_XPB** specify the file formats of the file, and other domains of **M3G_XPB** specify the data format of the file. When the audio and video files need to be recorded at the same time or the video/audio transcoding needs to be carried out, please use **M3G_CHAN_EXINFO** structure to indicate the channel extension information. The audio and video files recorded by the function are stored in the M3G control server.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
iottp	Points to the I/O transfer table structure. I/O transfer tables composed of one or more DX_IOTT structures. One DX_IOTT structure identifies a source or destination for audio and video data. During the files recording, I/O transfer tables must be global or static. Please refer to ISX4000 VOICE User Manual for the detail of DX_IOTT. Note: It must point to a global or static variable, otherwise unexpected errors will occur. Note: Recording only supports one IOTT now.
tptp	Points to Termination Parameter Table structure; the Termination Parameter Table composed of one or more DV_TPT structures. The DV_TPT structure is used to specify all conditions of play terminated. Please refer to ISX4000 VOICE User Manual for the detail of DV_TPT. Note: It must point to a global or static variable, otherwise unexpected errors will occur.
xpbp	Points to I/O parameter block structure. The parameter specifies the file format and data formats, please refer to M3G_XPB for detail.
eDirection	Please refer to **eM3G_DATA_DIR** for stream direction Note: The stream direction of the H324M channel always is TDM direction (please use M3G_DATA_DIR_TDM instead).
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
pExChanInfo	Additional extension information is used only when the audio and video files need to be recorded at the same time or the video/audio transcoding needs to be carried out. Please refer to **M3G_CHAN_EXINFO** for the details.
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_RECORD_CMPLT or M3GEV_RECORD_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_RECORD_BEGIN: The function completed successually and the file beginn to record.

M3GEV_RECORD_CMPLT: The file recording finished.
M3GEV_RECORD_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE, VIDEO or H324M channel if using the function, The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.
- Iottp and tptp parameters must point to a global variable, otherwise unexpected errors will occur.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_Record() function

```
#include <srllib.h>
#include <m3glib.h>
#define MAXLEN 10000
main()
{
   DV_TPT tpt;
   DX_IOTT iott;
   M3G_XPB xpb;
   M3GDEV chdev;
   char szFileName1[]={“file1.vox”};
   /*
    * open the channel
    */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
     /* process error */
   }
   /*
   .
   Set M3G channel type to VOICE
   .
   */

   /*
    * Set up the DV_TPT structures for MAXDTMF
    */
   ISX_dx_clrtpt(&tpt,1);
   tpt.tp_type   = IO_EOT;         /* last entry in the table */
   tpt.tp_termno = DX_MAXDTMF;      /* Maximum digits */
   tpt.tp_length = 1;             /* terminate on the first digit */
   tpt.tp_flags  = TF_MAXDTMF;      /* Use the default flags */
   /*
    * Set up the DX_IOTT. The application records the voice data to file
```

```
    */
   iott.io_type = IO_EOT;
   iott.io_bufp = szFileName1;
   iott.io_offset = 0;            /* Start at beginning of file */
   iott.io_length = MAXLEN;    /* Record 10,000 bytes of voice data */
   iott.io_fhandle = 0;
   /* set up M3G_XPB */
   xpb.ucFileType=M3G_FILE_TYPE_WAV;
   xpb.wav.ulCoder = 0;
   xpb.wav.ulBlockSize = 5;

   if (ISX_m3g_Record(chdev,&iott,&tpt,&xpb,EV_SYNC) == -1) {
     /* process error */
   }
   /* Analyze the data recorded */
   /*
   .
   continue processing
   .
   */
}
```

B> This example shows how to use the asynchronous mode of ISX_m3g_Record() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>
#define MAXLEN 10000
#define MAXCHAN 128
int record_handler();
DV_TPT tpt;
DX_IOTT iott[MAXCHAN];
M3GDEV chdev[MAXCHAN];
char FileName[MAXCHAN][MAX_PATH];
main()
{
   int i;

   /* Start asynchronous record on all the channels. */
   for (i=0; i<MAXCHAN; i++) {
      /* open the channel */
      if ( ISX_m3g_OpenEx(&chdev[i], 0, 0, i) == -1) {
        /* process error */
      }
      /*
       .
       Set M3G channel type to VOICE
       .
      */

     /*
      * Set up the DV_TPT structures for MAXDTMF
      */
      tpt.tp_type   = IO_EOT;         /* last entry in the table */
      tpt.tp_termno = DX_MAXDTMF;      /* Maximum digits */
      tpt.tp_length = 1;             /* terminate on the first digit */
      tpt.tp_flags  = TF_MAXDTMF;      /* Use the default flags */
     /*
      * Set up the DX_IOTT. The application records the voice data to the
     */
      iott[i].io_type = IO_EOT;
      sprintf(FileName[i], “RecFile%d.vox”, i);
```

```
     iott[i].io_bufp = FileName[i];
     iott[i].io_offset = 0;          /* Start at beginning of file */
     iott[i].io_length = MAXLEN;     /* Record 10,000 bytes voice data */
     iott[i].io_fhandle = 0;
     /* set up M3G_XPB */
     xpb.ucFileType=M3G_FILE_TYPE_WAV;
     xpb.wav.ulCoder = 0;
     xpb.wav.ulBlockSize = 5;

     /* Start asynchronous record on the channel */
     if (ISX_m3g_Record(chdev[i],&iott[i],&tpt,&xpb,EV_ASYNC) == -1) {
       /* process error */
     }
   }
   ISX_sr_waitevt(-1);
   record_handler();
   /*
   .
   continue processing
   .
   */
}
int record_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
        /*
        .
        . List of expected events
        .
       */
       //Expected reply to ISX_m3g_Record()
        case M3GEV_RECORD_BEGIN:
        printf("Received M3GEV_RECORD_BEGIN for the device,ulOperIndex=%u\n",
               ulOperIndex);
        break;
        case M3GEV_RECORD_CMPLT:
        printf("Received M3GEV_RECORD_CMPLT for the device,ulOperIndex=%u\n",
               ulOperIndex);
        long term;
        /* Use ISX_ATDX_TERMMSK() to get the reason for termination. */
        term = ISX_ATDX_TERMMSK(ISX_sr_getevtdev());
        if (term & TM_MAXDTMF) {
           printf("record terminated on receiving DTMF digit(s)\n");
        } else if (term & TM_NORMTERM) {
           printf("normal termination of ISX_m3g_Record()\n");
        } else {
           printf("Unknown termination reason: %x\n", term);
        }
        break;
        case M3GEV_RECORD_FAIL:
        printf("Received M3GEV_RECORD_FAIL for the device,ulOperIndex=%u\n",
               ulOperIndex);
        break;
        default:
        break;
    }
```

```
    /* Kick off next function in the state machine model. */
    /*
    .
    continue processing
    .
    */
    return 0;
}
```

References

- **ISX_m3g_Play( )**
- **ISX_m3g_StopCh( )**
- **ISX_m3g_PauseCh( )**
- **ISX_m3g_ResumeCh( )**

▪ *ISX_m3g_StopCh()*

Function name:	int ISX_m3g_StopCh(dev, mode)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned short mode	Mode
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StopCh()** to forces termination of currently active I/O functions on a M3G channel. If the specified channel has no active I/O functions, there is no action on the channel. After the active I/O functions were terminated, the terminated reason (TM_USRSTOP) can be obtained through the ISX_ATDX_TERMMSK() function.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
mode	**Call mode:** EV_SYNC - synchronous EV_ASYNC - asynchronous **I/O mode:** EV_STOPPLAY: terminate playing action EV_STOPREC: terminate recording action EV_STOPGETDIG: terminate receiving DIGIT action **Mode** can be the combination of **call mode** and **I/O mode** (Bitwise OR)

Terminated Event

M3GEV_PLAY_CMPLT: After the function completed, file playing action was forcibly terminated
M3GEV_RECORD_CMPLT: After the function completed, file recording action was forcibly terminated.
M3GEV_RECV_DIGITS_CMPLT: After the function completed, the DIGIT receiving action is forcibly stopped.
M3GEV_STOP_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- If the I/O actions had completed due to other reasons before the **ISX_m3g_StopCh()** function called, the terminated reasons will not reflect the call of **ISX_m3g_StopCh()**.
- The function does not support EV_STOPALL mode.
- The channel type must be VOICE, VIDEO or H324M channel if using the function, The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;
   /* Open the channel . Get channel handle in chdev.
    */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
    .
    continue processing
    .
   */

   if ( ISX_m3g_StopCh(chdev, EV_ASYNC|EV_STOPPLAY) == -1) {
      /* process error */
   }
}
```

References

- **ISX_m3g_Play( )**

- **ISX_m3g_Record( )**

ISX_m3g_PauseCh()

Function name:	int ISX_m3g_PauseCh(dev, mode)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned short mode	Mode
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_PauseCh()** to suspend the I/O actions of a M3G channel. If the specified channel has no I/O actions, there is no any influence on the channel. After the I/O actions were suspended, the **M3GEV_PAUSE_CMPLT** or **M3GEV_PAUSE_FAIL** event will occur. After the event is received, the currently suspended operation types (OPER_PLAY and OPER_RECORD) can be obtained through **ISX_sr_getevtopertype()** function.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**()
mode	**Call mode:** EV_SYNC - synchronous EV_ASYNC - asynchronous **I/O mode:** EV_ PAUSEPLAY - suspend the file playing EV_ PAUSEREC - suspend the file recording **Mode** can be the combination of **call mode** and **I/O mode** (Bitwise OR)

Stop Event

M3GEV_PAUSE_CMPLT : The function completed successfully.
M3GEV_PAUSE_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE, VIDEO or H324M channel if using the function, The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;
   /* Open the channel . Get channel handle in chdev.
    */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
   .
   continue processing
   .
   */
   if ( ISX_m3g_PauseCh(chdev, EV_ASYNC|EV_PAUSEPLAY) == -1) {
      /* process error */
   }
}
```

References

- **ISX_m3g_Play( )**
- **ISX_m3g_Record( )**
- **ISX_m3g_ResumeCh( )**

ISX_m3g_ResumeCh()

Function name:	int ISX_m3g_ResumeCh(dev, mode)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned short mode	Mode
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_ ResumeCh()** to resume the suspended I/O actions of a M3G channel. If the specified channel has no I/O actions suspended, there is no any influence on the channel. After the I/O actions are resumed, the **M3GEV_RESUME_CMPLT** or **M3GEV_RESUME_FAIL** event will occur. After the event is received, the currently suspended operation types (OPER_PLAY and OPER_RECORD) can be obtained through **ISX_sr_getevtopertype()** function.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
mode	**Call mode:** EV_SYNC - synchronous EV_ASYNC - asynchronous **I/O mode:** EV_ RESUMEPLAY - resume the file playing EV_ RESUMEREC - resume the file recording **Mode** can be the combination of **call mode** and **I/O mode** (Bitwise OR)

Stop Event

M3GEV_RESUME_CMPLT: The function completed successfully.
M3GEV_RESUME_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE, VIDEO or H324M channel if using the function, The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;
   /*
    * Open the channel . Get channel handle in chdev.
   */
```

```
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
   .
   continue processing
   .
   */
   if ( ISX_m3g_ResumeCh(chdev, EV_ASYNC|EV_RESUMEPLAY) == -1) {
      /* process error */
   }
}
```

References

- **ISX_m3g_Play( )**
- **ISX_m3g_Record( )**
- **ISX_m3g_PauseCh( )**

ISX_m3g_SendDigits()

Function name:	int ISX_m3g_SendDigits(dev, pDigitInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_DIGIT_INFO *pDigitInfo	The pointer of the DIGITS information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_SendDigits()** to send DIGIT with the specified M3G channel. When the sending direction is **DIGIT_H324M** and the DIGIT type is **DIGIT_ALPHA_NUMERIC**, the **pDigitInfo** structure contain the sending contents and the SDK will not convert before sending. Otherwise, the SDK will convert the sending content with the following table before sending.

DIGIT (character)	Value after conversion (hex)
'0'	0x00
'1'	0x01
'2'	0x02
'3'	0x03
'4'	0x04
'5'	0x05
'6'	0x06

'7'	0x07
'8'	0x08
'9'	0x09
'*'	0x0a
'#'	0x0b
'a' or 'A'	0x0c
'b' or 'B'	0x0d
'c' or 'C'	0x0e
'd' or 'D'	0x0f

Note: The above table is valid for the **DIGIT_TDM or DIGIT_IP** direction. And for the **DIGIT_IP** direction, DIGIT can only be sent one by one.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pDigitInfo	The pointer of the DIGITS information structure. Please refer to **M3G_DIGIT_INFO** for the details
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_SEND_DIGITS_CMPLT or M3GEV_SEND_DIGITS_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_SEND_DIGITS_CMPLT: The function completed successfully and all the digits sent.
M3GEV_SEND_DIGITS_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- The channel type must be VOICE, VIDEO or H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.
- When the sending direction specified by **M3G_DIGIT_INFO** is DIGIT_IP, DIGIT can only be sent one by one, and the receiving will not limited by this.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:

EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
M3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>

int senddigits_handler();
main()
{
   M3GDEV chdev;
   /*
    *Open the channel . Get channel handle in chdev.
   */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
   .
     Continue processing
  .
   */
   M3G_DIGIT_INFO DigitInfo;
   DigitInfo.ucDigitType = DIGIT_ALPHA_NUMERIC;
   DigitInfo.ucDigitDir= DIGIT_H324M;
   DigitInfo.ucDigitNum= 5;
   strcpy(DigitInfo.Digits, "12345");
   DigitInfo.ucDuration = 10;
   DigitInfo.ucInterval = 5;
   if (ISX_m3g_SendDigits(chdev, &DigitInfo, EV_ASYNC) == -1) {
      /* process error */
   }
   ISX_sr_waitevt(-1);
   senddigits_handler();

   /*
   .
     Continue processing
  .
   */

}

int senddigits_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
     /*
      .
      . List of expected events
      .
     */
     //Expected reply to ISX_m3g_SendDigits()
     case M3GEV_SEND_DIGITS_CMPLT:
          printf("Received M3GEV_SEND_DIGITS_CMPLT for the
               device,ulOperIndex=%u\n", ulOperIndex);
          break;
      case M3GEV_SEND_DIGITS_FAIL:
           printf("Received M3GEV_SEND_DIGITS_FAIL for the device,
                       ulOperIndex=%u\n", ulOperIndex);
          break;
      default:
```

```
            break;
        }
}
```

References

- **ISX_m3g_GetDigits( )**

▪ *ISX_m3g_GetDigits()*

Function name:	int ISX_m3g_GetDigits(dev, tptp, digitp, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	DV_TPT *tptp	pointer to the Termination Parameter Table
	DV_DIGIT * digitp	pointer to the DIGITS information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	I/O function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetDigits()** to gather digits of a M3G channel. After the action terminated, the gathered digits were written into the local buffer specified by **digitp** in ASCIIZ format. At present, the digits type only is DTMF and can not be other TONEs(such as MFs and user-defined TONEs). The action of gathering DIGITs can be stopped through **ISX_m3g_StopCh()** function.

Synchronous Operation

On default, the function called with synchronous mode. In synchronous mode, when the action was successfully completed, the result code is an integer and greater than 0, and the result code indicates the number of receiving digits (including NULL characters, therefore it is necessary to subtract 1 from the actual number of receiving digits); when the result code is 1, there are not any digit received at all, and the number 1 is just the terminative NULL character; the stop reasons can be obtained through **ISX_ATM3G_TERMMSK()** function.

Asynchronous Operation

If you want to call the function with asynchronous mode, you must set the **mode** to EV_ASYNC; in asynchronous mode, if the function call succeeds, the function immediately returns 0; when the action is completed, the **M3GEV_GET_DIGITS_CMPLT** event will occur; when an error occurs during receiving DIGITS data, the **M3GEV_GET_DIGITS_FAIL** event will occur. When the action is completed, the stop reasons can be obtained through **ISX_ATM3G_TERMMSK()** function.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )

tptp	Points to Termination Parameter Table structure; the Termination Parameter Table composed of one or more DV_TPT structures. The DV_TPT structure is used to specify all conditions of play terminated. Please refer to ISX4000 VOICE User Manual for the detail of DV_TPT.
digitp	The pointer pointing to DIGITS information structure; please refer to *ISX4000 VOICE User Manual* for the detailed information of **DV_DIGIT.**
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_GET_DIGITS_CMPLT or M3GEV_GET_DIGITS_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_GET_DIGITS_CMPLT: The event indicates that the DIGITS data is received.
M3GEV_GET_DIGITS_FAIL: The function completed unsuccessfully.

Warning

- If the specified device handle is invalid, the function will fails.
- During the action of gathering Digit, **digitp** must be maintained in its scope.
- The channel type must be VOICE, VIDEO or H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.
- The channel digit buffer can only save 31 digits, and are the FIFO; When the digit buffer is full, the subsequent digits will be lost; the application can call **ISX_m3g_ClrDigBuf()** function to empty the digit buffer. In the voclib.h header file, a macro DG_MAXDIGS (31) is defined, it is the DG_MAXDIGS digits that the **ISX_m3g_GetDigits()** function can obtain at one time.
- Tptp parameters must point to a global variable, otherwise unexpected errors will occur.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_GetDigits() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>
#include <voxlib.h>
main()
{
```

```
   DV_TPT tpt[2];
   DV_DIGIT digp;
   M3GDEV chdev;
   int numdigs, cnt;
   /*
    *open the channel . Obtain channel device handle in chdev
   */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
     /* process error */
   }
   /* initiate the call */
   /*
   .
   Continue processing
   .
   */
   /* Set up the DV_TPT and get the digits */
   ISX_dx_clrtpt(tpt,2);  // Refer to ISX4000 VOICE User Manual
   tpt[0].tp_type   = IO_CONT;
   tpt[0].tp_termno = DX_MAXDTMF;   /* Maximum number of digits */
   tpt[0].tp_length = 4;            /* terminate on 4 digits */
   tpt[0].tp_flags  = TF_MAXDTMF;   /* terminate if already in buf. */
   tpt[1].tp_type   = IO_EOT;
   tpt[1].tp_termno = DX_MAXTIME;  /* Function Time */
   tpt[1].tp_length = 100;   /* 10 seconds (100 msec resolution timer) */
   tpt[1].tp_flags  = TF_MAXTIME;   /* Edge-triggered */
   /* clear previously entered digits */
   if (ISX_m3g_ClrDigBuf(chdev) == -1) {
     /* process error */
   }
   if ((numdigs = ISX_m3g_GetDigits(chdev,tpt, &digp, EV_SYNC)) == -1) {
     /* process error */
   }
   for (cnt=0; cnt < numdigs; cnt++) {
      printf("\nDigit received = %c, digit type = %d",
               digp.dg_value[cnt], digp.dg_type[cnt]);
   }
   /* go to next state */
   /*
   .
   Continue processing
   .
   */
}
```

B> This example shows how to use the asynchronous mode of ISX_m3g_GetDigits() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>

int digit_handler();
DV_TPT tpt[2];
DV_DIGIT digp;

main()
{
     M3GDEV chdev;
     /* Open the channel. Obtain channel device handle in chdev */
     if (ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
         /* process error */
     }
```

```
     /* initiate the call */
     /*
      .
      Continue processing
      .
     */
     /* Set up the DV_TPT and get the digits */
     ISX_dx_clrtpt(tpt,2);  // Refer to ISX4000 VOICE User Manual

     tpt[0].tp_type  = IO_CONT;
     tpt[0].tp_termno = DX_MAXDTMF;      /* Maximum number of digits */
     tpt[0].tp_length = 4;               /* terminate on 4 digits */
     tpt[0].tp_flags  = TF_MAXDTMF;      /* terminate if already in buf*/
     tpt[1].tp_type  = IO_EOT;
     tpt[1].tp_termno = DX_MAXTIME;       /* Function Time */
     tpt[1].tp_length = 100;     /* 10 seconds (100 msec resolution timer) */
     tpt[1].tp_flags  = TF_MAXTIME;       /* Edge triggered */
     /* clear previously entered digits */
     if (ISX_m3g_ClrDigBuf(chdev) == -1) {
        /* process error */
     }
     if (ISX_m3g_GetDigits(chdev, tpt, &digp, EV_ASYNC) == -1) {
        /* process error */
     }
     ISX_sr_waitevt(-1);
     digit_handler();
     /*
     .
     Continue processing
     .
     */
}
int digit_handler()
{
   int cnt, numdigs;

    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    DV_DIGIT* pDigit = &digp;

    switch(nEventType)
    {
      /*
      .
      . List of expected events
      .
     */
     //Expected reply to ISX_m3g_GetDigits()
     case M3GEV_GET_DIGITS_CMPLT:
         printf("Received M3GEV_GET_DIGITS_CMPLT for the
                device,ulOperIndex=%u\n", ulOperIndex);
         numdigs = strlen(pDigit->dg_value);
         for(cnt=0; cnt < numdigs; cnt++) {
            printf("\nDigit received = %c, digit type = %d",
                  pDigit->dg_value[cnt], pDigit->dg_type[cnt]);
         }
         break;
     case M3GEV_GET_DIGITS_FAIL:
         printf("Received M3GEV_GET_DIGITS_FAIL for the device,
               ulOperIndex=%u\n", ulOperIndex);
```

```
        break;
        default:
        break;
    }

   /* Kick off next function in the state machine model. */
   /*
    .
      Continue processing
      .
    */
   return 0;
}
```

References

- **ISX_m3g_ClrDigBuf( )**
- **ISX_m3g_SendDigits( )**

ISX_m3g_ClrDigBuf()

Function name:	int ISX_m3g_ClrDigBuf(dev)	
Parameters:	**int dev**	Valid M3G channel device handle
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	I/O functions	
Mode：	Synchronous	

Description

The application use the function **ISX_m3g_ClrDigBuf()** to clear the digits buffer of the channel clearly.

Parameters	Description
dev	Valid M3G channel device handle (retrived by **ISX_m3g_OpenEx()**)

Warning

- If the channel handle is invalid or the channel is in busy state, the function returns -1 to prompt failure.
- The channel type must be VOICE, VIDEO or H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;    /* channel handle */

   /* Open Channel */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1 ) {
     /* process error */
   }
   /* Clear digit buffer */
   if (ISX_m3g_ClrDigBuf(chdev) == -1) {
     /* process error*/
   }
   /*
   .
   Continue processing
   .
   */
}
```

References

- **ISX_m3g_GetDigits( )**

2.6. Video Editing functions

- ***ISX_m3g_DownloadFont()***

Function name:	INT ISX_m3g_DownloadFont(ucIsxNo,ucBrdNo,szFileName,tTimeout, mode, ulOperIndex)	
Parameters:	**UCHAR ucIsxNo**	Valid iSX node number
	UCHAR ucBrdNo	Valid M3G board number
	char* szFileName	Points to the font file name that will be downloaded
	int tTimeout	timeout time during downloading
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode：	Synchronous and asynchronous	

Description

For the text-overlay functions, the application can select using the built-in ttf or the customized ttf that downloaded on the M3G board. The application use the function **ISX_m3g_DownloadFont()** to download the TrueType font file of the customer to the specified M3G board.

