ANALOG 用户手册

版本 2.0

版本变更历史

	日期	变更
VER 2.0.0	2008年 9月 15日	创建本文档
VER 2.0.1	2009年 2月 9日	修正文档的一些错误
VER 2.0.2	2009年 3月 31日	增加拍插簧识别功能
VER 2.0.3	2009年 4月 1日	增加AG_CAP.ca_cnct_t
VER 2.0.4	2009年 12月11日	增加事件MOEV_RNGON和MOEV_RNGOFF
VER 2.0.5	2010年4月16日	增加宏定义CR_SS_TELNUM
VER 2.0.5	2010年4月19日	增加AG_CAP域**ca_1strngbak**的支持
VER 2.0.6	2010年6月25日	增加事件MOEV_CALLERID和MSEV_RINGS 增加函数ISX_ags_SendCallerID
VER 2.0.7	2010年 7月 16日	修正文档的一些错误

欲获取最新产品及相关信息，请访问我们的网站：http://www.ehangcom.com

声明：

本材料受中国和世界其他各国的知识产权和商业机密相关法律保护。除非得到广州市毅航通信技术有限公司（下面有时简称毅航通信）的书面协议、合同或授权，任何复制、传播、或修改的行为（无论在本公司内部和外部）都是非法的。

本产品可能存在设计缺陷或错误，可能导致和该文档中的参数及描述有所偏差。我们一直努力使我们的产品手册在出版的时候更完整和更准确，但由于实际情况的不断变化，本文档只提供毅航通信产品的通用信息。

毅航通信对本文档不提供任何关于开发、销售任何特定功能的产品的承诺。也不提供任何对提到过或暗示过的产品应用的承诺。本文档的内容随时更新并且不做通知。因此，本文档的信息不应被看作是承诺和保证。

毅航通信不对本文档中任何技术或排版的错误、遗漏承担责任，也不对由此造成的任何损失承担责任。

毅航通信的产品并非针对且未被经认证授权可用于医疗、援救或生命维持等应用，以及任何可能会因为毅航通信的产品发生故障而导致人员伤亡的场合。

商标与知识产权

所有产品及服务的名称都是各个特定厂商的商标或者注册商标。本文档中提到的这些特定设备和软件的知识产权受中国和其他国家的相关法律条文的保护。

例如

Ehangcom, iSX, iSX4000, iSX UAP,Ehcomm 等都是毅航通信的注册商标。

Microsoft Windows, Microsoft Windows 98, Microsoft Windows 95, Microsoft NT等都是微软公司的注册商标。

SUN Solaris 是 SUN MicroSystem 公司的注册商标。

目　　录

关于本手册

欢迎阅读本文档，该文档是 ISX4000 ANALOG 用户手册，下面给出了有关本软件的使用目的、阅读对象、文档描述和相关信息。

目的

本手册提供了 iSX 通用业务平台 SDK 中有关 ANALOG 函数使用的相关信息。

阅读对象

1. 发布人员
2. 系统集成商
3. 工具包开发人员
4. 独立软件开发商
5. 系统买卖中间商
6. OEM 开发商

如何使用手册

该手册随软件的安装而产生，本文档主要描述 ISX1000 模拟部分 API，包含以下章节：

1. *接口函数说明：*提供打开、关闭、配置、交换模拟设备等函数使用说明。
2. *数据结构：*提供模拟设备所用到的主要数据结构说明。
3. *事件列表：*提供模拟通道所用到的主要事件说明。

相关信息

本手册相关信息请参考以下文档：

1. *ISX4000 CS 用户手册*
2. *ISX4000 DTI 用户手册*
3. *ISX4000 Global Call 用户手册*
4. *ISX4000 ISX CALL 用户手册*
5. *ISX4000 SRL 用户手册*
6. *ISX4000 SDK 编程指南*
7. *OAM SDK 用户手册*
8. *PRD SDK 用户手册*
9. *ISX4000 通用业务平台软件安装手册*

第一章 接口函数说明

2.1 打开关闭函数

▪ *ISX_ag_open()*

函数名:	int ISX_ag_open(nodeno, brdno, channel, iDspChH, pusrattr)	
输入参数:	**char nodeno**	节点号
	char brdno	子板号
	char channel	通道号
	int iDspChH	与该模拟线通道绑定的 DSP 语音通道设备句柄
	void* pusrattr	用户自定义属性
返回值:	>0 返回设备句柄 -1 失败	
头文件:	srllib.h voclib.h aglib.h	
类别:	模拟线设备管理	
模式:	同步	