Parameters	Description
ucIsxNo	Valid iSX node number, range: 0~ MAX_NODE_NUM
ucBrdNo	Valid M3G board number, range: 0~ MAX_M3G_BRD
szFileName	Points to the font file name.
tTimeout	timeout time during downloading, the default is -1 that the application will wait the file download completed. Unit: ms
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the font download is completed and the event **M3GEV_DOWNLOADFONT_CMPLT** occurs.

Stop Event

M3GEV_DOWNLOADFONT_CMPLT: The font download successfully.

M3GEV_DOWNLOADFONT_FAIL: The font download unsuccessfully.

Warning

- If the specified iSX node number and board number are invalid, the function fails.

Error

The result code -1 for the function fail. Please use the ISX_sr_getlasterr() to get failure reasons.

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"

UCHAR ucIsxNo = 0;
UCHAR ucBrdNo = 0;
CHAR szFontFile[] = {"MicrosoftYaHei.ttf"};

if (ISX_m3g_DownloadFont(ucIsxNo, ucBrdNo, szFontFile, -1, EV_ASYNC) == -1)
{
    printf( "Cannot DownloadFont() for m3gbrd. errcode = %s",
    ISX_sr_getlasterr());
    /* process error */
    return -1;
}
INT iRetVal = 0;
while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(DownloadFont_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int DownloadFont_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    //Expected reply to ISX_m3g_DownloadFont()
    case M3GEV_DOWNLOADFONT_CMPLT:
        printf("Received Download font success for the
        device,ulOperIndex=%u\n", ulOperIndex);
        iRetVal = 0;
        break;
    case M3GEV_DOWNLOADFONT_FAIL:
        printf("Received Download font fail for the device, ulOperIndex=%u\n",
        ulOperIndex);
        iRetVal = 0;
```

```
            break;

        default:
            break;
        }
        return iRetVal;
    }
```

- ***ISX_m3g_CtrlToolbox()***

Function name:	INT ISX_m3g_CtrlToolbox(dev, ucToolboxType, ucCtrlCode, pConfig, pszLogoFileName, mode, ulOperIndex)	
Parameters:	**int dev**	Valid channel device handle
	UCHAR ucToolboxType	Toolbox type
	UCHAR ucCtrlCode	Toolbox control code
	M3G_TOOLBOX_CONFIG *pConfig	Points to toolbox configuration information
	char *pszLogoFileName	Points to LOGO file name
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Conference management	
Mode：	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_CtrlToolbox()** to configure the toolbox of video channel. Before use the function, the application must start the video media of the channel (namely, the function ISX_m3g_StartMedia() must be successful).

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
ucCtrlCode	Please refer to **eM3G_TOOLBOX_CTRLCODE** for the toolbox control code
pConfig	The toolbox configuration information is specified by **M3G_TOOLBOX_CONFIG** structure.
pszLogoFileName	When the control code has M3G_VIDEO_LOGO_DOWNLOAD, pszLogoFileName is bitmap file name of the LOGO that will download to the toolbox and the LOGO file name can not exceed 256 bytes. On other control code, the pointer can be NULL.

mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the control toolbox is completed and the event **M3GEV_CTRLTOOLBOX_CMPLT** occurs.

Stop Event

M3GEV_CTRLTOOLBOX_CMPLT: The toolbox configured successfully.
M3GEV_CTRLTOOLBOX_FAIL: The toolbox configured unsuccessfully.

Warning

- If the device handle is not the video channel handle, the function fails.
- If the video media don't stared of the channel with the function ISX_m3g_StartMedia(), the function fails.
- The function configures the primary toolbox or mixer toolbox of the video channel. Only after the video channel was added into the video mixer, the mixer toolbox can be configured.
- If the control code has M3G_VIDEO_LOGO_DOWNLOAD, FileName must be provided; in other cases, the file name is unmeaningful.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
    #include "IsxApi.h"
    #include "srllib.h"
    #include "m3glib.h"

    M3GDEV m3gdev;
    /*
    * Open m3gdev...
    */

    UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;
    UCHAR ctrlcode   = M3G_CTRLCODE_LOGO_DOWNLOAD | M3G_CTRLCODE_LOGO_DISPLAY |
M3G_CTRLCODE_TEXT_OVERLAY;
    M3G_TOOLBOX_CONFIG ToolBoxCfg;
    memset(&ToolBoxCfg, 0, sizeof(ToolBoxCfg));
    M3G_TOOLBOX_PARAM *pToolbox = &ToolBoxCfg.toolbox;
    pToolbox->ucValid = 1;
    pToolbox->ucIsEnable = 1;
    pToolbox->ucFPS = 10;
    pToolbox->ulInputHeight = 144;
    pToolbox->ulInputWidth = 176;
    pToolbox->ulOutputHeight = 144;
    pToolbox->ulOutputWidth = 176;
```

```
    M3G_LOGO_PARAM *pLogoParm = &ToolBoxCfg.logo;
    pLogoParm->ucValid = 1;
    pLogoParm->ulLeftPos = 10;
    pLogoParm->ulTopPos = 20;
    pLogoParm->ulWidth = 76;
    pLogoParm->ulHeight = 40;
    pLogoParm->ucTransY = 0;
    pLogoParm->ucTransU = 0;
    pLogoParm->ucTransV = 0;
    pLogoParm->ucTransAlpha = 0;
    pLogoParm->ucAlphaBlending = 100;

    CHAR szLogoFile[] = {"surf_logo_76x40.bmp"};

    if (ISX_m3g_CtrlToolbox(m3gdev,tb_type,ctrlcode,&ToolBoxCfg, szLogoFile,
EV_ASYNC)==-1){
        printf( "Cannot GetLCParm() for m3g channel. errmsg = %s",
        ISX_ATDV_ERRMSGP(m3gdev) );
        return -1;
    }
    INT iRetVal = 0;
    while(1) {
        iRetVal=ISX_sr_waitevt(-1);
        if(iRetVal==0) {
            if(CtrlToolbox_handler()==0) break;
        }
        apr_sleep(10*1000);
    }

    int CtrlToolbox_handler()
    {
        int iRetVal = -1;
        int nEventType = ISX_sr_getevttype();
        int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
        switch(nEventType)
        {
        /*
        .
        . List of expected events
        .
        */
        //Expected reply to ISX_m3g_CtrlToolbox()
        case M3GEV_CTRLTOOLBOX_CMPLT:
            printf("Received CtrlToolbox success for the device,ulOperIndex=%u\n",
            ulOperIndex);
            iRetVal = 0;
            break;
        case M3GEV_CTRLTOOLBOX_FAIL:
            printf("Received CtrlToolbox fail for the device, ulOperIndex=%u\n",
            ulOperIndex);
            iRetVal = 0;
            break;

        default:
            break;
        }
```

```
    return iRetVal;
}
```

- ***ISX_m3g_DownloadLogo()***

Function name:	INT ISX_m3g_DownloadLogo(dev, ucToolboxType, pszFileName, pConfig, mode, ulOperIndex)	
Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	CHAR *pszFileName	Points to LOGO file name
	M3G_LOGO_CONFIG *pConfig	Points to LOGO configuration information
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode：	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_DownloadLogo( )** to download the LOGO bitmap file to the toolbox of the VIDEO channel. Before use the function, the application must start the video media of the channel (namely, the function ISX_m3g_StartMedia() must be successful).

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
pszFileName	Points to the LOGO bitmap filename. The length of LOGO file name can not exceed 256 bytes.
pConfig	LOGO configuration information is specified by **M3G_LOGO_CONFIG** structure. When it is NULL, only download the LOGO file to toolbox but not change the configuration of toolbox. After the file downloading is completed, the LOGO will displayed on default.
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous (default) EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the LOGO parameter configuring is completed and the event **M3GEV_SETLOGOPARM_CMPLT** occurs.

Stop Event

M3GEV_DOWNLOADLOGO_CMPLT: The LOGO file download successfully.
M3GEV_DOWNLOADLOGO_FAIL: The LOGO file download unsuccessfully.

Warning

- If the device handle is not the video channel handle, the function fails.
- If the video media of the channel does not started with the function ISX_m3g_StartMedia(), the function fails.
- Only when the video channel was added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"

M3GDEV m3gdev;
/*
* Open m3gdev...
*/

UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;

M3G_LOGO_CONFIG LogoCfg;
memset(&LogoCfg, 0, sizeof(LogoCfg));
LogoCfg.LogoParm.ucValid = 1;
LogoCfg.LogoParm.ulLeftPos = 10;
LogoCfg.LogoParm.ulTopPos = 20;
LogoCfg.LogoParm.ulWidth = 76;
LogoCfg.LogoParm.ulHeight = 40;
LogoCfg.LogoParm.ucTransY = 0;
LogoCfg.LogoParm.ucTransU = 0;
LogoCfg.LogoParm.ucTransV = 0;
LogoCfg.LogoParm.ucTransAlpha = 0;
LogoCfg.LogoParm.ucAlphaBlending = 100;

CHAR szLogoFile[] = {"surf_logo_76x40.bmp"};

if (ISX_m3g_DownloadLogo(m3gdev,tb_type,szLogoFile,&LogoCfg,
    EV_ASYNC)==-1){
    printf( "Cannot DownloadLogo() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m3gdev) );
    return -1;
}
```

```
INT iRetVal = 0;
while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(DownloadLogo_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int DownloadLogo_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
        /*
        .
        . List of expected events
        .
        */
        //Expected reply to ISX_m3g_DownloadLogo()
        case M3GEV_DOWNLOADLOGO_CMPLT:
            printf("Received download logo success for the
            device,ulOperIndex=%u\n", ulOperIndex);
            iRetVal = 0;
            break;
        case M3GEV_DOWNLOADLOGO_FAIL:
            printf("Received download logo fail for the device,
            ulOperIndex=%u\n", ulOperIndex);
            iRetVal = 0;
            break;

        default:
            break;
    }
    return iRetVal;
}
```

- ***ISX_m3g_SetLogoParm()***

Function name:	INT ISX_m3g_SetLogoParm (dev, ucToolboxType, pConfig, mode, ulOperIndex)	
Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	M3G_LOGO_CONFIG *pConfig	Points to LOGO configuration information
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	

Header file	srllib.h m3glib.h
Category:	Video editing
Mode：	Synchronous and asynchronous

Description

The application use the function **ISX_m3g_SetLogoParm()** to set the LOGO attribute such as overlay position, showing, hiding or other parameters of video channel. After the LOGO file downloaded to the toolbox of video channel(please refer to **ISX_m3g_DownloadLogo()** function), the application can change the attributes of the LOGO. Before use the function, the application must start the video media of the channel (namely, the function ISX_m3g_StartMedia() must be successful).

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
pConfig	Points to LOGO configuration information and it is **M3G_LOGO_CONFIG** structure. This pointer can not be NULL. The LOGO is display on default.
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous (default) EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the LOGO parameter setting is completed and the event **M3GEV_SETLOGOPARM_CMPLT** occurs.

Stop Event

M3GEV_SETLOGOPARM_CMPLT: The LOGO attributes set successfully.
M3GEV_SETLOGOPARM_FAIL: The LOGO attributes set unsuccessfully.

Warning

- If the device handle is not the video channel handle, the function fails.
- If the video media does not started with the function ISX_m3g_StartMedia(), the function fails.
- Only when the video channel is added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"
M3GDEV m3gdev;
/*
* Open m3gdev...
*/

UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;
UCHAR ctrlcode   = M3G_CTRLCODE_NO_OP;
M3G_LOGO_CONFIG LogoCfg;
memset(&LogoCfg, 0, sizeof(LogoCfg));
LogoCfg.LogoParm.ucValid = 1;
LogoCfg.LogoParm.ulLeftPos = 10;
LogoCfg.LogoParm.ulTopPos = 20;
LogoCfg.LogoParm.ulWidth = 32;
LogoCfg.LogoParm.ulHeight = 64;
LogoCfg.LogoParm.ucTransY = 11;
LogoCfg.LogoParm.ucTransU = 22;
LogoCfg.LogoParm.ucTransV = 33;
LogoCfg.LogoParm.ucTransAlpha = 44;
LogoCfg.LogoParm.ucAlphaBlending = 55;

if (ISX_m3g_SetLogoParm(m3gdev,tb_type,&LogoCfg, EV_ASYNC)==-1){
    printf( "Cannot SetLogoParm() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m3gdev) );
    return -1;
}
INT iRetVal = 0;
while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(SetLogoParm_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int SetLogoParm_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    //Expected reply to ISX_m3g_SetLogoParm()
    case M3GEV_SETLOGOPARM_CMPLT:
        printf("Received    set    logo    param    success    for    the
        device,ulOperIndex=%u\n", ulOperIndex);
        iRetVal = 0;
        break;
    case M3GEV_SETLOGOPARM_FAIL:
        printf("Received set logo param fail for the device, ulOperIndex=%u\n",
```

```
            ulOperIndex);
            iRetVal = 0;
            break;

        default:
            break;
        }
        return iRetVal;
    }
```

- ***ISX_m3g_StartTextOverlay()***

Function name:	INT ISX_m3g_StartTextOverlay(dev, ucToolboxType, ucStringId, pConfig, mode, ulOperIndex)	
Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	UCHAR ucStringId	Text string ID
	M3G_TEXTOVERLAY_CONFIG *pConfig	Points to text overlay configuration information
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode：	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StartTextOverlay()** to start the text overlay function corresponding to **ucStringId** of the video channel.

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
ucStringId	String ID of Text Overlay; each channel can have multiple text strings overlayed and each text string has a ID as denotation; The range is 0~7;
pConfig	Points to the **M3G_TEXTOVERLAY_CONFIG** structure
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous

ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the text overlay startup is completed and the event **M3GEV_STARTTEXTOVERLAY_CMPLT** occurs.

Stop Event

M3GEV_STARTTEXTOVERLAY_CMPLT: The text overlay started successfully.
M3GEV_STARTTEXTOVERLAY_FAIL: The text overlay started unsuccessfully.

Warning

- If the channel handle is not the video channel handle, the function fails.
- Only when the video channel is added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"
M3GDEV m3gdev;
/*
* Open m3gdev...
*/

INT iRetVal = 0;
UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;
UCHAR stringId   = 0;
M3G_TEXTOVERLAY_CONFIG TextCfg;
M3G_TEXTOVERLAY_CONFIG_USR *pTextCfg =
                (M3G_TEXTOVERLAY_CONFIG_USR *)&TextCfg;
M3G_TO_CARET_POSITION_USR *pCaretPos =
                (M3G_TO_CARET_POSITION_USR *)&pTextCfg->CaretPos;
M3G_TO_DISPLAY_MODE_USR *pDisplayMode =
                (M3G_TO_DISPLAY_MODE_USR *)&pTextCfg->DisplayMode;
M3G_TO_FONT_INFO_USR *pFontInfo =
                (M3G_TO_FONT_INFO_USR *)&pTextCfg->FontInfo;
M3G_TO_SCROLL_INFO_USR *pScrollInfo =
                (M3G_TO_SCROLL_INFO_USR *)&pTextCfg->ScrollInfo;
M3G_TO_COLOR_USR *pColorInfo =
                (M3G_TO_COLOR_USR *)&pTextCfg->ColorInfo;
M3G_TO_TEXT_USR *pText =
                (M3G_TO_TEXT_USR *)&pTextCfg->Text;
pTextCfg->Init();
pCaretPos->SetValue(TEXT_DIRECTION_LEFT_TO_RIGHT, 0, 120);
pDisplayMode->SetValue(M3G_DISPLAY_MODE_SHOW);
pFontInfo->SetValue(0, 16, TEXT_STYLE_BM_BOLD);
```

```
pScrollInfo->SetBackground(0, 120, 176, 20);
pScrollInfo->SetEnable(0);
pScrollInfo->SetScroll(SCROLL_DIRECTION_LEFT_TO_RIGHT, 0, 30, 0, 176);
pColorInfo->SetForeground(0,0,0,44);
pColorInfo->SetBackground(255,255,255,0);
char szText[256] = {"This is text overlay!"};
pText->SetValue(szText);

if (ISX_m3g_StartTextOverlay(m3gdev,tb_type,stringId, &TextCfg,
    EV_ASYNC)==-1){
    printf( "Cannot StartTextOverlay() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m_M3gCh[channel].devh) );
    return -1;
}

while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(TextOverlay_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int TextOverlay_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    //Expected reply to ISX_m3g_StartTextOverlay()
    case M3GEV_STARTTEXTOVERLAY_CMPLT:
        printf("Received    start    textoverlay    success    for    the
        device,ulOperIndex=%u\n", ulOperIndex);
        iRetVal = 0;
        break;
    case M3GEV_STARTTEXTOVERLAY_FAIL:
        printf("Received   start   textoverlay   fail   for   the   device,
        ulOperIndex=%u\n", ulOperIndex);
        break;

    default:
        break;
    }
    return iRetVal;
}
```

- ***ISX_m3g_StopTextOverlay()***

Function name:	INT ISX_m3g_StopTextOverlay(dev, ucToolboxType, ucStringId, mode, ulOperIndex)

Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	UCHAR ucStringId	Text string ID
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StopTextOverlay()** to stop the text overlay function corresponding to **ucStringId** of the video channel.

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
ucStringId	String ID of Text Overlay; each channel can have multiple text strings overlayed and each text string has a ID as denotation; The range is 0~7;
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the text overlay stop is completed and the event **M3GEV_STOPTEXTOVERLAY_CMPLT** occurs.

Stop Event

M3GEV_STOPTEXTOVERLAY_CMPLT: The event text overlay stoped successfully.
M3GEV_STOPTEXTOVERLAY_FAIL: The text overlay stoped unsuccessfully.

Warning

- If the channel handle is not the video channel handle, the function fails.
- Only when the video channel is added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"
M3GDEV m3gdev;
/*
* Open m3gdev...
*/

INT iRetVal = 0;

if (ISX_m3g_StopTextOverlay(m3gdev,tb_type,stringId, EV_ASYNC)==-1){
    printf( "Cannot StopTextOverlay() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m3gdev) );
    return -1;
}

while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(TextOverlay_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int TextOverlay_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    case M3GEV_STOPTEXTOVERLAY_CMPLT:
        printf("Received    stop    textoverlay    success    for    the
        device,ulOperIndex=%u\n", ulOperIndex);
        iRetVal = 0;
        break;
    case M3GEV_STOPTEXTOVERLAY_FAIL:
        printf("Received   stop   textoverlay   fail   for   the   device,
        ulOperIndex=%u\n", ulOperIndex);
        break;

    default:
        break;
    }
```

```
    return iRetVal;
}
```

- ***ISX_m3g_SetTextOverlayParm()***

Function name:	INT ISX_m3g_SetTextOverlayParm(dev, ucToolboxType, ucStringId, pConfig, mode, ulOperIndex)	
Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	UCHAR ucStringId	Text string ID
	M3G_TEXTOVERLAY_CONFIG *pConfig	Points to text overlay configuration information
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode：	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_SetTextOverlay()** to modify the text overlay configuration corresponding to **ucStringId** of the video channel.