描 述

ISX_ag_open( ) 用于打开模拟线通道设备，若打开成功则返回该设备的句柄。模拟线通道设备只能被一个进程打开，即若A进程已经成功打开某个模拟线通道设备，B进程打开该模拟线通道设备时则返回失败，除非A进程关闭了该模拟线通道设备。

该函数返回的设备句柄是由 EHangCom 自己定义的，并不是操作系统标准的文件句柄，所以使用操作系统命令如 **read()**、**write()**或 **ioctl()**来操作该句柄将会产生不可预期的后果。

参数	描述
nodeno	节点号；
brdno	模拟线子板号。
channel	模拟线通道号。
iDspChH	与该模拟线通道绑定的 DSP 语音通道设备句柄，若不绑定，则设为－1。
usrattr	保存用户自定义属性的指针

警 告

- 请不要使用操作系统的**open( )**函数来打开设备，否则将发生不可预期的结果。
- 当应用程序从父进程生成子进程时，子进程不能继承父进程的设备句柄。
- 由于目前模拟子板自身无产生 TONE 的能力，也不具备 DTMF 检测能力，因此需要用 DSP 语音通道来完成这些功能，因此需要把模拟线和 DSP 语音通道进行绑定，若不绑定，则下

列函数就无法工作：
ISX_ago_dial()，ISX_ags_genring()，ISX_ag_StartSendTone()以及 DTMF 检测。

错 误

如果该函数返回－1 说明函数调用失败，失败的原因请用 ISX_sr_getlasterr()来获得。

例 子

这个例子说明如何打开一个模拟线通道设备。

```
#include "srllib.h"
#include "voclib.h"
#include "aglib.h"
main()
{
   int agdev;       /* channel handle */
   int dspchdev;
   dspchdev = ISX_dx_open(DT_DSP_CH, 0, 0, 0);
   ...
   /* Open Channel */
   if ((agdev = ISX_ag_open(0,0,0, dspchdev, NULL)) == -1) {
     /* process error */
   }
   ...
}
```

相关信息

- **ISX_ag_close()**
- **ISX_sr_getlasterr()**

ISX_ag_close()

函数名:	int ISX_ag_close(dev, oflags)	
输入参数:	**int dev**	有效的 模拟线通道句柄
	int ofalgs	预留给将来使用
返回值:	0 成功 -1 失败	
头文件:	srllib.h voclib.h aglib.h	
类别:	模拟线设备管理	
模式:	同步	

描 述

该函数关闭一个模拟线通道设备 (用**ISX_ag_open()**函数打开返回的)。该函数不改变设备的状态（如交换状态），它只是释放设备句柄和断开调用进程与设备之间的联系。
注: ISX_ag_close( ) 执行成功后禁止所有与该设备有关的事件发生。

参数	描述
dev	指定一个有效的设备句柄(**ISX_ag_open()**返回的)

oflags	为将来预留的，目前无意义

警 告

- 一旦关闭设备，打开该设备的进程不能再使用该句柄来对设备进行操作。
- 请不要用操作系统的**close( )**函数关闭设备句柄，否则将发生不可预期的结果。
- **ISX_ag_close( )**函数清空被关闭的设备事件缓冲区。

错 误

如果该函数返回－1 说明函数调用失败，失败的原因请用 ISX_sr_getlasterr()来获得。

例 子

这个例子说明如何关闭一个模拟线通道设备。

```
#include <srllib.h>
#include <voclib.h>
main()
{
   int agdev;         /* channel descriptor */
   ...
   /* Open Channel */
   if ((agdev = ISX_ag_open(0,0,0, dspchdev, NULL)) == -1) {
     /* process error */
   }
   ...
   /* Close channel */
   if (ISX_ag_close(agdev,0) == -1) {
     /* process error */
   }
}
```

相关信息

- **ISX_ag_open()**
- **ISX_sr_getlasterr()**

2.2 配置函数

- ***ISX_ags_RingParam()***

函数名称：	int ISX_ags_RingParam(ucIsxNo, usRingOnDuration, usRingOffDuration)	
输入参数：	**UCHAR ucIsxNo**	节点号
	USHORT usRingOnDuration	振铃时长，单位为毫秒
	USHORT usRingOffDuration	两次铃声间隔时长，单位为毫秒
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	配置	
模式：	同步	

描 述

ISX_ags_RingParam( ) 配置振铃参数，该函数仅适用于拥有模拟线的节点。

参数	描述
ucIsxNo	节点号。
usRingOnDuration	振铃时长，单位为毫秒，不能为 0。
usRingOffDuration	两次铃声间隔时长，单位为毫秒，不能为 0。

警 告

无。

错 误

如果函数调用失败，则返回－1

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   USHORT usRND = 1500;
   USHORT usRFD = 4000;
   if (ISX_ag_RingParam(0, usRND, usRFD) == -1) {
      /* process error */
   }
   . . .
}
```

相关信息

无。

▪ *ISX_ags_SetHookParam()*

函数名称：	int ISX_ags_SetHookParam(ucIsxNo, usHookFlashTime)	
输入参数：	**UCHAR ucIsxNo**	节点号
	USHORT usHookFlashTime	检测拍插簧时长
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	配置	
模式：	同步	

描 述

ISX_ags_SetHookParam( ) 配置拍插簧识别的时间区间，即在定义的时长之内属于拍插簧事件，超过这个时长则属于挂机事件。该函数仅适用于拥有模拟线的节点。

参数	描述
ucIsxNo	节点号。
usHookFlashTime	检测拍插簧时长，单位为 10 毫秒，不能为 0； 默认为 50 个时间单位，即 500 毫秒。

警 告

无。

错 误

如果函数调用失败，则返回－1

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   if (ISX_ags_SetHookParam(0, 80) == -1) {
      /* process error */
   }
   . . .
}
```

相关信息

无。

▪ *ISX_ag_SetEvtMsk()*

函数名称：	int ISX_ag_SetEvtMsk(dev, mask, action)	
输入参数：	**int dev**	有效的模拟线通道设备句柄
	unsigned long mask	事件 bitmask

	int action	增减标志
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h	
类别：	配置	
模式：	同步	

描 述

ISX_ag_SetEvtMsk( ) 设置通道事件掩码，如果某个事件的掩码被清除，则该事件就不会送给应用程序，事件的掩码默认值参见表 1。

参数	描述
dev	模拟线设备句柄
mask	事件掩码，参见表 1
action	增加或清除标志： AGACT_SETMSK：允许接收指定的事件，而禁止没有指定的事件。 AGACT_ADDMSK：允许接收指定的事件。 AGACT_SUBMSK：不允许接收指定的事件。

表 1. 参数掩码

类型	描述	默认值
MSMM_OFFHOOK	设置是否接收**MSEV_SIGEVT**的摘机 事件	允许
MSMM_ONHOOK	设置是否接收**MSEV_SIGEVT**的挂机 事件	允许
MSMM_HOOKFLASH	设置是否接收**MSEV_SIGEVT**的拍插簧 事件	禁止
AGMM_DIGITS	设置是否接收**AGEV_DIGITS**事件	禁止
MOMM_RINGS	设置是否接收**MOEV_RINGS**事件	禁止
MOMM_CALLERID	设置是否接收**MOEV_CALLERID**事件	禁止
MSMM_RINGS	设置是否接收**MSEV_RINGS**事件	禁止

警 告

通过该函数设置**AGEV_DIGITS**不会影响绑定的DSP通道使用**ISX_dx_getdig()**函数。

错 误

如果函数调用失败，则返回－1；可用**ISX_ATDV_LASTERR()**取得错误原因，可能的原因如下：

EDX_BADPARM：无效的参数

EDX_BUSY：设备忙

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   int agdev;
   if ((agdev = ISX_ag_open(0, 0, 0, dspchdev, NULL)) == -1) {
      /* process error */
   }
    /* Set event mask to collect digits */
   if (ISX_ag_SetEvtMsk(agdev,AGMM_DIGITS) == -1) {
     /* error setting event */
   }
}
```

相关信息

无。

2.3 扩展属性函数

▪ *ISX_ATAG_TERMMSK()*

函数名称：	int ISX_ATAG_TERMMSK(dev,OperType)	
输入参数：	**int dev**	模拟线通道设备句柄
	int OperType	操作类型
返回值：	最后一次中止原因（位掩码） AT_FAILURE 错误	
头文件：	srllib.h voclib.h aglib.h	
类别：	扩展属性	
模式：	同步	

描 述

ISX_ATAG_TERMMSK( ) 函数返回包含最后一次I/O操作中止原因的一个整数。

参数	描述
dev	模拟线通道设备句柄。
OperType	操作类型，参考 OPERATIONS

中止原因如下：

ISX_ags_genring()可能出现的中止原因：
MSMM_RNGOFFHK：对 FXS 口进行振铃过程中遇到摘机而中止振铃。
MSMM_RNGSTOP：因调用 ISX_ags_stopfn()而中止振铃。
MSMM_TERM：振铃正常中止。

ISX_ago_dial()可能出现的中止原因：
CR_SS_TELNUM：发送电话号码成功而中止呼出。
CR_SF_TELNUM：因为发送电话号码失败而中止呼出操作。
CR_BUSY：线路忙
CR_NOANS：无应答
CR_NORB：无回铃声
CR_CNCT：应答
CR_NODIALTONE：无拨号音
CR_FAXTONE：传真

ISX_ag_StartSendTone()可能出现的中止原因：
TM_DIGIT：收到了指定的 digit
TM_EOD：数据完（如播放时到了文件，如录音时接收到指定长度的语音数据）
TM_ERROR：I/O 设备错误
TM_IDDTIME：按键间超时
TM_MAXDTMF：收到指定个数的 DTMF
TM_MAXTIME：超过了函数指定的操作时间
TM_USRSTOP：用户停止了操作
TM_BARGEIN：因检测到 VAD 而中断播放

警 告

- 如果多个中止条件同时满足，则原因就有多个位被同时置上

错 误

如果指定了无效的设备句柄，则函数返回 AT_FAILURE 。

例 子