Parameters	Description
dev	The valid video channel device handle retrived by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
ucStringId	String ID of Text Overlay; each channel can have multiple text strings overlayed and each text string has a ID as denotation; The range is 0~7;
pConfig	Points to the **M3G_TEXTOVERLAY_CONFIG** structure
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the text overlay parameter setting is completed and the event **M3GEV_SETTEXTOVERLAYPARM_CMPLT** occurs.

Stop Event

M3GEV_SETTEXTOVERLAYPARM_CMPLT: The text overlay attributes set successfully.
M3GEV_SETTEXTOVERLAYPARM_FAIL: The text overlay attributes set unsuccessfully.

Warning

- If the channel handle is not the video channel handle, the function fails.
- Only when the video channel is added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"
M3GDEV m3gdev;
/*
* Open m3gdev...
*/

INT iRetVal = 0;
UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;
UCHAR stringId   = 0;
M3G_TEXTOVERLAY_CONFIG TextCfg;
M3G_TEXTOVERLAY_CONFIG_USR *pTextCfg =
                (M3G_TEXTOVERLAY_CONFIG_USR *)&TextCfg;
M3G_TO_CARET_POSITION_USR *pCaretPos =
                (M3G_TO_CARET_POSITION_USR *)&pTextCfg->CaretPos;
M3G_TO_DISPLAY_MODE_USR *pDisplayMode =
                (M3G_TO_DISPLAY_MODE_USR *)&pTextCfg->DisplayMode;
M3G_TO_FONT_INFO_USR *pFontInfo =
                (M3G_TO_FONT_INFO_USR *)&pTextCfg->FontInfo;
M3G_TO_SCROLL_INFO_USR *pScrollInfo =
                (M3G_TO_SCROLL_INFO_USR *)&pTextCfg->ScrollInfo;
M3G_TO_COLOR_USR *pColorInfo =
                (M3G_TO_COLOR_USR *)&pTextCfg->ColorInfo;
M3G_TO_TEXT_USR *pText =
                (M3G_TO_TEXT_USR *)&pTextCfg->Text;
pTextCfg->Init();
pCaretPos->SetValue(TEXT_DIRECTION_LEFT_TO_RIGHT, 0, 120);
pDisplayMode->SetValue(M3G_DISPLAY_MODE_SHOW);
pFontInfo->SetValue(0, 16, TEXT_STYLE_BM_BOLD);
pScrollInfo->SetBackground(0, 120, 176, 20);
pScrollInfo->SetEnable(0);
pScrollInfo->SetScroll(SCROLL_DIRECTION_LEFT_TO_RIGHT, 0, 30, 0, 176);
pColorInfo->SetForeground(0,0,0,44);
pColorInfo->SetBackground(255,255,255,0);
char szText[256] = {"This is text overlay!"};
```

```
pText->SetValue(szText);

if (ISX_m3g_SetTextOverlayParm(m3gdev,tb_type,stringId, &TextCfg,
    EV_ASYNC)==-1){
    printf( "Cannot SetTextOverlayParm() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m3gdev) );
    return -1;
}

while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(TextOverlay_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int TextOverlay_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    case M3GEV_SETTEXTOVERLAYPARM_CMPLT:
        printf("Received    set    textoverlay    success    for    the
        device,ulOperIndex=%u\n", ulOperIndex);
        iRetVal = 0;
        break;
    case M3GEV_SETTEXTOVERLAYPARM_FAIL:
       printf("Received set textoverlay fail for the device, ulOperIndex=%u\n",
       ulOperIndex);
        iRetVal = 0;
        break;

    default:
        break;
    }
    return iRetVal;
}
```

- ***ISX_m3g_GetTextOverlayParm()***

Function name:	INT ISX_m3g_GetTextOverlayParm(dev, ucToolboxType, ucStringId, pConfig, mode, ulOperIndex)	
Parameters:	**int dev**	Valid video channel device handle
	UCHAR ucToolboxType	Toolbox type
	UCHAR ucStringId	Text string ID

	M3G_TEXTOVERLAY_CONFIG *pConfig	Points to the configuration information structure
	unsigned short mode	Synchronous/asynchronous mode
	ULONG ulOperIndex	asynchronous operation number
Return value	0 Success -1 Failure	
Header file	srllib.h m3glib.h	
Category:	Video editing	
Mode：	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetTextOverlayParm()** to retrive the text overlay configuration corresponding to **ucStringId** of the video channel.

Parameters	Description
dev	The valid video channel device handle returned by **ISX_m3g_OpenEx**()
ucToolboxType	Please refer to **eM3G_TOOLBOX_TYPE** for the toolbox type; only after the video channel was added to the video mixer, the appliction can configure the M3G_TOOLBOX_TYPE_VIDEOCONF.
ucStringId	String ID of Text Overlay; each channel can have multiple text strings overlayed and each text string has a ID as denotation; The range is 0~7;
pConfig	Textoverlay configuration information that retrived; Used only with the synchronous mode and is the **M3G_TEXTOVERLAY_CONFIG** structure. On asynchronous mode, the textoverlay attributes is obtained through **ISX_sr_getevtdatap ()**.
mode	Synchronous/asynchronous mode: EV_SYNC - synchronous EV_ASYNC - asynchronous
ulOperIndex	asynchronous operation number; if the asynchronous operation mode is used, the number can be obtained through ISX_sr_getevtoperindex() function when the text overlay parameter getting is completed and the event **M3GEV_GETTEXTOVERLAYPARM_CMPLT** occurs.

Stop Event

M3GEV_GETTEXTOVERLAYPARM_CMPLT: The text overlay attributes retrived successfully.
M3GEV_GETTEXTOVERLAYPARM_FAIL: The text overlay attributes retrived unsuccessfully.

Warning

- If the channel handle is not the video channel handle, the function fails.
- Only when the video channel is added into the video mixer, the toolbox type M3G_TOOLBOX_TYPE_VIDEOCONF is valid.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
LE_INVALID_DEVHDL: wrong device handle (when the device handle is invalid, please use ISX_sr_getlasterr() to obtain it)
EM3G_BADPARM: parameter error
EM3G_BUSY: device busy

Example

```
#include "IsxApi.h"
#include "srllib.h"
#include "m3glib.h"
M3GDEV m3gdev;
/*
* Open m3gdev...
*/

INT iRetVal = 0;
UCHAR tb_type = M3G_TOOLBOX_TYPE_PRIMARY;
UCHAR stringId   = 0;

if (ISX_m3g_GetTextOverlayParm(m3gdev,tb_type,stringId, NULL,
    EV_ASYNC)==-1){
    printf( "Cannot GetTextOverlayParm() for m3g channel. errmsg = %s",
    ISX_ATDV_ERRMSGP(m3gdev) );
    return -1;
}
while(1) {
    iRetVal=ISX_sr_waitevt(-1);
    if(iRetVal==0) {
        if(TextOverlay_handler()==0) break;
    }
    apr_sleep(10*1000);
}

int TextOverlay_handler()
{
    int iRetVal = -1;
    int nEventType = ISX_sr_getevttype();
    int nDeviceID = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    CHAR szRetConfig[TEMP_STR_LEN]={0};
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    case M3GEV_GETTEXTOVERLAYPARM_CMPLT:
        {
        printf("Received get textoverlay success for the
        device,ulOperIndex=%u\n", ulOperIndex);
        M3G_TEXTOVERLAY_CONFIG_USR *pTextCfg =
                  (M3G_TEXTOVERLAY_CONFIG_USR *)ISX_sr_getevtdatap();
        pTextCfg->TraceStr(szRetConfig);
        printf("GetParam = %s", szRetConfig);
```

```
            iRetVal = 0;
            break;
            }
        case M3GEV_GETTEXTOVERLAYPARM_FAIL:
          printf("Received get textoverlay fail for the device, ulOperIndex=%u\n",
          ulOperIndex);
            iRetVal = 0;
            break;

        default:
            break;
        }
        return iRetVal;
    }
```

2.7. H324M Call Functions

- ***ISX_m3g_StartH245()***

Function name:	int ISX_m3g_StartH245(dev, pParmInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_CHAN_PARAM *pParmInfo	Points to the M3G channel parameter information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StartH245()** to initiate the H.245 call of the M3G channel.

参数 Parameters	描述 Description
dev	The M3G channel device handle returned by **ISX_m3g_OpenEx**( )
pParmInfo	points to the M3G channel parameter information structure; please refer to **M3G_CHAN_PARAM** for the details.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the events M3GEV_START_H245_CMPLT or M3GEV_START_H245_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_START_H245_CMPLT: The H.245 call initiated successfully.
M3GEV_START_H245_FAIL: The H.245 call initiated unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>

int starth245_handler();
main()
{
   ULONG ulOperIndex = 0;
   M3GDEV devh;           /* channel device handle */

   /* Open m3g channel device */
   if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
       printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
       exit(1);
   }
   /*  Set m3g channel device param */
   M3G_CHAN_PARAM ParmInfo;
   ISX_m3g_Set2Default(M3G_CHAN_TYPE_H324M, &ParmInfo);
   ParmInfo.voice.SrcIpAddr.Valid = 1;
   strcpy(ParmInfo.voice.SrcIpAddr.IpAddr,"1.2.3.4");
   ParmInfo.voice.SrcRtpPort.Valid =1 ;
   ParmInfo.voice.SrcRtpPort.UdpPort = 9001;
   ParmInfo.voice.DstIpAddr.Valid = 1;
   strcpy(ParmInfo.voice.DstIpAddr.IpAddr,"5.6.7.8");
   ParmInfo.voice.DstRtpPort.Valid = 1 ;
   ParmInfo.voice.DstRtpPort.UdpPort = 9002;
   ParmInfo.voice.SrcRTCPPort.Valid = 1;
   ParmInfo.voice.SrcRTCPPort.UdpPort = 9003;
   ParmInfo.voice.DstRTCPPort.Valid = 1;
   ParmInfo.voice.DstRTCPPort.UdpPort = 9004;
   ParmInfo.voice.ucConnMode = 0;
   ParmInfo.voice.ucVoiceMode = 0;
   //...
   if( ISX_m3g_StartH245(devh, &ParmInfo, EV_ASYNC, ulOperIndex)==-1 ) {
      printf( "Cannot StartH245() for m3g channel. errmsg = %s\n",
                ISX_ATDV_ERRMSGP(devh) );
      /*
      .
      Perform Error Processing
      .
      */
   }
   ISX_sr_waitevt(-1);
   starth245_handler();

   /*
   .
```

```
   Continue processing
   .
   */
}

int starth245_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    switch(nEventType)
    {
    /*
    .
    . List of expected events
    .
    */
    //Expected reply to ISX_m3g_StartH245()
    case M3GEV_START_H245_CMPLT:
        printf("Received M3GEV_START_H245_CMPLT for the device,
               ulOperIndex=%u\n", ulOperIndex);
        break;
    case M3GEV_START_H245_FAIL:
        printf("Received M3GEV_START_H245_FAIL for the device,
                ulOperIndex=%u\n", ulOperIndex);
        break;
    default:
        break;
    }
}
```

References

- **ISX_m3g_Set2Default( )**
- **ISX_m3g_StopH245( )**

▪ *ISX_m3g_StopH245()*

Function name:	int ISX_m3g_StopH245(dev, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_StopH245()** to drop the H245 call of the M3G channel.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the events M3GEV_STOP_H245_CMPLT or M3GEV_STOP_H245_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_STOP_H245_CMPLT: The H245 call dropped successfully.
M3GEV_STOP_H245_FAIL: The H245 call dropped unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

例 子

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;
   /* Open the channel . Get channel handle in chdev.
    */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
    .
    continue processing
    .
   */

   if ( ISX_m3g_StopH245(chdev, EV_SYNC) == -1) {
      /* process error */
```

```
    }
}
```

References

- **ISX_m3g_StartH245( )**

▪ *ISX_m3g_GetCallStatus()*

Function name:	int ISX_m3g_GetCallStatus(dev, pStatusInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_CALL_STATUS *pStatusInfo	Points to the H324M call status information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetCallStatus()** to retrive the call status (including H223 link status and call progress status) after the 3G-324M call initiated. The retrived status information was written into the buffer specified by **pStatusInfo**.

Synchronous Operation

On default, the function called with synchronous mode. In synchronous mode, the result code is 0 when the function is successfully completed.

Asynchronous Operation

If the application called the function with asynchronous mode, the **mode** is EV_ASYNC. In asynchronous mode, the function completed immediately and the result code is 0 on successful. When the status is retrived, the event **M3GEV_GET_CALL_STATUS_CMPLT** will occur. The event data can be obtained through **ISX_sr_getevtdatap()** and mapped as structure **M3G_CALL_STATUS.** When an error occurs during retriving, theevent **M3GEV_GET_CALL_STATUS_FAIL** will occur.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pStatusInfo	points to the H324M call status information structure; please refer to **M3G_CALL_STATUS** for the details.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed, the number can

	be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_GET_CALL_STATUS_CMPLT: The call status retrived successfully.
M3GEV_GET_CALL_STATUS_FAIL: The call status retrived unsuccessfully.

Warning

- If the specified channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV devh;           /* channel device handle */
   /*
   * Open m3g channel device
   */
   if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
   printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
   exit(1);
   }
   /*
   * Get m3g channel device call status
    */
   M3G_CALL_STATUS StatusInfo;
   if ( ISX_m3g_GetCallStatus(devh, &StatusInfo) == -1 ) {
   printf( "Cannot GetCallStatus() for m3g channel. errmsg = %s",
   ISX_ATDV_ERRMSGP(devh) );
   exit(1);
   }

   /*
    * Continue processing
    */
   //...


   if (ISX_m3g_Close( devh ) == -1 ) {
```

```
        printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

References

- **ISX_m3g_StartH245( )**

▪ *ISX_m3g_GetMsdInfo()*

Function name:	int ISX_m3g_GetMsdInfo(dev, pMsdInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_MSD_INFO *pMsdInfo	points to the MSD information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetMsdInfo()** to retrive the MSD information (such as MSD signal status, local terminal type, remote terminal type and other informations) after the 3G-324M call initiated. The retrived MSD data information is written into the buffer specified by **pMsdInfo**.

Synchronous Operation

On default, the function called with synchronous mode. In synchronous mode, the result code is 0 when the MSD information retrived successful.

Asynchronous Operation

If the application called the function with asynchronous mode, **mode** must be EV_ASYNC. In asynchronous mode, the function completed immediately and the result code is 0 on successful. When the information is retrived, the event **M3GEV_GET_MSDINFO_CMPLT** will occur; the event data can be obtained through **ISX_sr_getevtdatap()** and mapped as data structure **M3G_MSD_INFO;** when an error occurs during retriving, the event **M3GEV_GET_MSDINFO_FAIL** will occur.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pMsdInfo	points to the MSD information structure; please refer to **M3G_MSD_INFO** for details.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous

ulOperIndex	Asynchronous operation number; when the operation is completed, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_GET_MSDINFO_CMPLT: The MSD information retrived successfully.
M3GEV_GET_MSDINFO_FAIL: The MSD information retrived unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- **ISX_m3g_StartMedia()、ISX_m3g_StartH245()**。
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV devh;           /* channel device handle */
   /*
   * Open m3g channel device
   */
   if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
   printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
   exit(1);
   }
   /*
   * Get m3g channel device MSD status
    */
   M3G_MSD_INFO MsdInfo;
   if ( ISX_m3g_GetMsdInfo(devh, &MsdInfo) == -1 ) {
   printf( "Cannot GetMsdInfo() for m3g channel. errmsg = %s",
   ISX_ATDV_ERRMSGP(devh) );
   exit(1);
   }

   /*
    * Continue processing
    */
   //...
```

```
   if (ISX_m3g_Close( devh ) == -1 ) {
       printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
   }
}
```

References

- **ISX_m3g_StartH245( )**

ISX_m3g_GetTermCaps()

Function name:	int ISX_m3g_GetTermCaps(dev, pTcsInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_TCS_INFO *pTcsInfo	points to the M3G terminal capacity set information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetTermCaps()** to retrive the local, remote and matching TCS information (TCS data) after the 3G-324M call initiated. The retrived TCS data information is written into the buffer specified by **pTcsInfo**. The local terminal capacity can be modified through **ISX_m3g_SetParm()** function.

Synchronous Operation

On default, the function called with synchronous mode. In synchronous mode, result code is 0 when the TCS is retrived successfully.