```
#include <srllib.h>
#include <voclib.h>
main()
{
   int agsdev;
   agsdev = ISX_ag_open(0, 0, 0, dspchdev, NULL);
   if (ISX_ags_genring(agsdev, 30, EV_ASYNC, 0) == -1) {
     /* process error */
   }
   ……
   /* Examine bitmap to determine if offhook caused termination */
   if((term = ISX_ATDX_TERMMSK(chdev)) == AT_FAILURE) {
     /* Process error */
   }
   if(term & MSMM_RNGOFFHK) {
     printf("Terminated on the line offhook\n");
      ...
   }
}
```

相关信息

- DV_TPT

ISX_ag_GetUsrAttr()

函数名称：	int ISX_ag_GetUsrAttr(dev, usr_attrp)	
输入参数：	**int dev**	有效的模拟线通道句柄
	void usr_attrp**	用来保存用户属性地址的指针
返回值：	0 成功 <0 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	扩展属性	
模式：	同步	

描 述

ISX_ag_GetUsrAttr( ) 函数获取用户自定义的属性（先前通过ISX_ag_open()和ISX_ag_SetUsrAttr()设置的用户属性）。

参数	描述
dev	有效的模拟线通道句柄
usr_attrp	获取用户属性地址的指针

警 告

无。

错 误

如果函数调用失败则返回值小于 0，失败原因可通过调用**ISX_ATDV_LASTERR( )**函数获得。

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
#define MAXCH   30        /* max. number of channels in system */
/*
 * Data structure which stores all information for each ag channel
 */
struct chbag {
   int     dev;                /* ag channel handle */
   UCHAR   ucIsxNo;            /* ISX node no. */
   UCHAR   ucBrdNo;            /* ag Board no. */
   UCHAR   ucCh;               /* ag channel no. */
} agport[MAXCH+1];

int OpenAGCh()
{
  for(int j=0; j<MAXCH; j++){
     agport[j].ucIsxNo = 0;
     agport[j].ucBrdNo = 0;
     agport[j].ucCh = j;
     agport[j].dev = ISX_ag_open(0, 0, j, -1, (void*)&agport[j]);
     if(agport[j].dev == -1){
        printf("open ag channel fail.reason:%x\n", ISX_sr_getlasterr());
        return(-1);
     }
  }
}

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

    if ( ISX_ag_GetUsrAttr( dev, (void**)&vattrp) == -1 ) {
       printf ("ISX_ag_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->ucCh);
    return (0);
}
```

相关信息

- **ISX_ag_SetUsrAttr()**
- **ISX_ag_open()**

ISX_ag_SetUsrAttr()

函数名称：	int ISX_ag_SetUsrAttr(dev, usrattr)	
输入参数：	**int dev**	有效的模拟线通道句柄

	void* usrattr	用户属性
返回值：	0 成功 <0 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	扩展属性	
模式：	同步	