Asynchronous Operation

If the application called the function with asynchronous mode, the **mode** must be EV_ASYNC; In asynchronous mode, the function completed immediately and the result code is 0 on successful. When the TCS is retrived, the event **M3GEV_GET_TERM_CAPS_CMPLT** will occur; the event data can be obtained through **ISX_sr_getevtdatap()** and mapped as data structure **M3G_TCS_INFO;** when an error occurs during the retriving, the event **M3GEV_GET_TERM_CAPS_FAIL** will occur.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pTcsInfo	points to the M3G terminal capacity set information structure; please refer to **M3G_TCS_INFO** for details.

mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the events M3GEV_GET_TERM_CAPS_CMPLT or M3GEV_GET_TERM_CAPS_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_GET_TERM_CAPS_CMPLT: The TCS information retrived successfully.
M3GEV_GET_TERM_CAPS_FAIL: The TCS information retrived unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:

EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_GetTermCaps() function

```
#include <srllib.h>
#include <m3glib.h>
main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
        printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /*
    * Get m3g channel device terminal capilites
     */
    M3G_TCS_INFO TcsInfo;
    if ( ISX_m3g_GetTermCaps(devh, &TcsInfo) == -1 ) {
        printf( "Cannot GetTermCaps() for m3g channel. errmsg = %s",
        ISX_ATDV_ERRMSGP(devh) );
        exit(1);
    }

    /*
     * Continue processing
```

```
     */
    //...


    if (ISX_m3g_Close( devh ) == -1 ) {
        printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
    }
}
```

B> This example shows how to use the asynchronous mode of ISX_m3g_GetTermCaps() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>

int gettcs_handler();
M3G_TCS_INFO TcsInfo;

main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
        printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /*
    * Get m3g channel device terminal capilites
     */

    if ( ISX_m3g_GetTermCaps(devh, &TcsInfo, EV_ASYNC) == -1 ) {
        printf( "Cannot GetTermCaps() for m3g channel. errmsg = %s",
        ISX_ATDV_ERRMSGP(devh) );
        exit(1);
    }
    ISX_sr_waitevt(-1);
    gettcs_handler();

    /*
     .
     . Continue processing
     .
    */

}
int gettcs_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    M3G_TCS_INFO *pTcsInfo = &TcsInfo;
    /*
    or M3G_TCS_INFO *pTcsInfo = (M3G_TCS_INFO *)ISX_sr_getevtdatap();
    */
    switch(nEventType)
    {
     /*
      .
      . List of expected events
      .
```

```
    */
    //Expected reply to ISX_m3g_GetTermCaps()
    case M3GEV_GET_TERM_CAPS_CMPLT:
        printf("Received M3GEV_GET_TERM_CAPS_CMPLT for the
              device,ulOperIndex=%u\n", ulOperIndex);
        /*
        .
        Process pTcsInfo data;
        .
        */
        break;
    case M3GEV_GET_TERM_CAPS_FAIL:
        printf("Received M3GEV_GET_TERM_CAPS_FAIL for the device,
              ulOperIndex=%u\n", ulOperIndex);
        break;
    default:
        break;
    }

   /* Kick off next function in the state machine model. */
   /*
    .
     Continue processing
    .
    */
   return 0;
}
```

References

- **ISX_m3g_StartH245( )**

▪ *ISX_m3g_SendH245MiscCmd()*

Function name	int ISX_m3g_SendH245MiscCmd(dev, pMiscInfo, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	M3G_H245_MISC_INFO *pMiscInfo	points to the MISC command information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call function	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_SendH245MiscCmd()** to send MISC command to the remote of the H.245 call. The command must be sent after the H245 call initiated(namely, after the **ISX_m3g_StartH245()** function is successfully completed).

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
pMiscInfo	points to the MISC command information structure; please refer to **M3G_H245_MISC_INFO** for details.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_SEND_H245_MISC_CMD_CMPLT or M3GEV_SEND_H245_MISC_CMD_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_SEND_H245_MISC_CMD_CMPLT: The MISC command sent successfully.
M3GEV_SEND_H245_MISC_CMD_FAIL: The MISC command sent unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV chdev;
   /* Open the channel . Get channel handle in chdev.
    */
   if ( ISX_m3g_OpenEx(&chdev, 0, 0, 0) == -1) {
      /* process error */
   }
   /*
    .
    continue processing
```

```
    .
   */

   M3G_H245_MISC_INFO MiscInfo;
   /*
   对 MiscInfo 赋值；Assign value to MiscInfo;
   */
   MiscInfo.ucLogicChanNo = M3G_LC_VIDEO_OUTPUT;
   MiscInfo.ulMiscCmdId = M3G_MISC_ID_VIDEO_FAST_UPDATE_GOB;
   MiscInfo.vfug.ulFirstGOB = 1;
   MiscInfo.vfug.ulNumberOfGOBs = 2;
   if (ISX_m3g_SendH245MiscCmd(chdev, &MiscInfo) == -1) {
      /* process error */
   }
}
```

References

- **ISX_m3g_StartH245( )**

ISX_m3g_GetLCParm()

Function name	int ISX_m3g_GetLCParm(dev, pLcParam, mode, ulOperIndex)	
Parameters:	**int dev**	Valid M3G channel device handle
	eM3G_LC_NUMBER LogicChanNo	M3G logical channel number
	M3G_LC_PARAM *pLcParam	points to the M3G logical channel parameter information structure
	unsigned short mode	Mode
	ULONG ulOperIndex	asynchronous operation number
Return value:	0 Success -1 Failure	
Header file:	srllib.h m3glib.h	
Category:	Call functions	
Mode:	Synchronous and asynchronous	

Description

The application use the function **ISX_m3g_GetLCParm()** to retrive the LC parameters(Logic Channel parameters) of H324M channel. The function must be called after receiving the **M3GEV_LCINFO_STATUS** event corresponding to the logical channel. The retrived LC parameter is written into the buffer specified by **pLcParam**.

Synchronous Operation

On default, the function called with synchronous mode. In synchronous mode, result code is 0 when the LC parameter is successfully retrived.

Asynchronous Operation

If the application called the function with asynchronous mode, the **mode** must be EV_ASYNC; in

asynchronous mode, the function completed immediately and the result code is 0 on successful; when the LC parameter is retrived, the event **M3GEV_GET_LC_PARM_CMPLT** will occur; the event data can be obtained through **ISX_sr_getevtdatap()** and mapped as data structure **M3G_LC_PARAM;** when an error occurs during the retriving, the event **M3GEV_GET_LC_PARM_FAIL** will occur.

Parameters	Description
dev	The M3G channel device handle retrived by **ISX_m3g_OpenEx**( )
LogicChanNo	Please refer to **eM3G_LC_NUMBER** for the valid value of H324M logical channel number
pLcParam	points to the M3G logical channel parameter information structure; please refer to **M3G_LC_PARAM** for details.
mode	EV_ASYNC - asynchronous EV_SYNC - synchronous
ulOperIndex	Asynchronous operation number; when the operation is completed and the M3GEV_GET_LC_PARM_CMPLT or M3GEV_GET_LC_PARM_FAIL occurs, the number can be obtained through ISX_sr_getevtoperindex() function.

Stop Event

M3GEV_GET_LC_PARM_CMPLT: The LC parameter retrived successfully.
M3GEV_GET_LC_PARM_FAIL: The LC parameter retrived unsuccessfully.

Warning

- If the channel handle is invalid, the function will fails.
- The channel type must be H324M channel if using the function. The functions that influence the channel type are **ISX_m3g_SetParm(), ISX_m3g_StartMedia() and ISX_m3g_StartH245()**.

Error

The result code -1 for the function fail. Please use the SRL function ISX_ATDV_LASTERR() to retrive the error code, or use the function ISX_ATDV_ERRMSGP() to retrive the error description.

The function may return the following error codes:
EM3G_BADPARM: parameter error
EM3G_TIMEOUT: function execution timeout
EM3G_BUSY: device busy
EM3G_OPERTIMEOUT: operation timeout, no receipt of device response within the specified time
EM3G_SYSTEM: system error (for example, the network connection with MasterController is disconnected)

Example

A> This example shows how to use the synchronous mode of ISX_m3g_GetLCParm() function

```
#include <srllib.h>
#include <m3glib.h>
main()
{
   M3GDEV devh;           /* channel device handle */
   /*
   * Open m3g channel device
   */
   if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
```

```
      printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
      exit(1);
   }
   /*
   * Get m3g LC channel device param
    */
   M3G_LC_PARAM LCParm;
   if ( ISX_m3g_GetLCParm(devh, M3G_LC_VIDEO_OUTPUT, &LCParm) == -1 ) {
      printf( "Cannot GetLCParm() for m3g channel. errmsg = %s",
      ISX_ATDV_ERRMSGP(devh) );
      exit(1);
   }

   /*
    * Continue processing
    */
   //...


   if (ISX_m3g_Close( devh ) == -1 ) {
       printf( "Cannot close M3G channel. errno = %d", ISX_sr_getlasterr() );
   }
}
```

B> This example shows how to use the asynchronous mode of ISX_m3g_GetLCParm() function

```
#include <stdio.h>
#include <srllib.h>
#include <m3glib.h>

int getlcparm_handler();
M3G_LC_PARAM LCParm;

main()
{
    M3GDEV devh;           /* channel device handle */
    /*
    * Open m3g channel device
    */
    if ( ISX_m3g_OpenEx(&devh, 0, 0, 0) == -1 ) {
        printf( "Cannot open m3g channel. errno = %d", ISX_sr_getlasterr() );
        exit(1);
    }
    /*
    * Get m3g LC channel device param
     */

    if (ISX_m3g_GetLCParm(devh,M3G_LC_VIDEO_OUTPUT,&LCParm,EV_ASYNC)==-1){
        printf( "Cannot GetLCParm() for m3g channel. errmsg = %s",
        ISX_ATDV_ERRMSGP(devh) );
        exit(1);
    }
    ISX_sr_waitevt(-1);
    getlcparm_handler();

    /*
     .
     . Continue processing
     .
    */

}
```

```
int getlcparm_handler()
{
    int nEventType = ISX_sr_getevttype();
    int chdev = ISX_sr_getevtdev();
    ULONG ulOperIndex = ISX_sr_getevtoperindex();
    M3G_LC_PARAM *pLCParm = &LCParm;
    /*
    or M3G_LC_PARAM *pLCParm = (M3G_LC_PARAM *)ISX_sr_getevtdatap();
    */
    switch(nEventType)
    {
     /*
      .
      . List of expected events
      .
     */
     //Expected reply to ISX_m3g_GetLCParm()
     case M3GEV_GET_LC_PARM_CMPLT:
         printf("Received M3GEV_GET_LC_PARM_CMPLT for the
              device,ulOperIndex=%u\n", ulOperIndex);
         /*
         .
         Process pLCParm data;
         .
         */
         break;
     case M3GEV_GET_LC_PARM_FAIL:
         printf("Received M3GEV_GET_LC_PARM_FAIL for the device,
             ulOperIndex=%u\n", ulOperIndex);
         break;
     default:
         break;
     }

    /* Kick off next function in the state machine model. */
    /*
     .
     Continue processing
     .
    */
    return 0;
}
```

References

- **ISX_m3g_StartH245( )**

2.8. Enumerated Constants

▪ *eM3G_CHAN_TYPE*

```
typedef enum {
    M3G_CHAN_TYPE_VOICE     = 4, /* The channel is a voice channel          */
    M3G_CHAN_TYPE_VIDEO     = 9, /* The channel is an video channel         */
    M3G_CHAN_TYPE_H324M     = 18,/* The channel is an 3G-324M channel       */
}eM3G_CHAN_TYPE;
```

Description

eM3G_CHAN_TYPE constants defined the M3G channel types. The definitions are in the m3glib.h header file and now the channel types are VOICE, VIDEO and H324M.

The H324M channel type also contains 4 logical channels (refer to **eM3G_LC_NUMBER**).

▪ *eM3G_LC_NUMBER*

```
typedef enum {
    M3G_LC_VIDEO_OUTPUT     = 0, /* video output logical channel */
    M3G_LC_VOICE_OUTPUT     = 1, /* voice output logical channel */
    M3G_LC_VIDEO_INPUT      = 2, /* video input logical channel*/
    M3G_LC_VOICE_INPUT      = 3, /* voice input logical channel*/
}eM3G_LC_NUMBER;
```

Description

eM3G_LC_NUMBER constants defined the H324M logical channel number. The definitions are in the m3glib.h header file.

▪ *eM3G_VOICE_MODE*

```
typedef enum {
    M3G_VOICE_MODE_VOIP             = 0,  /* Voice channel mode: TDM<->IP conversion */
    M3G_VOICE_MODE_P2P              = 1,  /* Voice channel mode: voice transcoding*/
    M3G_VOICE_MODE_FE_IP            = 2,  /* Front End IP - for conference */
    M3G_VOICE_MODE_FE_TDM           = 3,  /* Front End TDM - for conference */
    M3G_VOICE_MODE_RTC              = 4,  /* Run To Completion */
    M3G_VOICE_MODE_PTH223           = 5,  /* Packet to H.223 channel */
    M3G_VOICE_MODE_RX_ONLY          = 6,  /* The TX path is disabled */
    M3G_VOICE_MODE_EVD_EVG          = 7,  /* Only Event Detector and Generator are enabled */
    M3G_VOICE_MODE_FE_IP_NO_VAD     = 8,  /* IP Front End no VAD-for3\4way conference */
    M3G_VOICE_MODE_FE_TDM_NO_VAD    = 9,/* TDM Front End no VAD-for 3\4 way conference */
    M3G_VOICE_MODE_MOIP_RFC2833_ONLY = 10,/* only RFC2833 packet (not voice) */
}eM3G_VOICE_MODE;
```

Description

eM3G_VOICE_MODE constants defined the mode of voice channel. The definitions are in the m3glib.h header file.

▪ *eM3G_VOICE_CODER*

```
typedef enum {
    M3G_VOCODER_G711A     = 0,          /* G.711 A-law protocol  */
    M3G_VOCODER_G711MU = 1,             /* G.711 Mu-law protocol */
    M3G_VOCODER_G723_1H_30MS = 3,       /* G.723.1 High rate protocol (6.3Kbit/s) */
    M3G_VOCODER_G723_1L_30MS = 4,       /* G.723.1 Low rate protocol (5.3Kbit/s) */
    M3G_VOCODER_G726_40 = 6,            /* G.726 protocol at 40Kbit/s */
    M3G_VOCODER_G726_32 = 7,            /* G.726 protocol at 32Kbit/s */
    M3G_VOCODER_G726_24 = 8,            /* G.726 protocol at 24Kbit/s */
    M3G_VOCODER_G726_16 = 9,            /* G.726 protocol at 16Kbit/s */
```

```
    M3G_VOCODER_G729_AB = 16,              /* G.729.Annex A + Annex B protocol */
    M3G_VOCODER_GSM_FR = 17,               /* GSM Full Rate */
    M3G_VOCODER_GSM_EFR = 18,              /* GSM Enhanced Full Rate */
    M3G_VOCODER_GSM_AMR475_BE = 19,        /* GSM AMR 4.75Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR515_BE = 20,        /* GSM AMR 5.15Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR59_BE = 21,         /* GSM AMR 5.9Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR67_BE = 22,         /* GSM AMR 6.7Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR74_BE = 23,         /* GSM AMR 7.4Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR795_BE = 24,        /* GSM AMR 7.95Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR102_BE = 25,        /* GSM AMR 10.2Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR122_BE = 26,        /* GSM AMR 12.2Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_AMR475_OA = 27,        /* GSM AMR 4.75Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR515_OA = 28,        /* GSM AMR 5.15Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR59_OA = 29,         /* GSM AMR 5.9Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR67_OA = 30,         /* GSM AMR 6.7Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR74_OA = 31,         /* GSM AMR 7.4Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR795_OA = 32,        /* GSM AMR 7.95Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR102_OA = 33,        /* GSM AMR 10.2Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_AMR122_OA = 34,        /* GSM AMR 12.2Kbit/s Octet Aligned packing */
    M3G_VOCODER_LINEAR = 35,               /* Linear (16bit) samples */
    M3G_VOCODER_CLEAR = 36,                /* Clear Mu/A-law channel */
    M3G_VOCODER_FULL_EVRC_HFF= 37,         /* EVRC Full Rate (9.6Kbit/s) Header Free Format
                                              (A.K.A evrc0) */
    M3G_VOCODER_HALF_EVRC_HFF= 38,         /* EVRC Half Rate (4.8Kbit/s) Header Free Format
                                              (A.K.A evrc0) */
    M3G_VOCODER_FULL_EVRC_BF = 39,         /* EVRC Full Rate (9.6Kbit/s) Bundled Format */
    M3G_VOCODER_HALF_EVRC_BF = 40,         /* EVRC Half Rate (4.8Kbit/s) Bundled Format */
    M3G_VOCODER_GSM_WB_AMR66_BE = 41, /* WB-AMR 6.6Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR885_BE = 42,/* WB-AMR 8.85Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR1265_BE = 43,/*WB-AMR 12.65Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR1425_BE = 44,/*WB-AMR 14.25Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR1585_BE = 45,/*WB-AMR 15.85Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR1825_BE = 46,/*WB-AMR 18.25Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR1985_BE = 47,/*WB-AMR 19.85Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR2305_BE = 48,/*WB-AMR 23.05Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR2385_BE = 49,/*WB-AMR 23.85Kbit/s Bandwidth Efficient packing */
    M3G_VOCODER_GSM_WB_AMR66_OA = 50,    /* WB-AMR 6.6Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR885_OA = 51, /* WB-AMR 8.85Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR1265_OA = 52,/* WB-AMR 12.65Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR1425_OA = 53,/* WB-AMR 14.25Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR1585_OA = 54,/* WB-AMR 15.85Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR1825_OA = 55,/* WB-AMR 18.25Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR1985_OA = 56,/* WB-AMR 19.85Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR2305_OA = 57,/* WB-AMR 23.05Kbit/s Octet Aligned packing */
    M3G_VOCODER_GSM_WB_AMR2385_OA = 58,/* WB-AMR 23.85Kbit/s Octet Aligned packing */
    M3G_VOCODER_ILBC_20MS = 59,            /* iLBC 20ms frames (15.2Kbit/s) */
    M3G_VOCODER_ILBC_30MS = 60,            /* iLBC 30ms frames (13.3Kbit/s) */
}eM3G_VOICE_CODER;
```

Description

eM3G_VOICE_CODER constants defined the voice coder type. The definitions are in the m3glib.h header file. Except that the packet duration of M3G_VOCODER_ILBC_30MS is 30ms, the others are 20ms.

▪ *eM3G_VIDEO_RTP_PAYLOAD*

```
typedef enum {
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC2190 = 0,    /* Video data is H.263 and RFC 2190 */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_NONE = 1, /* No additional encapuslation RFC is supported */
```

```
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC2429 = 2, /* The compressed video data is H.263 (used for
                                                MPEG4). RFC 2429 and RFC4629 are both supported. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC3984 = 3, /* The compressed video data is H.264 and RFC
                                                3984 is supported. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC2190_MODE_B_ONLY = 20,    /* The compressed video
                                                data is H.263 and RFC 2190 mode B will be used. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC2190_MODE_A_ONLY = 21,    /* The compressed video
                                                data is H.263 and RFC 2190 mode A will be used. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC3984_MODE_1 = 30,   /* The compressed video data is
                                                H.264 and RFC 3984 mode 1 will be used. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC3984_STAP_A_ONLY = 31, /* The compressed video data
                                                is H.264 and RFC 3984 STAP A will be used. */
    M3G_VIDEO_RTP_PAYLOAD_ENCAP_RFC3984_FU_A_ONLY = 32,   /* The compressed video data is
                                                H.264 and RFC 3984 FU A will be used. */
}eM3G_VIDEO_RTP_PAYLOAD;
```

Description

eM3G_VIDEO_RTP_PAYLOAD constants the encapsulation type of RTP compressed video data. The definitions are in the m3glib.h header file.

▪ *eM3G_VIDEO_CODER*

```
typedef enum {
    M3G_VIDEO_CODER_MPEG4   = 0,  /* Mpeg4 video encoding standard */
    M3G_VIDEO_CODER_H263    = 1,  /* H.263 video encoding standard */
    M3G_VIDEO_CODER_H264    = 2,  /* H.264 video encoding standard */
    M3G_VIDEO_CODER_WMV9    = 3,  /* WMV video encoding standard */
} eM3G_VIDEO_CODER;
```

Description

eM3G_VIDEO_CODER constants defined the video coding standard. The definitions are in the m3glib.h header file.

▪ *eM3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL*

```
typedef enum {
    M3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL0 = 0,
    M3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL1 = 1,
    M3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL2 = 2,
    M3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL3 = 3,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL1B = 9,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL10 = 10,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL11 =11,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL12 =12,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL13 =13,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL20 =20,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL21 =21,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL22 =22,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL30 =30,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL31 =31,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL32 =32,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL40 =40,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL41 =41,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL42 =42,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL50 =50,
    M3G_VIDEO_H264_BASELINE_PROFILE_LEVEL51 =51,
} eM3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL;
```

Description

eM3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL constants defined the video compression Profile

and Level. The definitions are in the m3glib.h header file.

▪ *eM3G_H324_AUDIO_CODEC*

```
typedef enum {
    M3G_H324_AUDIO_CODEC_NONE = 0,   /* Defines no audio codec */
    M3G_H324_AUDIO_CODEC_AMR = 1,    /* Defines AMR-NB audio codec */
    M3G_H324_AUDIO_CODEC_G723 = 2,   /* Defines G.723.1 audio codec */
} eM3G_H324_AUDIO_CODEC;
```

Description

eM3G_H324_AUDIO_CODEC constants defined the supported audio coding type of H324M, and One terminal can simultaneously support 4 different audio codecs. The definitions are in the m3glib.h header file.

▪ *eM3G_H324_VIDEO_CODEC*

```
typedef enum {
    M3G_H324_VIDEO_CODEC_NONE = 0,   /* Defines no video codec */
    M3G_H324_VIDEO_CODEC_MPEG4 = 1, /* Defines MPEG-4 video codec */
    M3G_H324_VIDEO_CODEC_H263 = 2,   /* Defines H.263 video codec */
}eM3G_H324_VIDEO_CODEC;
```

Description

eM3G_H324_AUDIO_CODEC constants defined the supported video coding type of H324M, and one terminal can simultaneously support 4 different video codecs. The definitions are in the m3glib.h header file.

▪ *eM3G_FILE_TYPE*

```
typedef enum {
    M3G_FILE_TYPE_BIN = 1,      /* Binary file format */
    M3G_FILE_TYPE_WAV = 2,      /* WAV file format */
    M3G_FILE_TYPE_3GP = 3,      /* 3GP file format */
    M3G_FILE_TYPE_MP4 = 4,      /* MPEG4 file format */
    M3G_FILE_TYPE_AVI = 5,      /* AVI file */
    M3G_FILE_TYPE_ASF = 6,      /* ASF file format */
    M3G_FILE_TYPE_3G2 = 7,      /* 3G2 file format */
} eM3G_FILE_TYPE;
```

Description

eM3G_FILE_TYPE constants defined the supported file format of audio/video files. The definitions are in the m3glib.h header file.

▪ *eM3G_WAV_BLOCK_SIZE*

```
typedef enum {
    M3G_WAV_BLOCK_SIZE_5MS      =0,     /* 5ms */
    M3G_WAV_BLOCK_SIZE_10MS     =1,     /* 10ms */
    M3G_WAV_BLOCK_SIZE_20MS     =3,     /* 20ms */
    M3G_WAV_BLOCK_SIZE_30MS     =5,     /* 30ms */
}eM3G_WAV_BLOCK_SIZE;
```

Description

eM3G_WAV_BLOCK_SIZE constants defined the duration of the data block of the WAV file type (**eM3G_FILE_TYPE**). The definitions are in the m3glib.h header file.

▪ *eM3G_SAMPLE_TYPE*

```
typedef enum {
    M3G_SAMPLE_TYPE_AAC = 1,        /* AAC Audio codec - currently is not supported */
    M3G_SAMPLE_TYPE_AAC_PLUS = 2,   /* AAC Plus Audio codec - currently is not supported */
```

```
    M3G_SAMPLE_TYPE_AMR_NB = 3,          /* Narrow-band AMR voice codec */
    M3G_SAMPLE_TYPE_AMR_WB = 4,          /* Wide-band AMR voice codec */
    M3G_SAMPLE_TYPE_G711_A_LAW = 5,      /* G.711 a-law voice codec */
    M3G_SAMPLE_TYPE_G711_MU_LAW = 6,     /* G.711 mu-law voice codec */
    M3G_SAMPLE_TYPE_G7231 = 7,           /* G.723 voice codec */
    M3G_SAMPLE_TYPE_G726_16 = 8,         /* G.726-16 voice codec */
    M3G_SAMPLE_TYPE_G728 = 9,            /* G.728 voice codec - currently is not supported */
    M3G_SAMPLE_TYPE_G729A = 10,          /* G.729A voice codec */
    M3G_SAMPLE_TYPE_GSM_FR = 11,         /* GSM-FR voice codec */
    M3G_SAMPLE_TYPE_GSM_EFR = 12,        /* GSM-EFR voice codec */
    M3G_SAMPLE_TYPE_GSM_HR = 13,         /* GSM-HR voice codec - currently is not supported */
    M3G_SAMPLE_TYPE_MPEG1_L2 = 14,       /* MPEG1-L2 audio codec - currently is not supported */
    M3G_SAMPLE_TYPE_PCM = 15,            /* PCM - linear voice data */
    M3G_SAMPLE_TYPE_H263 = 16,           /* H.263 Video codec */
    M3G_SAMPLE_TYPE_H264 = 17,           /* H.264 Video codec */
    M3G_SAMPLE_TYPE_MPEG4 = 18,          /* MPEG-4 Video codec */
    M3G_SAMPLE_TYPE_YUV = 19,            /* MPEG-4 Video codec */
    M3G_SAMPLE_TYPE_WMA = 20,            /* WMA audio codec */
    M3G_SAMPLE_TYPE_WMV9 = 21,           /* WMV9 video codec */
    M3G_SAMPLE_TYPE_G726_24 = 22,        /* G.726-24 voice codec */
    M3G_SAMPLE_TYPE_G726_32 = 23,        /* G.726-32 voice codec */
    M3G_SAMPLE_TYPE_G726_40 = 24,        /* G.726-40 voice codec */
    M3G_SAMPLE_TYPE_WMA7 = 25,           /* WMA v.7 audio codec */
    M3G_SAMPLE_TYPE_WMA9 = 26,           /* WMA v.9 audio codec */
    M3G_SAMPLE_TYPE_WMAL = 27,           /* WMA Losseless audio codec */
    M3G_SAMPLE_TYPE_G729D = 28,          /* G.729D voice codec */
    M3G_SAMPLE_TYPE_G729E = 29,          /* G.729E voice codec */
} eM3G_SAMPLE_TYPE;
```

Description

eM3G_SAMPLE_TYPE constants defined the media sample type of the non-WAV file type (**eM3G_FILE_TYPE**) , such as BIN/3GP/MP4/AVI/ASF/3G2 type. The definitions are in the m3glib.h header file.

▪ *eM3G_FILE_TIMING_SOURCE_SAMPLE*

```
typedef enum {
    M3G_FILE_TIMING_SOURCE_SAMPLE_TIMESTAMPS = 0,     /* timestamp from RTP */
    M3G_FILE_TIMING_SOURCE_SAMPLE_HEADERS = 1,        /* timestamp from video head*/
} eM3G_FILE_TIMING_SOURCE_SAMPLE;
```

Description

eM3G_FILE_TIMING_SOURCE_SAMPLE constants defined the source of the timestamp of the non-WAV file type (**eM3G_FILE_TYPE**) ,such as BIN/3GP/MP4/AVI/ASF/3G2 type. The definitions are in the m3glib.h header file.

▪ *eM3G_TCS_STATUS*

```
typedef enum {
    M3G_TCS_STATUS_NONE,                 /* TCS initial status */
    M3G_TCS_STATUS_UNKNOWN,              /* Receive TCS request */
    M3G_TCS_STATUS_STARTED,              /* Send TCS */
    M3G_TCS_STATUS_RESOLVED,             /* Send TCS, and receive TCS of the other side */
    M3G_TCS_STATUS_RESOLVED_FINISHED,    /* TCS is completed and successful */
    M3G_TCS_STATUS_ERROR,                /* TCS is rejected or timeout */
    M3G_TCS_STATUS_IDLE                  /* Idle status*/
}eM3G_TCS_STATUS;
```

Description

eM3G_TCS_STATUS constants defined the status of TCS (Terminal Capability set) negotiation. The

definitions are in the m3glib.h header file.

▪ *eM3G_MISC_ID*

```
typedef enum {
    M3G_MISC_ID_VIDEO_FAST_UPDATE_PICTURE,
    M3G_MISC_ID_VIDEO_FAST_UPDATE_GOB,
    M3G_MISC_ID_VIDEO_FAST_UPDATE_MB,
    M3G_MISC_ID_VIDEO_SEND_SYNC_EVERY_GOB,
    M3G_MISC_ID_VIDEO_SEND_SYNC_EVERY_GOB_CANCEL,
    M3G_MISC_ID_VIDEO_FREEZE_PICTURE,
    M3G_MISC_ID_VIDEO_TEMPORAL_SPATIAL_TRADEOFF,
    M3G_MISC_ID_SKEW_INDICATION,
} eM3G_MISC_ID;
```

Description

eM3G_MISC_ID constants defined the command ID of Misc command. The definitions are in the m3glib.h header file.

▪ *eM3G_LC_STATUS*

```
typedef enum {
    M3G_LC_STATUS_CREATED,              /* LC intiated status*/
    M3G_LC_STATUS_UNKNOWN,              /* Ready to send OLC request */
    M3G_LC_STATUS_STARTED,              /* Have sent OLC request */
    M3G_LC_STATUS_RESOLVED,             /* Receive OLC request of the other side*/
    M3G_LC_STATUS_RESOLVED_FINISHED,    /* OLC is completed and successful*/
    M3G_LC_STATUS_ERROR,                /* OLC is rejected or timeout */
    M3G_LC_STATUS_IDLE                  /* Idle status */
}eM3G_LC_STATUS;
```

Description

eM3G_LC_STATUS constants defined the status of LC (logical channel) negotiation of H324M. The definitions are in the m3glib.h header file.

▪ *eM3G_CP_STATUS*

```
typedef enum {
    M3G_CP_STATUS_IDLE,             /* Call is in idle status*/
    M3G_CP_STATUS_INITIALIZING,     /* Call is started*/
    M3G_CP_STATUS_ACTIVING,         /* Link is established */
    M3G_CP_STATUS_ACTIVED,          /* MSD and TCS are completed*/
    M3G_CP_STATUS_DEACTIVING,       /* Call is stopped*/
    M3G_CP_STATUS_CLEAN_UP,         /* Call is cleared*/
}eM3G_CP_STATUS;
```

Description

eM3G_CP_STATUS constants defined the call progress status of H324M calls. The definitions are in the m3glib.h header file.

▪ *eM3G_H223_LINK_STATUS*

```
typedef enum {
    M3G_H223_LINK_STATUS_NONE,          /* Link initial status */
    M3G_H223_LINK_STATUS_SYNCHROZING,   /* Link is in synchronization */
    M3G_H223_LINK_STATUS_CONNECTED,     /* Link synchronization is successful */
    M3G_H223_LINK_STATUS_ERROR,         /* Link synchronization is unsuccessful*/
    M3G_H223_LINK_STATUS_IDLE           /* Link idle status*/
}eM3G_H223_LINK_STATUS;
```

Description

eM3G_H223_LINK_STATUS constants defined the H223 link status of H324M calls. The definitions are in the m3glib.h header file.

▪ *eM3G_MSD_STATUS*

```
typedef enum {
    M3G_MSD_STATUS_NONE,            /* MSD initial status */
    M3G_MSD_STATUS_UNKNOWN,         /* Receive MSD request*/
    M3G_MSD_STATUS_STARTED,         /* Send MSD*/
    M3G_MSD_STATUS_RESOLVED,        /* MSD    Send MSD, and receive MSD of the other side */
    M3G_MSD_STATUS_RESOLVED_FINISHED, /* MSD is completed and successful*/
    M3G_MSD_STATUS_ERROR,           /* MSD is rejected or timeout*/
    M3G_MSD_STATUS_IDLE             /* Idle status*/
}eM3G_MSD_STATUS;
```

Description

eM3G_MSD_STATUS constants defined the MSD status of MSD negotiation. The definitions are in the m3glib.h header file.

▪ *eM3G_BRD_CONN_STATUS*

```
typedef enum {
    M3G_BRD_CONN_STATUS_DEACTIVE,           /* Deactive status*/
    M3G_BRD_CONN_STATUS_ACTIVE_DISCONNED,   /* Active status+Disconned status*/
    M3G_BRD_CONN_STATUS_ACTIVE_CONNECTED,   /* Active status+Connected status */
}eM3G_BRD_CONN_STATUS;
```

Description

eM3G_BRD_CONN_STATUS constants defined the connection status between the M3G board from M3GC server. The definitions are in the m3glib.h header file.

▪ *eM3G_DATA_DIR*

```
typedef enum
{
    M3G_DATA_DIR_AFTER_ENCODEC      = 0x00,             // After encoder
    M3G_DATA_DIR_BEFORE_DECODEC     = 0x01,             // Before decoder
    M3G_DATA_DIR_TOIP=M3G_DATA_DIR_AFTER_ENCODEC,   //data to IP side, same as after encoder
    M3G_DATA_DIR_FROMIP=M3G_DATA_DIR_BEFORE_DECODEC,//data from IP side, same as before decoder
    M3G_DATA_DIR_TDM    =   M3G_DATA_DIR_TOIP,      // Only for H324M channel
    M3G_DATA_DIR_IP     =   M3G_DATA_DIR_FROMIP     // Compatible with the old version
}eM3G_DATA_DIR;
```

Description

eM3G_DATA_DIR constants defined the data direction of playing/recording audio and video. The definitions are in the m3glib.h header file.

Note: The data direction of the H324M channel always points to TDM side (please use the M3G_DATA_DIR_TDM).

▪ *eM3G_VIDEO_TRANSPORT_TYPE*

```
typedef enum
{
    /* Streaming files from host to dsp and vice versa through a reliable (proprietary) socket connection */
    M3G_VIDEO_TRANSPORT_TYPE_STREAMING  = 0,
    /* Ip packets through UDP socket connection */
    M3G_VIDEO_TRANSPORT_TYPE_IP         = 1,
    /* UDP packets from the H.223 channel */
```

```
    M3G_VIDEO_TRANSPORT_TYPE_H223          = 2
}eM3G_VIDEO_TRANSPORT_TYPE;
```

Description

eM3G_VIDEO_TRANSPORT_TYPE constants defined the video transport type. The definitions are in the m3glib.h header file.

▪ *eM3G_MEDIA_CHAN_OPMODE*

```
typedef enum {
    M3G_MEDIA_CHAN_OPMODE_NONE              = 0,    //no operation
    M3G_MEDIA_CHAN_OPMODE_PASS_THROUGH      = 1,    // pass through mode
    M3G_MEDIA_CHAN_OPMODE_TRANSCODE         =2,   // transcode mode
}eM3G_MEDIA_CHAN_OPMODE;
```

Description

eM3G_MEDIA_CHAN_OPMODE constants defined the operation type of media channel. The definitions are in the m3glib.h header file.

▪ *eM3G_TOOLBOX_TYPE*

```
typedef enum {
    M3G_TOOLBOX_TYPE_PRIMARY   = 0,       /* primary toolbox*/
    M3G_TOOLBOX_TYPE_VIDEOCONF     = 1   /* video mixer toolbox*/
} eM3G_TOOLBOX_TYPE;
```

Description

eM3G_TOOLBOX_TYPE constants defined the toolbox type. The definitions are in the m3glib.h header file.

▪ *eM3G_TOOLBOX_CTRLCODE*

```
typedef enum {
    M3G_CTRLCODE_NO_OP                  = 0x00,   /* No operation*/
    M3G_CTRLCODE_LOGO_DOWNLOAD          = 0x01,   /* LOGO download*/
    M3G_CTRLCODE_LOGO_DISPLAY           = 0x02,   /* LOGO display */
    M3G_CTRLCODE_LOGO_HIDE              = 0x20,   /* LOGO hide*/
    M3G_CTRLCODE_COLOR_REMOVE           = 0x04,   /* color remove */
    M3G_CTRLCODE_COLOR_DISPLAY          = 0x40,   /* Display normal color*/
    M3G_CTRLCODE_TEXT_OVERLAY           = 0x08,   /* Start text overlay function */
    M3G_CTRLCODE_TEXT_OVERLAY_STOP      = 0x80,   /* Stop TEXT OVERLAY function*/
} eM3G_TOOLBOX_CTRLCODE;
```

Description

eM3G_TOOLBOX_CTRLCODE constants defined the toolbox control code. The definitions are in the m3glib.h header file.

▪ *eM3G_TEXT_DIRECTION*

```
typedef enum {
    TEXT_DIRECTION_LEFT_TO_RIGHT    = 0x00,
    TEXT_DIRECTION_RIGHT_TO_LEFT    = 0x01,
    TEXT_DIRECTION_UP_TO_DOWN       = 0x02,
    TEXT_DIRECTION_DOWN_TO_UP       = 0x03
}eM3G_TEXT_DIRECTION;
```

Description

eM3G_TEXT_DIRECTION constants defined the text direction of the textoverlay. The definitions are in the m3glib.h header file.

▪ *eM3G_TEXT_STYLE*

```
typedef enum {
    TEXT_STYLE_BM_BOLD              = 0x00000001,
    TEXT_STYLE_BM_ITALIC            = 0x00000002,
    TEXT_STYLE_BM_UNDERLINE         = 0x00000004,
    TEXT_STYLE_BM_OUTLINE           = 0x00000008,
    TEXT_STYLE_BM_STRIKETHROUGH     = 0x00000010,
    TEXT_STYLE_BM_SUBSCRIPT         = 0x00000020,
    TEXT_STYLE_BM_SUPERSCRIPT       = 0x00000040,
    TEXT_STYLE_BM_SHADOW            = 0x00000080,
    TEXT_STYLE_BM_ENGRAVE           = 0x00000100
}eM3G_TEXT_STYLE;
```

Description

eM3G_TEXT_STYLE constants defined the font style and it is bitwise. The definitions are in the m3glib.h header file.

▪ *eM3G_SCROLL_DIRECTION*

```
typedef enum {
    SCROLL_DIRECTION_LEFT_TO_RIGHT = 0x00,
    SCROLL_DIRECTION_RIGHT_TO_LEFT = 0x01,
}eM3G_SCROLL_DIRECTION;
```

Description

eM3G_SCROLL_DIRECTION constants defined the text scroll direction. The definitions are in the m3glib.h header file.

▪ *eM3G_DISPLAY_MODE*

```
typedef enum {
    M3G_DISPLAY_MODE_HIDE   = 0,
    M3G_DISPLAY_MODE_SHOW   = 1
} eM3G_DISPLAY_MODE;
```

Description

eM3G_DISPLAY_MODE constants defined the display mode, such as showing or hiding. The definitions are in the m3glib.h header file.

2.9. Data Structures

▪ *M3G_VOICE_CODER*

```
typedef struct tagM3G_VOICE_CODER{
    UCHAR  ucValid;          //the structure valid flag
    UCHAR  ucVoCoder;        // voice encoder type
    UCHAR  ucPayloadType;    // Payload type
    UCHAR  ucCNG;            // CNG of the decoder;
    UCHAR  ucPLC;            // PLC of the decoder;
} M3G_VOICE_CODER;
```

Description

M3G_VOICE_CODER data structure is used to configure the coder/decoder of voice channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VOICE_CODER** is described as follows:

ucValid： The structure valid flag:

0: invalid or 1: valid.

ucVoCoder： Please refer to **eM3G_VOICE_CODER** for the valid value of voice encoder type

ucPayloadType： Payload type

ucCNG： CNG of the decoder;

ucPLC： PLC of the decoder.

▪ *M3G_VOICE_ATTR_CONFIG*

```
typedef struct tagM3G_VOICE_ATTR_CONFIG{
    M3G_VOICE_CODER          VoEnCoder;       // Encoder configuration
    M3G_VOICE_CODER          VoDeCoder;       // Decoder configuration
    VoiceConfig_PayloadSize  PayloadSize;     // Load size
    VoiceConfig_SilenceSuppress SilenceSuppress; // Silence suppression
    VoiceConfig_EchoCancel   EchoCancel;      // Echo cancellation
    VoiceConfig_JitterBufferDelay MinJitterBuffer; // Minimum Jitter buffer delay
    VoiceConfig_JitterBufferDelay MaxJitterBuffer; // Maximum Jitter buffer delay
    VoiceConfig_AdaptationRate AdaptationRate; // Jitter buffer adaptation rate
} M3G_VOICE_ATTR_CONFIG;
```

Description

M3G_VOICE_ATTR_CONFIG data structure defined the voice attribute of M3G voice channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VOICE_ATTR_CONFIG** is described as follows:

VoEnCoder： Please refer to **M3G_VOICE_CODER** for the valid value of encoder configuration.

VoDeCoder： Please refer to **M3G_VOICE_CODER** for the valid value of decoder configuration.

PayloadSize： Please refer to *ISX4000 IPM User Manual* for the valid value of load size.

SilenceSuppress： Please refer to *ISX4000 IPM User Manual* for the valid value of silence suppression mark.

EchoCancel： Please refer to *ISX4000 IPM User Manual* for the valid value of echo cancellation mark.

MinJitterBuffer： Please refer to *ISX4000 IPM User Manual* for the valid value of minimum Jitter buffer delay.

MaxJitterBuffer： Please refer to *ISX4000 IPM User Manual* for the valid value of maximum Jitter buffer delay.

AdaptationRate： Please refer to *ISX4000 IPM User Manual* for the valid value of Jitter buffer adaptation rate.