描 述

ISX_ag_SetUsrAttr( ) 设置通道的属性（用户自定义）。用户设置的用户属性可以通过**ISX_ag_GetUsrAttr()**函数获得该属性。

参数	描述
dev	有效的模拟线通道句柄
usrattr	用户自定义属性，应用程序可以通过**ISX_ag_GetUsrAttr()**函数获得该属性。

警 告

无。

错 误

如果函数调用失败则返回值小于 0，失败原因可通过调用**ISX_ATDV_LASTERR( )**函数获得。

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
#define MAXCH   30              /* max. number of channels in system */
/*
 * Data structure which stores all information for ag channel
 */
struct chbag {
   int     dev;                 /* ag channel handle */
   UCHAR    ucIsxNo;            /* ISX node no. */
   UCHAR    ucBrdNo;            /* ag Board no. */
   UCHAR    ucCh;               /* ag channel no. */
} agport[MAXCH+1];
int OpenDspCh()
{
  for(int j=0; j<MAXCH; j++){
     agport[j].ucIsxNo = 0;
     agport[j].ucBrdNo = 0;
     agport[j].ucCh = j;
     agport[j].dev = ISX_ag_open(0, 0, j, -1, NULL);
     if(agport[j].dev == -1){
        printf("open ag channel fail.reason:%x\n", ISX_sr_getlasterr());
        return(-1);
     }
     ISX_dx_SetUsrAttr(agport[j].dev, (void*)&agport[j]);
  }
}
```

相关信息

- **ISX_ag_SetUsrAttr()**

- ISX_ag_open()

ISX_ag_getbrdtype()

函数名称：	int ISX_ag_getbrdtype(ucIsxNo, ucBrdNo)	
输入参数：	**UCHAR ucIsxNo**	节点号
	UCHAR ucBrdNo	模拟子板号
返回值：	-1：函数调用失败，请检查参数。 AG_BRDTYPE_FXO：FXO 子板，该子板上所有通道都是 FXO 口。 AG_BRDTYPE_FXS：FXS 子板，该子板上所有通道都是 FXS 口。 AG_BRDTYPE_NULL：目前还未知该子板类型，请稍后重新查询。	
头文件：	srllib.h voclib.h aglib.h	
类别：	扩展属性	
模式：	同步	

描 述

ISX_ag_getbrdtype( )查询指定模拟子板的板类型。

参数	描述
ucIsxNo	节点号。
ucBrdNo	模拟子板号。

警 告

无。

错 误

如果指定了无效的参数，则函数返回－1。

例 子

```
#include <srllib.h>
#include <voclib.h>
main()
{
   ISX_sr_waitevt(-1);
   int iEvtType = ISX_sr_getevttype();
   if(iEvtType == SYSEV_AG_STATUS){
      SYS_EVT_DATA* pEvtData = (SYS_EVT_DATA*)ISX_sr_getevtdatap();
      int iBrdType = ISX_ag_getbrdtype(pEvtData->cIsxNo, pEvtData->cBrdNo);
      if(iBrdType >= 0) printf("AG BRD[%d:%d] is %s\n",
                              iBrdType==AG_BRDTYPE_FXO ? "FXO":"FXS");
   }
}
```

相关信息

无。

2.4 交换函数

▪ *ISX_ag_listen()*

函数名:	int ISX_ag_listen(dev, sc_tsinfop)	
输入参数:	**int dev**	模拟线设备句柄
	SC_TSINFO* sc_tsinfop	指向交换矩阵时隙信息结构
返回值:	0 成功 -1 失败	
头文件:	voclib.h aglib.h	
类别:	时隙交换	
模式:	同步	

描 述

该函数把模拟线通道设备的接收端(接收时隙)与一个时隙(另一个设备的发送端)建立交换，从而使该模拟线通道接收来自这个时隙的数据，这个函数实现了通道的半交换。交换双方的设备必须位于同个节点内，跨节点的设备要交换须用到光口中继通道，详细请参见有关文档说明。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
sc_tsinfop	指定一个指向**SC_TSINFO**结构的指针

虽然多个语音通道设备接收端可以同时连接到同一个发送时隙上，但一个语音通道设备接收端只能连接到一个发送时隙上；而不能同时连接多个发送时隙。

警 告

- 如果指定的设备句柄无效，或指定的时隙无效，函数将返回失败。
- 强烈建议使用**ISX_nr_scroute()**来实现交换，因为**ISX_nr_scroute()**的效率比xx_listen()高。

错 误

如果函数调用失败则返回－1，请使用 SRL 标准属性函数 **ISX_ATDV_LASTERR( )**获得错误代码，或使用 **ISX_ATDV_ERRMSGP( )**获得错误描述。可能返回以下的错误代码：

EDX_BADPARM：参数错误
EDX_TIMEOUT：函数执行超时
EDX_BUSY：设备忙
EDX_OPERTIMEOUT：操作超时，在规定的时间内没有收到设备回应
EDX_SYSTEM：系统错误(例如与 MasterController 网络连接断开)

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   int       dev;              /* ag channel device handle */
   SC_TSINFO sc_tsinfo;        /* Time slot information structure */
   long      scts;             /* Switch matrix time slot */
```

```
        if ((dev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
           printf("Can't open ag channel.  errno = %d", ISX_sr_getlasterr());
           exit(1);
        }
        /* Fill in the switch matrix time slot information */
        sc_tsinfo.sc_numts = 1;
        sc_tsinfo.sc_tsarrayp = &scts;
        /* Get switch matrix time slot connected to transmit of dti channel, suppose
         the dti channel was already opened and it's device handle is dtidev*/
       if (ISX_dt_getxmitslot(dtidev, &sc_tsinfo) == -1) {
           printf("Error message = %s", ISX_ATDV_ERRMSGP(dtidev));
           exit(1);
        }
        /* Connect the receive of ag channel to switch matrix time slot of dti channel
         */
        if (ISX_ag_listen(dev, &sc_tsinfo) == -1) {
           printf("Error message = %s", ISX_ATDV_ERRMSGP(dev));
           exit(1);
        }
     }
```