▪ *M3G_CHAN_TYPE_VOICE_PARAM*

```
typedef struct tagM3G_CHAN_TYPE_VOICE_PARAM{
    IpConfig_IP_addr            SrcIpAddr;
    IpConfig_UdpPort            SrcRtpPort;
    IpConfig_IP_addr            DstIpAddr;
    IpConfig_UdpPort            DstRtpPort;
    IpConfig_UdpPort            SrcRTCPPort;    // Source RTCP port number
    IpConfig_UdpPort            DstRTCPPort;    // Destination RTCP port number
    UCHAR                       ucConnMode;     // Connection mode
    UCHAR                       ucVoiceMode;    // Working mode of voice channel
    M3G_VOICE_ATTR_CONFIG       VoiceCfg;       // Voice configuration
    DtmfConfig                  DtmfCfg;        // DTMF configuration
} M3G_CHAN_TYPE_VOICE_PARAM;
```

Description

M3G_CHAN_TYPE_VOICE_PARAM data structure defined the voice attributes of M3G voice channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_CHAN_TYPE_VOICE_PARAM** is described as follows:

SrcIpAddr：Please refer to *ISX4000 IPM User Manual* for the valid value of source RTP IP address.

SrcRtpPort：Please refer to *ISX4000 IPM User Manual* for the valid value of source RTP port.

DstIpAddr：Please refer to *ISX4000 IPM User Manual* for the valid value of destination RTP IP address.

DstRtpPort：Please refer to *ISX4000 IPM User Manual* for the valid value of destination RTP port.

SrcRTCPPort：Please refer to *ISX4000 IPM User Manual* for the valid value of source RTCP port.

DstRTCPPort：Please refer to *ISX4000 IPM User Manual* for the valid value of destination RTCP port.

ucConnMode：Please refer to **eCONN_MODE** in *ISX4000 IPM User Manual* for the connection mode:

```
typedef enum{
    ConnMode_FullDup,       //Full duplex
    ConnMode_RecvOnly,      //Receive only
    ConnMode_SendOnly,      // Send only
    ConnMode_Hold,          // Hold, namely do not receive and send voice data
}eCONN_MODE;
```

VoiceMode：Please refer to **eM3G_VOICE_MODE** for the valid value of working mode of voice channel.

VoiceCfg：Please refer to **M3G_VOICE_ATTR_CONFIG** for the valid value of voice attribute configuration.

DtmfCfg：Please refer to *ISX4000 IPM User Manual* for the valid value of DTMF configuration.

▪ *M3G_VIDEO_RTP*

```
typedef struct tagM3G_VIDEO_RTP{
    UCHAR  ucRtpPayloadType;      // RTP payload type
    ULONG  ulRtpPayloadEncap;     // Encapsulation type of RTP compressed video data
    USHORT usSrcUdpPort;          // RTP source port number
} M3G_VIDEO_RTP;
```

Description

M3G_VIDEO_RTP data structure defined the video RTP attributes. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_RTP** is described as follows:

ucRtpPayloadType：RTP payload type

ulRtpPayloadEncap：Please refer to **eM3G_VIDEO_RTP_PAYLOAD** for the encapsulation type of RTP compressed video data.

usSrcUdpPort: RTP source port number.

▪ *M3G_VIDEO_RTCP*

```
typedef struct tagM3G_VIDEO_RTCP{
    UCHAR     ucValid;          // the structure valid flag
    USHORT    usSrcUdpPort;     // Source port number
    USHORT    usDstUdpPort;     // Destination port number
    char      DstIpAddr[32];    // Destination IP address
} M3G_VIDEO_RTCP;
```

Description

M3G_VIDEO_RTCP data structure defined the video RTCP attributes. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_RTCP** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

usSrcUdpPort: Receive/send UPD port number of RTCP.

usDstUdpPort: remote port number of RTCP

DstIpAddr: remote IP address of RTCP

▪ *M3G_VIDEO_INPUT*

```
typedef struct tagM3G_VIDEO_INPUT{
    UCHAR                ucValid;          // the structure valid flag
    UCHAR                ucIsEnable;       // the video input module enable flag
    UCHAR                ucTransportType;  // Please refer to eM3G_VIDEO_TRANSPORT_TYPE
    M3G_VIDEO_RTP        Rtp;              // RTP attributes
    M3G_VIDEO_RTCP       Rtcp;             // RTCP attributes
} M3G_VIDEO_INPUT;
```

Description

M3G_VIDEO_INPUT data structure defined the attributes of the video input module. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_INPUT** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucIsEnable: The video input modle enable flag

0: disabled or 1: enabled.

ucTransportType: Please refer to **eM3G_VIDEO_TRANSPORT_TYPE** for the transport type

Rtp: Please refer to **M3G_VIDEO_RTP** for RTP attributes.

Rtcp: Please refer to **M3G_VIDEO_RTCP** for RTCP attributes.

▪ *M3G_VIDEO_DECODER*

```
typedef struct tagM3G_VIDEO_DECODER{
    UCHAR  ucValid;              // the structure valid flag
    UCHAR  ucIsEnable;           // the decoder module enable flag
    ULONG  ulCodingStandard;     // Coding standard
    ULONG  ulInputHeight;        // Input height
    ULONG  ulInputWidth;         // Input width
    ULONG  ulOutputHeight;       // Output height (YUV)
    ULONG  ulOutputWidth;        // Output width (YUV)
```

```
    ULONG  ulFramesPerSec;          // FPS: frames per second
    ULONG  ulDeblocking_smoth;      // Smoothness between blocks
    ULONG  ulDeblocking_strength;   // Block boundary strength
    ULONG  ulReferFrames;           // The maximum number of reference frames (reserved for H.264)
} M3G_VIDEO_DECODER;
```

Description

M3G_VIDEO_DECODER data structure defined the attributes of the video decode module. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_DECODER** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucIsEnable: The video decoder module enable flag

0: disabled or 1: enabled.

ulCodingStandard: Please refer to **eM3G_VIDEO_CODER** for the coding standard.

ulInputHeight: Input height. range: 16~288, must be a multiple of 16.

ulInputWidth: Input width. range: 16~352, must be a multiple of 16.

ulOutputHeight: Output height (YUV). range: 48~576, must be a multiple of 16.

ulOutputWidth: Output width (YUV). range: 48~720, must be a multiple of 16.

ulFramesPerSec: FPS: frames per second. range: 4~30.

ulDeblocking_smoth: Smoothness between blocks. range: 0~0x7FFF. 0 means that there is no such function.

ulDeblocking_strength: Block boundary strength. range: 0~0x7FFF. 0 means that there is no such function.

ulReferFrames: The maximum number of reference frames (reserved for H.264). range: 0~11.

▪ *M3G_VIDEO_ENCODER*

```
typedef struct tagM3G_VIDEO_ENCODER{
    UCHAR  ucValid;                 // The structure valid flag
    UCHAR  ucIsEnable;              // The encode module enable flag
    ULONG  ulCodingStandard;        // Coding standard
    ULONG  ulProfileLevel;          // Profile 和 Level  Video compression Profile and Level
    ULONG  ulInputHeight;           // Input height (YUV)
    ULONG  ulInputWidth;            // Input width (YUV)
    ULONG  ulFramesPerSec;          // FPS: frames per second
    ULONG  ulBitRate;               // Compressed Bit rate
    UCHAR  ucIsVariableBitRate;     // Whether is the variable bit rate
    ULONG  ulVbrQualityMin;         // Quality minimum of VBR compression
    ULONG  ulVbrQualityMax;         // Quality maximum of VBR compression
    ULONG  ulIFrameFrequency;       // I frame frequency. Default value: 99.
} M3G_VIDEO_ENCODER;
```

Description

M3G_VIDEO_ENCODER data structure defined the attributes of video encode module. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_ENCODER** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucIsEnable: The video encode module enable flag

0: disabled or 1: enabled.

ulCodingStandard: Please refer to **eM3G_VIDEO_CODER** for the coding standard.

ulProfileLevel:

Please refer to **eM3G_VIDEO_MPEG4_SIMPLE_PROFILE_LEVEL** for the video compression Profile and Level

ulInputHeight： Input height (YUV). range: 16~576.

ulInputWidth： Input width (YUV). range: 16~720.

ulFramesPerSec： FPS: frames per second. range: 4~30.

ulBitRate：

Compressed Bit rate. The following must be met:

If the output resolution is CIF or above, BitRate> = 8000 * FPS; otherwise, BitRate> = 2000 * FPS. (FPS: frames per second).

ucIsVariableBitRate： Whether is the variable bit rate

0: no or 1: yes.

ulVbrQualityMin： Quality minimum of VBR compression. range: 0~0x7FFF.

ulVbrQualityMax： Quality maximum of VBR compression. range: 0~0x7FFF.

ulIFrameFrequency： I frame frequency. Default value: 99.

▪ *M3G_VIDEO_DEST_ITEM*

```
typedef struct tagM3G_VIDEO_DEST_ITEM{
    char DstIpAddr[MAX_IP_LEN];          // RTP destination IP
    USHORT usDstUdpPort;                 // RTP destination UDP port
    char DstRtcpIpAddr[MAX_IP_LEN];      // If there is no RTCP, it will be NULL string.
    USHORT usDstRtcpUdpPort;             // If there is no RTCP, it will be 0.
} M3G_VIDEO_DEST_ITEM;
```

Description

M3G_VIDEO_DEST_ITEM data structure defined the attributes of the video output destinations. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_DEST_ITEM** is described as follows:

DstIpAddr： RTP destination IP address.

usDstUdpPor： RTP destination UDP port.

DstRtcpIpAddr： RTCP address; if there is no RTCP, it will be NULL string.

usDstRtcpUdpPort： RTCP port; if there is no RTCP, it will be 0.

▪ *M3G_VIDEO_DEST*

```
typedef struct tagM3G_VIDEO_DEST{
    UCHAR  ucDestNum;          // At most 16 destinations can be 0.
    M3G_VIDEO_DEST_ITEM        DestItem[MAX_VIDEO_DEST_ITEM];
} M3G_VIDEO_DEST;
```

Description

M3G_VIDEO_DEST data structure defined the more video output destinations. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_DEST** is described as follows:

ucDestNum： Destination number, MAX_VIDEO_DEST_ITEM=16.

DestItem： Please refer to **M3G_VIDEO_DEST_ITEM** for the destination attributes.

▪ *M3G_VIDEO_OUTPUT*

```
typedef struct tagM3G_VIDEO_OUTPUT{
    UCHAR          ucValid;          // The structure valid flag
    UCHAR          ucIsEnable;       // The output module enable flag
```

```
    UCHAR           ucTransportType;  // Please refer to eM3G_VIDEO_TRANSPORT_TYPE
    M3G_VIDEO_RTP   Rtp;              // VIDEO output RTP attributes
    M3G_VIDEO_RTCP  Rtcp;             // output RTCP attributes
    M3G_VIDEO_DEST  Dest;             // output destinations
} M3G_VIDEO_OUTPUT;
```

Description

M3G_VIDEO_OUTPUT data structure defined the attributes of the video output module. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_VIDEO_OUTPUT** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucIsEnable: The video output module enable flag

0: disabled or 1: enabled.

ucTransportType: Please refer to **eM3G_VIDEO_TRANSPORT_TYPE** for the transport type

Rtp: Please refer to **M3G_VIDEO_RTP** for RTP attributes

Rtcp: Please refer to **M3G_VIDEO_RTCP** for RTCP attributes.

Dest: Please refer to **M3G_VIDEO_DEST** for output destinations.

▪ *M3G_CHAN_TYPE_VIDEO_PARAM*

```
typedef struct tagM3G_CHAN_TYPE_VIDEO_PARAM{
    M3G_VIDEO_INPUT     VideoInput;     // Input module attributes
    M3G_VIDEO_DECODER   VideoDecoder;   // Decode module attributes
    M3G_VIDEO_ENCODER   VideoEncoder;   // ecode module attributes
    M3G_VIDEO_OUTPUT    VideoOutput;    // Output module attributes
} M3G_CHAN_TYPE_VIDEO_PARAM;
```

Description

M3G_CHAN_TYPE_VIDEO_PARAM data structure defined the attributes of the VIDEO channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_CHAN_TYPE_VIDEO_PARAM** is described as follows:

VideoInput: Please refer to **M3G_VIDEO_INPUT** for the valid value of video input module attributes.

VideoDecoder: Please refer to **M3G_VIDEO_DECODER** for the valid value of video decode module attributes.

VideoEncoder: Please refer to **M3G_VIDEO_ENCODER** for the valid value of video encode module attributes.

VideoOutput: Please refer to **M3G_VIDEO_OUTPUT** for the valid value of video output module attributes.

▪ *M3G_TERM_CAPS*

```
typedef struct tagM3G_TERM_CAPS{
    UCHAR  ucValid;             // The structure valid flag
    UCHAR  ucAudioCodes[4];     // Simultaneously supported audio encoding type
    UCHAR  ucVideoCodes[4];     // Simultaneously supported video encoding type
    UCHAR  ucDtmfSignal[2];     // Simultaneously supported DTMF transport type
} M3G_TERM_CAPS;
```

Description

M3G_TERM_CAPS data structure defined the Terminal Capabilities set of H324M channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_TERM_CAPS** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucAudioCodes:

Simultaneously supported audio encoding type includes at most 4 types; please refer to **eM3G_H324_AUDIO_CODEC** for the valid value.

ucVideoCodes:

Simultaneously supported video encoding type includes at most 4 types; please refer to **eM3G_H324_VIDEO_CODEC** for the valid value.

ucDtmfSignal: Simultaneously supported DTMF transport type includes at most 2 types

0: no or 1: yes.

ucDtmfSignal[0] indicates whether to support the string mode to send DTMF;

ucDtmfSignal[1] indicates whether to support the single send.

▪ *M3G_H223_PARAMS*

```
typedef struct tagM3G_H223_PARAMS{
    UCHAR  ucValid;                    // The structure valid flag
    ULONG  ulAl2WithSeqNumberAudio;    // The audio with or without sequence number
    ULONG  ulAl2WitchSeqNumberVideo;   // The video with or without sequence number
    ULONG  ulMaxPdu;                   // Maximum PDU length
} M3G_H223_PARAMS;
```

Description

M3G_H223_PARAMS data structure defined the H.223 protocol parameter. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_H223_PARAMS** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ulAl2WithSeqNumberAudio: The audio with or without sequence number

0: no or 1: yes.

ulAl2WitchSeqNumberVideo: The video with or without sequence number

0: no or 1: yes.

ulMaxPdu: The maximum PDU length supported by the local terminal. Default value: 160.

▪ *M3G_MEDIA_CHAN*

```
typedef struct tagM3G_MEDIA_CHAN{
    UCHAR      ucValid;          // The structure valid flag
    UCHAR      ucRtpPayloadType; // RTP Payload type
    USHORT     usSrcUdpPort;     // Source UDP port
    USHORT     usDstUdpPort;     // Destination UDP port
    char       DstIpAddr[32];    // Destination IP address
    ULONG      ulCodec;          // codec type of audio/video
    UCHAR      ucIsMuted;        // mute flag
} M3G_MEDIA_CHAN;
```

Description

M3G_MEDIA_CHAN data structure defined the media configuration of logic channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_MEDIA_CHAN** is described as follows:

ucValid: The structure valid flag

0: invalid or 1: valid.

ucRtpPayloadType: RTP Payload type.

usSrcUdpPort: Source UDP port.

usDstUdpPort: Destination UDP port.

DstIpAddr: Destination IP address.

ulCodec:

Codec type of audio/video; if the media type is audio, please refer to **eM3G_H324_AUDIO_CODEC**; if the media type is video, please refer to **eM3G_H324_VIDEO_CODEC**.

ucIsMuted: muted flag

0: no or 1: yes.

▪ *M3G_CHAN_TYPE_H324M_PARAM*

```
typedef struct tagM3G_CHAN_TYPE_H324M_PARAM{
    UCHAR                   ucTermType;     // Terminal type
    M3G_TERM_CAPS           LocalCaps;      // Local TCS
    M3G_H223_PARAMS         H223Params;     // H.223 parameters
    M3G_MEDIA_CHAN          MediaChan[4];   // Media configurations of logic channels
} M3G_CHAN_TYPE_H324M_PARAM;
```

Description

M3G_CHAN_TYPE_H324M_PARAM data structure defined the parameter of H324M channel. The definitions are in the m3glib.h header file.

Domain Description

The **M3G_CHAN_TYPE_H324M_PARAM** is described as follows:

ucTermType: Terminal type

LocalCaps: Please refer to **M3G_TERM_CAPS** for the valid value of local TCS.

H223Params: Please refer to**M3G_H223_PARAMS** for the valid value of H.223 parameters.

MediaChan: Please refer to **M3G_MEDIA_CHAN** for the valid value of media configuration of logic channels, in which:

MediaChan[0] for the video output logical channel.

MediaChan[1] for the audio output logical channel.

MediaChan[2] for the video input logical channel.

MediaChan[3] for the audio input logical channel.

▪ *M3G_CHAN_PARAM*

```
typedef struct tagM3G_CHAN_PARAM {
    UCHAR       ucChanType;         // M3G channel type
    union {
        M3G_CHAN_TYPE_VOICE_PARAM voice;   // Voice channel attributes
        M3G_CHAN_TYPE_VIDEO_PARAM video;   // Video channel attributes
        M3G_CHAN_TYPE_H324M_PARAM h324m; // H324M channel attributes
    };
}M3G_CHAN_PARAM, *PM3G_CHAN_PARAM;
```

Description

M3G_CHAN_PARAM data structure defined the attributes of M3G channels. The definitions are in the m3glib.h header file. The M3G channel types are VOICE, VIDEO and H324M. the attributes corresponding to each different channel type are completely different, each is corresponding to a data structure:

When the channel is VOICE type, please use **M3G_CHAN_TYPE_VOICE_PARAM** structure;

When the channel is VIDEO type, please use **M3G_CHAN_TYPE_VIDEO_PARAM** structure;

When the channel is H324M type, please use **M3G_CHAN_TYPE_H324M_PARAM** structure.

Domain Description

The **M3G_CHAN_PARAM** is described as follows:

ucChanType： Please refer to **eM3G_CHAN_TYPE** for the valid value of M3G channel type.
voice：Please refer to **M3G_CHAN_TYPE_VOICE_PARAM** for the valid attributes of VOICE channel.
video：Please refer to **M3G_CHAN_TYPE_VIDEO_PARAM** for the valid attributes of VIDEO channel.
h324m：Please refer to **M3G_CHAN_TYPE_H324M_PARAM** for the valid attributes of H324M channel.

▪ *M3G_CHAN_EXINFO*

```
typedef struct tagM3G_CHAN_EXINFO{
    UCHAR ucValid; //0：invlid；1：uVoiceRelated； 2：uH324mRelated
    union{
        struct { // Only valid for voiceRelated
            M3GDEV VideoDev;// Video channel handle on simultaneously playing/recording audio/video file
        }uVoiceRelated;
        struct { // Only valid for H324MRelated
            UCHAR    ucAudioMode;// Voice mode
            M3GDEV   AudioDev;   // Voice channel handle
            UCHAR    ucVideoMode;// Video mode
            M3GDEV   VideoDev;   // Video channel handle
        }uH324mRelated;
    };
}M3G_CHAN_EXINFO, *PM3G_CHAN_EXINFO;
```

Description

M3G_CHAN_EXINFO data structure defined the extension channel information when the file recording or playing. When the ucValid is 0, the all the structure is invalid; when the ucValid is 1, the uVoiceRelated structure is valid; when the ucValid is 2, the uH324mRelated structure is valid; the definitions are in the m3glib.h header file.

When the application use the VOICE channel to do the audio/video file recording or playing, the VIDEO channel handle was set in the uVoiceRelated of **M3G_CHAN_EXINFO**. The video channel is used to process the VIDEO of the file.

When the channel is H324M channel, the uH324mRelated of **M3G_CHAN_EXINFO** is used to set the transcoding voide and video channel handles; There are a number of conditions shown in the following table:

index	Function description	uH324mRelated	Remarks
1	Only the the voice is recorded or played, and use the pass through mode.	ucAudioMode=1; ucVideoMode=0;	AudioDev is invalid VideoDev is invalid
2	Only the voice is recorded or played, and uses the transcode mode.	ucAudioMode=2; ucVideoMode=0;	AudioDev: VOICE channel handle VideoDev is invalid
3	Only the video is recorded or played, and uses the pass through mode.	ucAudioMode=0; ucVideoMode=1;	VideoDev is invalid VideoDev is invalid
4	Only the video is recorded or played, and uses the transcode mode.	ucAudioMode=0; ucVideoMode=2;	AudioDev is invalid VideoDev: VIDEO channel handle
5	The voice and video are recorded or played simultaneously, and the voice uses the pass through mode and the video also uses the pass through mode.	ucAudioMode=1; ucVideoMode=1;	VideoDev is invalid VideoDev is invalid
6	The voice and video are recorded or played simultaneously, and the voice uses the pass through mode and the video uses the transcode mode.	ucAudioMode=1; ucVideoMode=2;	VideoDev is invalid VideoDev: VIDEO channel handle
7	The voice and video are recorded or played simultaneously, and the voice uses the transcode mode and the video uses the	ucAudioMode=2; ucVideoMode=1;	AudioDev: VOICE channel handle VideoDev is invalid

	pass through mode.		
8	The voice and video are recorded or played simultaneously, and the audio uses the transcode mode and the video also uses the transcode mode.	ucAudioMode=2; ucVideoMode=2;	AudioDev: VOICE channel handle VideoDev: VIDEO channel handle

Domain Description

The **M3G_CHAN_EXINFO** is described as follows:

ucValid: The mark indicating The structure valid flag

0: invalid, 1: uVoiceRelated or 2: uH324mRelated.

VideoDev: M3G video channel handle (retrived by ISX_m3g_OpenEx).

ucAudioMode: The voice mode

0: no operation,

1: pass through mode or

2: transcode mode. Please refer to **eM3G_MEDIA_CHAN_OPMODE**.

AudioDev: M3G voice channel handle (retrived by ISX_m3g_OpenEx).

ucVideoMode: The video mode

0: no operation,

1: pass through mode or

2: transcode mode. Please refer to **eM3G_MEDIA_CHAN_OPMODE.**

VideoDev: M3G video channel handle (retrived by ISX_m3g_OpenEx).

▪ *M3G_FILE_WAV_PARAMS*

```
typedef struct tagM3G_FILE_WAV_PARAMS{
    ULONG  ulCoder;      /* encoder */
    ULONG  ulBlockSize;  /* Data block length*/
} M3G_FILE_WAV_PARAMS;
```

Description

M3G_FILE_WAV_PARAMS data structure defined the parameter of WAV file format. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_WAV_PARAMS** is described as follows:

ulCoder: Please refer to **eM3G_VOICE_CODER** for the valid audio encoder types.

ulBlockSize: Please refer to **eM3G_WAV_BLOCK_SIZE** for the data block length.

▪ *M3G_FILE_AV_PLAY_TRACK_PARAM*

```
typedef struct tagM3G_FILE_AV_PLAY_TRACK_PARAM{
    UCHAR    ucSampleType;        /* Media sample type */
    UCHAR    ucTimeStampSource;   /* Timestamp source*/
    USHORT   usFPS;               /* FPS: frame per second */
} M3G_FILE_AV_PLAY_TRACK_PARAM;
```

Description

M3G_FILE_AV_PLAY_TRACK_PARAM data structure defined the media track informations of non-WAV format (3GP/MP4/AVI/ASF/3G2) when playing audio/video files. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_AV_PLAY_TRACK_PARAM** is described as follows:

ucSampleType: Please refer to **eM3G_SAMPLE_TYPE** for the media sample type.

ucTimeStampSource: Please refer to **eM3G_FILE_TIMING_SOURCE_SAMPLE** for the timestamp source.

usFPS: FPS (frame per second) can be 1~30.

▪ *M3G_FILE_AV_PLAY_PARAMS*

```
typedef struct tagM3G_FILE_AV_PLAY_PARAMS{
        UCHAR  ucTrackNum;              /* Number of media tracks in the file*/
        M3G_FILE_AV_PLAY_TRACK_PARAM TrackParam[2];     /* Media track information*/
} M3G_FILE_AV_PLAY_PARAMS;
```

Description

M3G_FILE_AV_PLAY_PARAMS data structure defined the parameter of non-WAV format (3GP/MP4/AVI/ASF/3G2) when playing audio/video files. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_AV_PLAY_PARAMS** is described as follows:

ucTrackNum: Number of media tracks in the file, range:0~1.

TrackParam: Please refer to **M3G_FILE_AV_PLAY_TRACK_PARAM** for the media track information.

▪ *M3G_FILE_AV_REC_TRACK_PARAM*

```
typedef struct tagM3G_FILE_AV_REC_TRACK_PARAM{
    UCHAR       ucSampleType;       /* Media sample type */
    UCHAR       ucTimeStampSource;  /* Timestamp source*/
    ULONG       ulWidth;            /* width of video frame in dots */
    ULONG       ulHeight;           /* height of video frame in dots */
    USHORT      usFPS;              /* FPS: frame per second */
} M3G_FILE_AV_REC_TRACK_PARAM;
```

Description

M3G_FILE_AV_REC_TRACK_PARAM data structure defined the media tracks of non-WAV format (3GP/MP4/AVI/ASF/3G2) when recording audio/video files. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_AV_REC_TRACK_PARAM** is described as follows:

ucSampleType: Please refer to **eM3G_SAMPLE_TYPE** for the media sample type.

ucTimeStampSource: Please refer to **eM3G_FILE_TIMING_SOURCE_SAMPLE** for the timestamp source.

ulWidth: The lattice width of video frame is valid at WMV9 and YUV; range: 16~720.

ulHeight: The lattice height of video frame is valid at WMV9 and YUV; range: 16~576.

usFPS: FPS (frame per second) can be 1~30.

▪ *M3G_FILE_AV_REC_PARAMS*

```
typedef struct tagM3G_FILE_AV_REC_PARAMS{
    UCHAR  ucHintTracks;                    /* index track flag */
    UCHAR  ucTrackNum;                      /* Number of media tracks in the file */
    M3G_FILE_AV_REC_TRACK_PARAM             TrackParam[2];      /* Media track information*/
} M3G_FILE_AV_REC_PARAMS;
```

Description

M3G_FILE_AV_REC_PARAMS data structure defined the parameters of non-WAV format (3GP/MP4/AVI/ASF/3G2) when recording audio/video files. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_AV_REC_PARAMS** is described as follows:

ucHintTracks: index track flag,

0: no index track or 1: has index track.

ucTrackNum: Number of media tracks in the file, range: 0~1.

TrackParam: Please refer to **M3G_FILE_AV_PLAY_TRACK_PARAM** for the media track information.

▪ *M3G_XPB*

```
typedef struct tagM3G_XPB{
    UCHAR       ucFileType;
    union {
        M3G_FILE_WAV_PARAMS          wav;
        M3G_FILE_AV_PLAY_PARAMS      avp;
        M3G_FILE_AV_REC_PARAMS       avr;
    };
}M3G_XPB;
```

Description

M3G_FILE_AV_REC_PARAMS data structure defined the audio and video data formats during the playing/recording, which has different explanations for binary files (BIN without parameters), WAV files and audio/video files (3GP/MP4/AVI/ASF/3G2); for audio/video files (3GP/MP4/AVI/ASF/3G2), playing and recording have also different explanations. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_XPB** is described as follows:

ucFileType：

The file type currently supports binary files, wav files and audio/video files (3GP/MP4/AVI/ASF/3G2); please refer to **eM3G_FILE_TYPE**.

wav： Please refer to **M3G_FILE_WAV_PARAMS** for the wav file parameters.

avp：

Please refer to **M3G_FILE_AV_PLAY_PARAMS** for non-wav audio/video file (3GP/MP4/AVI/ASF/3G2) parameters during playing.

avr：

Please refer to **M3G_FILE_AV_REC_PARAMS** for non-wav audio/video file (3GP/MP4/AVI/ASF/3G2) parameters during recording.

Note: the av_play is for playing, and av_rec is for recording.

▪ *M3G_DIGIT_INFO*

```
typedef struct tagM3G_DIGIT{
    UCHAR  ucDigitType;                 // Digit type
    UCHAR  ucDigitDir;                  // Digit direction
    UCHAR  ucDigitNum;                  //Digit The number of Digits in the Digit buffer
    char   Digits[MAX_IPM_DIGITS];      // Digit buffer
    UCHAR  ucDuration;                  // Digit activity duration
    UCHAR  ucInterval;                  // Interval between Digits
} M3G_DIGIT_INFO;
```

Description

M3G_DIGIT_INFO data structure defined the DIGITS information used by the **ISX_m3g_SendDigits()** function. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_DIGIT_INFO** is described as follows:

ucDigitType： Digit type, range: 0~6; please refer to *ISX4000 IPM User Manual* for the valid value.

ucDigitDir： Digit direction, range: 0~2; please refer to *ISX4000 IPM User Manual* for the valid value.

ucDigitNum： The number of Digits in the Digit buffer, maximum value: 32.

Digits： Digit buffer (MAX_IPM_DIGITS=32).

ucDuration： Digit activity duration.

ucInterval： Interval between Digits.

▪ *M3G_TCS_INFO*

```
typedef struct tagM3G_TCS_STATUS_INFO{
    UCHAR          ucTCSStatus;    // TCS signal status
    M3G_TERM_CAPS  localcaps;      // Local terminal capacity
    M3G_TERM_CAPS  remotecaps;     // Remote terminal capacity
    M3G_TERM_CAPS  matchcaps;      // Matching terminal capacity
}M3G_TCS_INFO;
```

Description

M3G_TCS_INFO data structure defined the TCS signal status and TCS capacities. When received the event **M3GEV_TCSINFO_STATUS** , the data information mapped to this structure. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_FILE_AV_PLAY_TRACK_PARAM** is described as follows:

ucTCSStatus： Please refer to **eM3G_TCS_STATUS** for the TCS signal status.

localcaps： Please refer to **M3G_TERM_CAPS** for the local terminal capacity.

remotecaps： Please refer to **M3G_TERM_CAPS** for the remote terminal capacity.

matchcaps： Please refer to **M3G_TERM_CAPS** for the matching terminal capacity.

▪ *M3G_VIDEO_FAST_UPDATE_GOB*

```
typedef struct tagM3G_VIDEO_FAST_UPDATE_GOB{
    ULONG  ulFirstGOB;          // Specify the video group block number starting to refresh
    ULONG  ulNumberOfGOBs;      // Specify the number of refreshed group blocks
} M3G_VIDEO_FAST_UPDATE_GOB;
```

Description

M3G_VIDEO_FAST_UPDATE_GOB data structure defined the parameters when the MISC command ID (**eM3G_MISC_ID**) is **M3G_MISC_ID_VIDEO_FAST_UPDATE_GOB**. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_VIDEO_FAST_UPDATE_GOB** is described as follows:

ulFirstGOB： Specify the video group block number starting to refresh.

ulNumberOfGOBs： Specify the number of refreshed group blocks.

▪ *M3G_VIDEO_FAST_UPDATE_MB*

```
typedef struct tagM3G_VIDEO_FAST_UPDATE_MB{
    ULONG  ulFirstGOB;          // Specify the video group block number starting to refresh
    ULONG  ulFirstMB;           // Specify the video macroblock number starting to refresh
    ULONG  ulNumberOfMBs;       // Specify the number of refreshed macroblocks
} M3G_VIDEO_FAST_UPDATE_MB;
```

Description

M3G_VIDEO_FAST_UPDATE_MB data structure defined the parameters when the MISC command ID (**eM3G_MISC_ID**) is **M3G_MISC_ID_VIDEO_FAST_UPDATE_MB**. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_VIDEO_FAST_UPDATE_MB** is described as follows:

ulFirstGOB： Specify the video group block number starting to refresh.

ulFirstMB： Specify the video macroblock number starting to refresh.

ulNumberOfMBs： Specify the number of refreshed macroblocks.

▪ *M3G_SKEW_INDICATION*

```
typedef struct tagM3G_SKEW_INDICATION{
    ULONG  ulSkew;        // The maximum delay from the first logical channel to the second logical channel
} M3G_SKEW_INDICATION;
```

Description

M3G_SKEW_INDICATION data structure defined the parameters when the MISC command ID (**eM3G_MISC_ID**) is **M3G_MISC_ID_SKEW_INDICATION**. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_SKEW_INDICATION** is described as follows:

ulSkew： The maximum delay from the first logical channel to the second logical channel, unit: ms.

▪ *M3G_H245_MISC_INFO*

```
typedef struct tagM3G_H245_MISC_INFO{
    UCHAR       ucLogicChanNo;
    ULONG       ulMiscCmdId;
    union{
        M3G_VIDEO_FAST_UPDATE_GOB     vfug;
        M3G_VIDEO_FAST_UPDATE_MB      vfum;
        M3G_SKEW_INDICATION           si;
    };
} M3G_H245_MISC_INFO;
```

Description

M3G_H245_MISC_INFO data structure defined the MISC command parameters in the H245 signal process. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_H245_MISC_INFO** is described as follows:

ucLogicChanNo： Please refer to **eM3G_LC_NUMBER** for the valid value of logical channel number.

ulMiscCmdId： Please refer to **eM3G_MISC_ID** for the MISC command ID.

vfug：

Please refer to **M3G_VIDEO_FAST_UPDATE_GOB** for the parameters when the MISC command ID is M3G_MISC_ID_VIDEO_FAST_UPDATE_GOB.

vfum：

Please refer to **M3G_VIDEO_FAST_UPDATE_MB** for the parameters when the MISC command ID is M3G_MISC_ID_VIDEO_FAST_UPDATE_MB.

si：

Please refer to **M3G_SKEW_INDICATION** for the parameters when the MISC command ID is M3G_MISC_ID_SKEW_INDICATION.

▪ *M3G_LC_PARAM*

```
typedef struct tagM3G_LC_PARAM{
    UCHAR               ucLogicChanNo;   // Logical channel number
    UCHAR               ucLCStatus;      // Logical channel status
    M3G_MEDIA_CHAN      parm;            // Logical channel parameter
} M3G_LC_PARAM;
```

Description

M3G_LC_PARAM data structure defined the parameters of H324M logical channel. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_LC_PARAM** is described as follows:

ucLogicChanNo： Please refer to **eM3G_LC_NUMBER** for the valid value of logical channel number.
ucLCStatus： Please refer to **eM3G_LC_STATUS** for the valid value of logical channel status.
parm： Please refer to **M3G_MEDIA_CHAN** for the valid value of logical channel parameters.

▪ *M3G_CALL_STATUS*

```
typedef struct tagM3G_CALL_STATUS {
    UCHAR  ucLinkStatus;  // Link status
    UCHAR  ucCPStatus;    // Call progress status
} M3G_CALL_STATUS;
```

Description

M3G_CALL_STATUS data structure defined the signal status during the H324M calls. When received the event **M3GEV_CALL_STATUS** or **M3GEV_GET_CALL_STATUS_CMPLT** , the data information mapped as this structure. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_CALL_STATUS** is described as follows:
ucLinkStatus： Please refer to **eM3G_H223_LINK_STATUS** for the valid value of H223 link status.
ucCPStatus： Please refer to **eM3G_CP_STATUS** for the valid value of call progress status.

▪ *M3G_MSD_INFO*

```
typedef struct tagM3G_MSD_INFO{
    UCHAR  ucMsdStatus;          // MSD signal status
    UCHAR  ucLocalIsMaster;      // Local terminal is master
    UCHAR  ucLocalTermType;      // Local terminal type
    UCHAR  ucRemoteTermType;     // Remote terminal type
} M3G_MSD_INFO;
```

Description

M3G_MSD_INFO data structure defined the MSD information. When received the event **M3GEV_MSDINFO_STATUS**, the data information mapped as the structure. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_MSD_INFO** is described as follows:
ucMsdStatus：Please refer to **eM3G_MSD_STATUS** for the valid value of MSD signal status.
ucLocalIsMaster：The local terminal is master,
0: no or 1: yes.
ucLocalTermType： Local terminal type.
ucRemoteTermType： Remote terminal type.

▪ *M3G_LC_INFO*

```
typedef struct tagM3G_LC_INFO{
    UCHAR  ucLogicChanNo;    // Logical channel number
    UCHAR  ucLCStatus;       // Logical channel status
    ULONG  ulCodec;          // codec type of audio and video
    UCHAR  ucIsMuted;        // Mute flag
} M3G_LC_INFO;
```

Description

M3G_LC_INFO data structure defined the signal status information of H324M logical channel. When received the event **M3GEV_LCINFO_STATUS**, mapped the data information to this structure. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_LC_INFO** is described as follows:

ucLogicChanNo： Please refer to **eM3G_LC_NUMBER** for the valid value of logical channel number.

ucLCStatus： Please refer to **eM3G_LC_STATUS** for the valid value of logical channel status.

ulCodec：codec type of audio and video;
if the media type is VOICE, please refer to **eM3G_H324_AUDIO_CODEC**;
if the media type is VIDEO, please refer to **eM3G_H324_VIDEO_CODEC**.

ucIsMuted：Mute flag; if it is Muted, it will not send data to the remote end, and it can be 0: no or 1: yes.

▪ *M3G_TOOLBOX_PARAM*

```
typedef struct tagM3G_TOOLBOX_PARAM{
    UCHAR  ucValid;        /* the structure valid flag */
    UCHAR  ucIsEnable;     /* toolbox enable flag */
    UCHAR  ucFPS;          /* FPS*/
    UCHAR  ucRev;
    ULONG  ulInputHeight;
    ULONG  ulInputWidth;
    ULONG  ulOutputHeight;
    ULONG  ulOutputWidth;
} M3G_TOOLBOX_PARAM;
```

Description

M3G_TOOLBOX_PARAM data structure defined the toolbox parameters. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TOOLBOX_PARAM** is described as follows:

ucValid：the structure valid flag,
0: invalid or 1: valid.

ucIsEnable：the toolbox enable flag,
0: disable or 1: enable.

ucFPS： FPS

ucRev： Reserved domain

ulInputHeight： Input height

ulInputWidth： Input width

ulOutputHeight： Output height

ulOutputWidth： Output width

▪ *M3G_LOGO_PARAM*

```
typedef struct tagM3G_LOGO_PARAM{
    UCHAR  ucValid;          /* the structure valid flag */
    ULONG  ulHeight;         /* LOGO image height, unit: pixel, 0~144 */
    ULONG  ulWidth;          /* LOGO image width, unit: pixel, 0~176 */
    ULONG  ulTopPos;         /* LOGO image top coordinate, unit: pixel, 0~575*/
    ULONG  ulLeftPos;        /* LOGO image left coordinate, unit: pixel, 0~719*/
    UCHAR  ucTransY;
    UCHAR  ucTransU;
    UCHAR  ucTransV;
    UCHAR  ucTransAlpha;
    UCHAR  ucAlphaBlending;
} M3G_LOGO_PARAM;
```

Description

M3G_LOGO_PARAM data structure defined the LOGO parameters. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_LOGO_PARAM** is described as follows:

ucValid：the structure valid flag,
0: invalid or 1: valid.

ulHeight：LOGO image height, unit: pixel, 0~144.

ulWidth： LOGO image width, unit: pixel, 0~176.

ulTopPos： LOGO image top coordinate, unit: pixel, 0~575.

ulLeftPos： LOGO image left coordinate, unit: pixel, 0~719.

ucTransY：Y value of YUV.

ucTransU：U value of YUV.

ucTransV：V value of YUV.

ucTransAlpha：Alpha value.

ucAlphaBlending：Alpha Blending value.

▪ *M3G_TOOLBOX_CONFIG*

```
typedef struct tagM3G_TOOLBOX_CONFIG {
    M3G_TOOLBOX_PARAM   toolbox;
    M3G_LOGO_PARAM      logo;
} M3G_TOOLBOX_CONFIG;
```

Description

M3G_TOOLBOX_CONFIG data structure defined the toolbox configurations. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TOOLBOX_CONFIG** is described as follows:

toolbox：Please refer to **M3G_TOOLBOX_PARAM** for toolbox parameters.

logo ： Please refer to **M3G_LOGO_PARAM** for LOGO parameters.