相关信息

- **ISX_dt_getxmitslot( )**
- **ISX_ag_unlisten()**

ISX_ag_unlisten()

函数名称:	int ISX_ag_unlisten(dev)	
输入参数:	**int dev**	模拟线通道句柄
返回值:	0 成功 -1 失败	
头文件:	srllib.h voclib.h aglib.h	
类别:	时隙交换	
模式:	同步	

描 述

该函数解除模拟线通道接收端与其相连的发送时隙之间的交换。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄

警 告

- 如果指定的设备句柄无效，或指定的时隙无效，函数将返回失败。
- 强烈建议使用**ISX_nr_scunroute()**来解除交换，因为**ISX_nr_scunroute()**效率比xx_unlisten()高。

错 误

如果函数调用失败则返回－1，请使用 SRL 标准属性函数 **ISX_ATDV_LASTERR( )**获得错误代码，或使用 **ISX_ATDV_ERRMSGP( )**获得错误描述。可能返回以下的错误代码：

EDX_BADPARM：参数错误
EDX_TIMEOUT：函数执行超时
EDX_BUSY：设备忙
EDX_OPERTIMEOUT：操作超时，在规定的时间内没有收到设备回应
EDX_SYSTEM：系统错误(例如与 MasterController 网络连接断开)

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   int dev;                /* ag channel device handle */
   if ((dev = ISX_dx_open(0, 0, 0, -1, NULL)) == -1) {
      printf("Can't open channel 0.  errno = %d", ISX_sr_getlasterr());
      exit(1);
   }
   /* Disconnect receive of ag channel from Switch Matrix time slots */
   if(ISX_ag_unlisten(dev) == -1) {
      printf("Error message = %s", ISX_ATDV_ERRMSGP(chdev));
      exit(1);
   }
}
```

相关信息

- ISX_ag_listen()

ISX_ag_getxmitslot()

函数名称：	int ISX_ag_getxmitslot(dev, sc_tsinfop)	
输入参数：	**int dev**	有效的模拟线通道设备句柄
	SC_TSINFO *sc_tsinfop	指向时隙信息的结构指针
返回值：	0 成功 -1 失败	
头文件：	voclib.h aglib.h	
类别	时隙交换	
模式：	同步	

描 述

ISX_ag_getxmitslot( )函数返回指定模拟线通道发送端的时隙。时隙信息保存在SC_TSINFO结构里。

参数	描述
dev	有效的模拟线通道设备句柄
sc_tsinfop	SC_TSINFO结构指针，用来获取时隙信息。

警 告

- 如果指定的通道句柄无效，则函数调用将失败

错 误

如果函数调用失败则返回－1，请使用 SRL 标准属性函数 **ISX_ATDV_LASTERR( )**获得错误代码，或

使用 **ISX_ATDV_ERRMSGP( )**获得错误描述。可能返回以下的错误代码：

EDX_BADPARM：参数错误

EDX_BUSY：设备忙

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
   int      dev;          /* Channel device handle */
   SC_TSINFO sc_tsinfo;    /* Time slot information structure */
   long     scts;          /* Switch Matrix time slot */
   if ((dev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
      printf("Cannot open channel.  errno = %d", ISX_sr_getlasterr());
      exit(1);
   }
   /* Fill in the Switch Matrix time slot information */
   sc_tsinfo.sc_numts = 1;
   sc_tsinfo.sc_tsarrayp = &scts;
   /* Get Switch Matrix time slot connected to transmit of ag channel */
   if (ISX_ag_getxmitslot(dev, &sc_tsinfo) == -1) {
       printf("Error message = %s", ISX_ATDV_ERRMSGP(chdev));
       exit(1);
   }
   printf("The ag channel transmitting on Switch Matrix time slot %d", scts);
   return(0);
}
```