▪ *M3G_TO_DISPLAY_MODE*

```
/* Display mode: show or hide*/
typedef struct tagM3G_TO_DISPLAY_MODE {
    UCHAR  ucValid;     /* the structure valid flag */
    UCHAR  ucMode;      /* 0: hide; 1: show*/
} M3G_TO_DISPLAY_MODE;
```

Description

M3G_TO_DISPLAY_MODE data structure defined the toolbox configurations. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_DISPLAY_MODE** is described as follows:

ucValid：the structure valid flag,
0: invalid or 1: valid.

ucMode：Display mode;
0: hide or 1: show. Please refer to **eM3G_DISPLAY_MODE**.

▪ *M3G_LOGO_CONFIG*

```
typedef struct tagM3G_LOGO_CONFIG {
    M3G_LOGO_PARAM      LogoParm;       /* LOGO parameters */
    M3G_TO_DISPLAY_MODE DisplayMode;    /* Display mode */
} M3G_LOGO_CONFIG;
```

Description

M3G_LOGO_CONFIG data structure defined the LOGO configurations. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_LOGO_CONFIG** is described as follows:

LogoParm ： Please refer to **M3G_LOGO_PARAM** for LOGO parameters.

DisplayMode： Please refer to **M3G_TO_DISPLAY_MODE** for LOGO display mode.

▪ *M3G_TO_CARET_POSITION*

```
typedef struct tagM3G_TO_CARET_POSITION {
    UCHAR  ucValid;        /* the structure valid flag */
    UCHAR  ucTextDir;      /* Text direction */
    ULONG  ulLeftPos;      /* 0～351 */
    ULONG  ulTopPos;       /* 0～287 */
} M3G_TO_CARET_POSITION;
```

Description

M3G_TO_CARET_POSITION data structure defined the position information of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_CARET_POSITION** is described as follows:

ucValid： the structure valid flag,

0: invalid or 1: valid.

ucTextDir： Please refer to **eM3G_TEXT_DIRECTION** for text the direction.

ulLeftPos： Left position, range: 0~351.

ulTopPos： Top position, range: 0~287.

▪ *M3G_TO_FONT_INFO*

```
typedef struct tagM3G_TO_FONT_INFO {
    UCHAR  ucValid;          /* the structure valid flag */
    UCHAR  ucFontIndex;      /* Font index currently is 0*/
    ULONG  ulFontSize;       /* Font size: 0~255*/
    ULONG  ulFontStyle;      /* Font style */
} M3G_TO_FONT_INFO;
```

Description

M3G_TO_FONT_INFO data structure defined the font information of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_FONT_INFO** is described as follows:

ucValid： the structure valid flag,

0: invalid or 1: valid.

ucFontIndex： Font index, currently is 0。

ulFontSize： Font size: 0~255.

ulFontStyle： Please refer to **eM3G_TEXT_STYLE** for the font style.

▪ *M3G_TO_COLOR*

```
typedef struct tagM3G_TO_COLOR {
    UCHAR  ucValid;        /* the structure valid flag */
    UCHAR  ucForegroundR;
```

```
    UCHAR  ucForegroundG;
    UCHAR  ucForegroundB;
    UCHAR  ucForegroundAlpha;
    UCHAR  ucBackgroundR;
    UCHAR  ucBackgroundG;
    UCHAR  ucBackgroundB;
    UCHAR  ucBackgroundAlpha;
} M3G_TO_COLOR;
```

Description

M3G_TO_COLOR data structure defined the color information of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_COLOR** is described as follows:

ucValid：the structure valid flag,
0: invalid or 1: valid.
ucForegroundR： R value of foreground RGB.
ucForegroundG： G value of foreground RGB.
ucForegroundB： B value of foreground RGB.
ucForegroundAlpha： Foreground Alpha value.
ucBackgroundR： R value of background RGB.
ucBackgroundG： G value of background RGB.
ucBackgroundB： B value of background RGB.
ucBackgroundAlpha： Background Alpha value.

▪ *M3G_TO_SCROLL_INFO*

```
typedef struct tagM3G_TO_SCROLL_INFO {
    UCHAR  ucValid;             /* the structure valid flag */
    UCHAR  ucEnable;            /* the scroll function enable flag */
    UCHAR  ucScrollDir;         /* Scroll direction */
    UCHAR  ucEndlessLoop;       /* the scroll loop flag */
    ULONG  ulSpeed;             /* Scroll speed: pixels/second */
    ULONG  ulLeftPos;           /* Left position of scroll box*/
    ULONG  ulRightPos;          /* Right position of scroll box*/
    ULONG  ulBackgroundTopPos;
    ULONG  ulBackgroundLeftPos;
    ULONG  ulBackgroundWidth;
    ULONG  ulBackgroundHeight;
} M3G_TO_SCROLL_INFO;
```

Description

M3G_TO_SCROLL_INFO data structure defined the informations of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_SCROLL_INFO** is described as follows:

ucValid：the structure valid flag,
0: invalid or 1: valid.
ucEnable：enable flag,
0: disable or 1: enable;
ucScrollDir：Please refer to **eM3G_SCROLL_DIRECTION** for the scroll direction.
ucEndlessLoop：scroll loop flag:
0: no or 1: yes.
ulSpeed：Scroll speed: pixels/second.

ulLeftPos：Left position of scroll box.
ulRightPos：Right position of scroll box.
ulBackgroundTopPos：Background top position
ulBackgroundLeftPos：Background left position
ulBackgroundWidth：Background width.
ulBackgroundHeight：Background height.

▪ *M3G_TO_TEXT*

```
typedef struct tagM3G_TO_TEXT {
    UCHAR  ucValid;          /* the structure valid flag */
    UCHAR  ucTextLen;        /* Text length: 0~80*/
    CHAR   szTextBuf[MAX_TEXT_BUF_LEN];
} M3G_TO_TEXT;
```

Description

M3G_TO_TEXT data structure defined the text of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TO_TEXT** is described as follows:
ucValid：the structure valid flag, 0: invalid or 1: valid.
ucTextLen：Text length: at most 160 English characters, or at most 80 Chinese characters.
szTextBuf：Text content.

▪ *M3G_TEXTOVERLAY_CONFIG*

```
typedef struct tagM3G_TEXTOVERLAY_CONFIG {
    M3G_TO_CARET_POSITION   CaretPos;     /* Text position information */
    M3G_TO_FONT_INFO        FontInfo;     /* font information*/
    M3G_TO_COLOR            ColorInfo;    /* Foreground/background color informtion*/
    M3G_TO_DISPLAY_MODE     DisplayMode;  /* Display mode*/
    M3G_TO_SCROLL_INFO      ScrollInfo;   /* text scroll information*/
    M3G_TO_TEXT             Text;         /* Text*/
} M3G_TEXTOVERLAY_CONFIG;
```

Description

M3G_TEXTOVERLAY_CONFIG data structure defined the configurations of text overlay. The definitions are in the m3glib.h header file.

Domain Description

Each data domain of **M3G_TEXTOVERLAY_CONFIG** is described as follows:
ucValid：the structure valid flag,
0: invalid or 1: valid.
CaretPos：Please refer to **M3G_TO_CARET_POSITION** for the text position information.
FontInfo：Please refer to **M3G_TO_FONT_INFO** for the font information.
ColorInfo：Please refer to **M3G_TO_COLOR** for the foreground/background color configuration informtion.
DisplayMode：Please refer to **M3G_TO_DISPLAY_MODE** for the display mode.
ScrollInfo：Please refer to **M3G_TO_SCROLL_INFO** for the text scroll information.
Text：Please refer to **M3G_TO_TEXT** for the text information.

2.10. Event Lists

- ***M3GEV_OPEN_CMPLT***

The event shows that **ISX_m3g_OpenEx()** asynchronous operation has been completed. Channel has been successfully opened.

- ***M3GEV_OPEN_FAIL***

The event shows that **ISX_m3g_OpenEx()** asynchronous operation has been completed and failed.

- ***M3GEV_SET_PARM_CMPLT***

The event shows that **ISX_m3g_SetParm()** asynchronous operation has been completed. Media has been successfully changed.

- ***M3GEV_SET_PARM_FAIL***

The event shows that **ISX_m3g_SetParm()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_PARM_CMPLT***

The event shows that **ISX_m3g_GetParm()** asynchronous operation has been completed.

- ***M3GEV_GET_PARM_FAIL***

The event shows that **ISX_m3g_GetParm()** asynchronous operation has been completed and failed.

- ***M3GEV_START_MEDIA_CMPLT***

The event shows that **ISX_m3g_StartMedia()** asynchronous operation has been completed. Media has been successfully started.

- ***M3GEV_START_MEDIA_FAIL***

The event shows that **ISX_m3g_StartMedia()** asynchronous operation has been completed and failed.

- ***M3GEV_STOP_MEDIA_CMPLT***

The event shows that **ISX_m3g_StopMedia()** asynchronous operation has been completed. Media has been successfully stopped.

- ***M3GEV_STOP_MEDIA_FAIL***

The event shows that **ISX_m3g_StopMedia()** asynchronous operation has been completed and failed.

- ***M3GEV_PLAY_BEGIN***

The event shows that **ISX_m3g_Play()** asynchronous operation has been completed. Files playing started.

- ***M3GEV_PLAY_FAIL***

The event shows that **ISX_m3g_Play()** asynchronous operation has been completed and failed.

- ***M3GEV_PLAY_CMPLT***

The event shows that **ISX_m3g_Play()** asynchronous operation has been completed. Files playing completed.

- ***M3GEV_RECORD_BEGIN***

The event shows that **ISX_m3g_Record()** asynchronous operation has been completed. File recording started.

- ***M3GEV_RECORD_FAIL***

The event shows that **ISX_m3g_Record()** asynchronous operation has been completed and failed.

- ***M3GEV_RECORD_CMPLT***

The event shows that **ISX_m3g_Record()** asynchronous operation has been completed. File recording completed.

- ***M3GEV_PAUSE_CMPLT***

The event shows that **ISX_m3g_PauseCh()** has been completed successfully..

- ***M3GEV_PAUSE_FAIL***

The event shows that **ISX_m3g_PauseCh()** asynchronous operation has been completed and failed.

- ***M3GEV_RESUME_CMPLT***

The event shows that **ISX_m3g_ResumeCh()** has been completed successfully

- ***M3GEV_RESUME_FAIL***

The event shows that **ISX_m3g_ResumeCh()** asynchronous operation has been completed unsuccessfully.

- ***M3GEV_SEND_DIGITS_CMPLT***

The event shows that **ISX_m3g_SendDigits()** asynchronous operation has been completed. DIGITS has been successfully sent.

- ***M3GEV_SEND_DIGITS_FAIL***

The event shows that **ISX_m3g_SendDigits()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_DIGITS_CMPLT***

The event shows that **ISX_m3g_GetDigits()** asynchronous operation has been completed. DIGITS has been gathered.

- ***M3GEV_GET_DIGITS_FAIL***

The event shows that **ISX_m3g_GetDigits()** asynchronous operation has been completed and failed.

- ***M3GEV_DIGITS_RECEIVED***

The event is an active event and has only one Digit; the event data is mapped as the **M3G_DIGIT_INFO** structure. On default, the event is prohibited, and use the function **ISX_m3g_SetEvtMsk()** to enable the event.

Note: The function ISX_m3g_GetDigits() can't influence the event.

- ***M3GEV_CST***

The event is an active event and the event data is mapped as the DX_CST structure. On default, the event is prohibited, and use the function **ISX_m3g_SetEvtMsk()** to enable the event. The event CST currently only is DE_DIGITS, so whenever DTMF is detected, an event of TDX_CST will occur. See the following example:

```
case M3GEV_CST:
{
    DX_CST *cstp;
    cstp = (DX_CST *)ISX_sr_getevtdatap();
    int iDev = ISX_sr_getevtdev();
    if(cstp-> cst_event == DE_DIGITS) {
        printf("dev:%d, receive DTMF:%c\n", iDev, cstp->cst_data);
    }
    break;
}
```

Note: The function ISX_m3g_GetDigits() can't influence the event.

- ***M3GEV_START_H245_CMPLT***

The event shows that **ISX_m3g_StartH245()** asynchronous operation has been completed. The H.245 call has been successfully initiated.

- ***M3GEV_START_H245_FAIL***

The event shows that **ISX_m3g_StartH245()** asynchronous operation has been completed and failed.

- ***M3GEV_STOP_H245_CMPLT***

The event shows that **ISX_m3g_StopH245()** asynchronous operation has been completed. The H.245 call has been successfully terminated.

- ***M3GEV_STOP_H245_FAIL***

The event shows that **ISX_m3g_StopH245()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_CALL_STATUS_CMPLT***

The event shows that **ISX_m3g_GetCallStatus()** asynchronous operation has been completed and the TCS status has been successfully retrived.

- ***M3GEV_GET_CALL_STATUS_FAIL***

The event shows that **ISX_m3g_GetCallStatus()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_MSDINFO_CMPLT***

The event shows that **ISX_m3g_GetMsdInfo()** asynchronous operation has been completed and the MSD information has been successfully retrived.

- ***M3GEV_GET_MSDINFO_FAIL***

The event shows that **ISX_m3g_GetMsdInfo()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_TERM_CAPS_CMPLT***

The event shows that **ISX_m3g_GetTermCaps()** asynchronous operation has been completed and terminal configures has been successfully retrived.

- ***M3GEV_GET_TERM_CAPS_FAIL***

The event shows that **ISX_m3g_GetTermCaps()** asynchronous operation has been completed and failed.

- ***M3GEV_SEND_H245_MISC_CMD_CMPLT***

The event shows that **ISX_m3g_SendH245MiscCmd()** asynchronous operation has been completedand a MISC command has been successfully sent.

- ***M3GEV_SEND_H245_MISC_CMD_FAIL***

The event shows that **ISX_m3g_SendH245MiscCmd()** asynchronous operation has been completed and failed.

- ***M3GEV_GET_LC_PARM_CMPLT***

The event shows that **ISX_m3g_GetLCParm()** asynchronous operation has been completed and the LC configures has been successfully retrived.

- ***M3GEV_GET_LC_PARM_FAIL***

The event shows that **ISX_m3g_GetLCParm()** asynchronous operation has been completed and failed.

- ***M3GEV_OPEN_LC_CMPLT***

Not supported temporarily.
The event shows that the the H324M logical channel has been opened successfully.
The logical channel number can be obtained through **ISX_sr_getevtdatap()**.

- ***M3GEV_OPEN_LC_FAIL***

Not supported temporarily.
The event shows that the H324M logical channel open failed.
The logical channel number can be obtained through **ISX_sr_getevtdatap()**.

- ***M3GEV_LCINFO_STATUS***

The event is an active event; when the H324M logical channel status changed, the event will occur.
The status information can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **M3G_LC_INFO** structure.

- ***M3GEV_CLOSE_LC_CMPLT***

The event shows that the H324M logical channel has been successfully closed.
The logical channel number can be obtained through **ISX_sr_getevtdatap()**.

- ***M3GEV_CLOSE_LC_FAIL***

The event shows that the the H324M logical channel unsuccessfully closed.
The logical channel number can be obtained through **ISX_sr_getevtdatap()**.

M3GEV_CALL_STATUS

The event is an active event; when the H324M call status changed, the event will occur. The call status can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **M3G_CALL_STATUS** structure.

The H324 call status will changed by the functions **ISX_m3g_StartH245()** and **ISX_m3g_StopH245()**.

M3GEV_MSDINFO_STATUS

The event is an active event; after the H324M call is initiated, the event is used to report the MSD information. The MSD information data can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **M3G_MSD_INFO** structure.

M3GEV_TCSINFO_STATUS

The event is an active event; after the H324M call is initiated, the event is used to report the TCS negotiate information. The TCS negotiate information data can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **M3G_TCS_INFO** structure.

SYSEV_M3G_STATUS

The event is an active event; after the system is started and initialized, the event is used to report the working status of M3G daughter board. The working status can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **SYS_EVT_DATA** structure. Please refer to the *ISX4000 SRL User Manual* for details.

SYSEV_M3G_BRD_CAP

The event is an active event; after the system is started and initialized, the event is used to report the capacity of M3G daughter board. The capcity information can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **SYS_EVT_DATA** structure. Please refer to the *ISX4000 SRL User Manual* for details.

SYSEV_M3G_CONN_STATUS

The event is an active event; after the system is started and initialized, the event is used to report the connection status between the M3G daughter and M3GC server. The connection status can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **SYS_EVT_DATA** structure. Please refer to the *ISX4000 SRL User Manual* for details.

Example: Processing of M3G daughter board connection status data

```
SYS_EVT_DATA* pEvtData = ISX_sr_getevtdatap();
printf("M3G_BRD<%d.%d> conn_status=%d, m3gcno=%d", pEvtData->cIsxNo, pEvtData->cBrdNo,
        pEvtData->m3gConnSt.conn_status, pEvtData->m3gConnSt.m3gcno);
if(pEvtData->m3gConnSt.conn_status == M3G_BRD_CONN_STATUS_ACTIVE_CONNECTED) {
      printf("M3G board connect ok!");
}
```

M3GEV_DOWNLOADFONT_CMPLT

The event shows that **ISX_m3g_DownloadFon()** asynchronous operation has been successfully completed.

M3GEV_DOWNLOADFONT_FAIL

The event shows that **ISX_m3g_DownloadFon()** asynchronous operation has been completed and failed.

M3GEV_CTRLTOOLBOX_CMPLT

The event shows that **ISX_m3g_CtrlToolbox()** asynchronous operation has been successfully completed.

- ***M3GEV_CTRLTOOLBOX_FAIL***

The event shows that **ISX_m3g_CtrlToolbox()** asynchronous operation has been completed and failed.

- ***M3GEV_SETLOGOPARM_CMPLT***

The event shows that **ISX_m3g_SetLogoParm()** asynchronous operation has been successfully completed.

- ***M3GEV_SETLOGOPARM_FAIL***

The event shows that **ISX_m3g_SetLogoParm()** asynchronous operation has been completed and failed.

- ***M3GEV_DOWNLOADLOGO_CMPLT***

The event shows that **ISX_m3g_DownloadLogo()** asynchronous operation has been successfully completed.

- ***M3GEV_DOWNLOADLOGO_FAIL***

The event shows that **ISX_m3g_DownloadLogo()** asynchronous operation has been completed and failed.

- ***M3GEV_STARTTEXTOVERLAY_CMPLT***

The event shows that **ISX_m3g_StartTextOverlay()** asynchronous operation has been successfully completed.

- ***M3GEV_STARTTEXTOVERLAY_FAIL***

The event shows that **ISX_m3g_StartTextOverlay()** asynchronous operation has been completed and failed.

- ***M3GEV_STOPTEXTOVERLAY_CMPLT***

The event shows that **ISX_m3g_StopTextOverlay()** asynchronous operation has been successfully completed.

- ***M3GEV_STOPTEXTOVERLAY_FAIL***

The event shows that **ISX_m3g_StopTextOverlay()** asynchronous operation has been completed and failed.

- ***M3GEV_SETTEXTOVERLAYPARM_CMPLT***

The event shows that **ISX_m3g_SetTextOverlayParm()** asynchronous operation has been successfully completed.

- ***M3GEV_SETTEXTOVERLAYPARM_FAIL***

The event shows that **ISX_m3g_SetTextOverlayParm()** asynchronous operation has been completed and failed.

- ***M3GEV_GETTEXTOVERLAYPARM_CMPLT***

The event shows that **ISX_m3g_GetTextOverlayParm()** asynchronous operation has been completed. Text overlay parameters have been successfully retrived. The text overlay parameters can be obtained through **ISX_sr_getevtdatap()**, and the data is mapped as the **M3G_TEXTOVERLAY_CONFIG** structure.

- ***M3GEV_GETTEXTOVERLAYPARM_FAIL***

The event shows that **ISX_m3g_GetTextOverlayParm()** asynchronous operation has been completed and

failed.

2.11. Error Code

0x00	------	No Errors
0x01	------	System Error
0x03	------	Function Timed Out
0x04	------	Invalid Entry in the DX_IOTT
0x05	------	Invalid Entry in the DX_TPT
0x09	------	Device is Already Busy
0x16	------	Operation timeout
0x44	------	The state is invalid
0xa0	------	Channel type error