相关信息

- **ISX_ag_listen()**
- **ISX_ag_listen( )**

2.5 FXO 操作函数

▪ *ISX_ago_sethook()*

函数名称：	int ISX_ago_sethook(dev, HookState, mode, ulOperIndex)	
输入参数：	**int dev**	模拟线(FXO 口)通道设备句柄
	int HookState	摘挂机标志
	unsigned short mode	同步异步模式
	unsigned long ulOperIndex	操作编号
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXO 操作函数	
模式：	同步、异步	

描 述

该函数对指定的模拟线通道进行摘机或挂机操作，该函数只适用 FXO 口线路。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
HookState	摘机挂机标志，可以为： AG_OFFHOOK：摘机 AG_ONHOOK：挂机
mode	同步/异步模式： EV_SYNC ：同步 EV_ASYNC ：异步
ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**MOEV_SETHOOK**或**AGEV_ERROR**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  int chdev;
  /* open a channel with chdev as descriptor */
```

```
        if ((chdev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
           /* process error */
        }
        /* put the channel on-hook */
        if (ISX_ago_sethook(chdev,AG_ONHOOK, EV_SYNC, 0) == -1) {
           /* error setting hook state */
        }
        ...
      }
```

相关信息

- ISX_ag_open()

▪ *ISX_ago_dial()*

函数名称：	int ISX_ago_dial(dev, dialstrp, capp, iPrdDev, mode, ulOperIndex)	
输入参数：	**int dev**	模拟线(FXO 口)通道设备句柄
	const char *dialstrp	欲拨的号码
	const AG_CAP *capp	call progress analysis 参数结构
	int iPrdDev	PRD 设备句柄
	unsigned short mode	呼叫模式，同步异步模式
	ULONG ulOperIndex	操作编号
返回值：	异步模式：0 成功，-1 失败 同步模式： －1：失败 CR_SF_TELNUM：发送号码失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXO 操作函数	
模式：	同步、异步	

描 述

ISX_ago_dial( )对指定的模拟线启动一个呼叫，该函数只适用FXO口的模拟线路。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
dialstrp	拨号的号码，以 0 结尾的字符串。号码可以为’0’～’9’，’a’，’b’，’c’，’d’，’*’，’#’，’,’字符；其中’,’为间隔符，每个’,’的表示的间隔为 2.5 秒。最大号码长度不能超过 MAX_AGTELNUM_LEN。
capp	call progress analysis参数结构，参见**AG_CAP**结构说明。若为NULL，则采用系统默认参数。

iPrdDev	PRD 设备句柄；由于目前模拟子板无 DTMF 产生能力，该函数需要进行的拨号由 DSP 语音通道进行播放 DTMF 语音文件来完成，因此在此必须设置 DTMF 文件所在 PRD 的设备句柄。
mode	呼叫模式，同步/异步模式： AG_CALLP：允许call progress analysis，只有该模式才进行拨号音、忙音等信号音频测 EV_SYNC ：同步 EV_ASYNC ：异步
ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**MOEV_CALLP**或**MOEV_DIAL**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

警 告

- 调用该函数前，必须先确定通道处于摘机状态，否则函数调用会失败。
- 由于目前模拟子板无产生DTMF能力，需要DSP通道来进行协助拨号，因此必须对模拟线路＆DSP通道进行绑定才能使用该函数；与DSP通道进行绑定参见**ISX_ag_open()**说明。
- 若在异步模式下启动该函数，若指定了AG_CALLP，操作结束时会产生**MOEV_CALLP**事件；若不指定AG_CALLP，操作结束时会产生**MOEV_DIAL**事件。中止原因可通过**ISX_ATAG_TERMMSK()**函数获得。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  int agdev, dspchdev, iPrdDev;
  int car;
  char* dialstrg;
  dialstrg = "1234";

  dspchdev = ISX_dx_open(DT_DSP_CH, 0, 0, 0);
  iPrdDev = ISX_dx_open(DT_PRD, 0, 0, 0);
  agdev = ISX_ag_open(0, 0, 0, dspchdev, NULL);

  //Set off Hook
  ISX_ago_sethook(agdev, AG_OFFHOOK, EV_SYNC);


  //Dial
  if((car = ISX_ago_dial(agdev, dialstrg, NULL, iPrdDev, EV_SYNC))==-1){
         /* handle error */
  }
  switch( car ) {
  case CR_SF_TELNUM:
         printf("Dial '%s' fail.\n", dialstrg);
         break;
  default:
         break;
```

```
        }

        .....

        //Set on Hook
        if ((ISX_ago_sethook(agdev, AG_ONHOOK, EV_SYNC )) == -1) {
                /* handle error */
        }
        .....
      }
```

相关信息

- **ISX_ag_open()**
- ISX_ago_sethook()

ISX_ago_wtring()

输入参数：	int ISX_ago_wtring(dev, numRings, HookState, timeout, mode, ulOperIndex)	
	int dev	模拟线(FXO 口)通道设备句柄
	int numRings	等待铃声次数
	int HookState	摘挂机标志
	int timeout	超时时间
	unsigned short mode	同步异步模式
	ULONG ulOperIndex	操作编号
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXO 操作函数	
模式：	同步、异步	

描 述

ISX_ago_wtring( )函数等待指定的铃声次数，若检测到指定的铃声次数，则摘机或维持挂机状态（由HookState参数决定）。该函数只适用FXO口的模拟线路。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
numRings	指定等待的铃声次数
HookState	摘机挂机标志，若为 AG_OFFHOOK，在收到 numRings 次铃声后自动执行摘机；若为 AG_ONHOOK，在收到 numRings 次铃声后则不执行摘机，维持在非摘机状态。
timeout	超时时长，单位为毫秒，若为－1 则为无限等待。仅当为同步模式下有意义。

mode	同步/异步模式： EV_SYNC ： 同步 EV_ASYNC ： 异步
ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**MOEV_WTRING**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

警 告

- 若两次铃声间隔超过 MAX_RING_GAP，则认为是新的呼叫，因此需要重新计数铃声的次数。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  int chdev; /* channel descriptor */
  ......
  /* Open Channel */
  if ((chdev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
        /* process error */
  }
  /* Wait for two rings on this channel - no timeout */
  if (ISX_ag_wtring(chdev, 2, AG_OFFHOOK, -1, EV_SYNC, 0) == -1) {
        /* process error */
  }
  ......
}
```

相关信息

- **ISX_ag_open()**

ISX_ago_gtcallid()

输入参数：	int ISX_ago_gtcallid(dev, bufp)	
	int dev	模拟线(FXO 口)通道设备句柄
	char *bufp	保存主叫号码的缓冲区
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXO 操作函数	
模式：	同步	

描 述

ISX_ago_gtcallid( )函数获得主叫号码信息，主叫号码一般在第一次铃声的间隙发送过来。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
bufp	保存主叫号码信息，主叫号码信息最大长度为 MAX_CALLID_DATA_LEN。

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  int chdev; /* channel descriptor */
  ......
  /* Open Channel */
  if ((chdev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
         /* process error */
  }
  char CallerID[MAX_CALLID_DATA_LEN];
  if (ISX_ago_gtcallid(chdev,CallerID) == -1) {
         /* process error */
  }
  ......
}
```

相关信息

- **ISX_ag_open()**

ISX_ago_gtextcallid()

输入参数：	int ISX_ago_gtextcallid(dev, bufp)	
	int dev	模拟线(FXO 口)通道设备句柄
	int infotype	CALLER ID 信息类型
	char *bufp	保存主叫号码的缓冲区
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXO 操作函数	
模式：	同步	

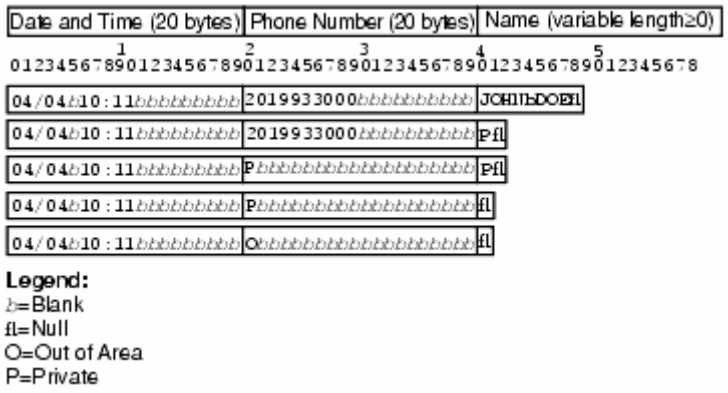
描 述

ISX_ago_gtextcallid( )函数获得CALLER ID信息，CALLER ID一般在第一次铃声的间隙发送过来。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
infotype	CALLER ID 信息类型 CLIDINFO_CMPLT ： 完整的 CALLERID 信息（包含信息头和长度），最大 240 个字节。 CLIDINFO_GENERAL： 普通 CALLERID 信息（仅包含日期时间、主叫号码、用户名） CLIDINFO_CALLID：主叫号码 CLIDINFO_FRAMETYPE：帧类型方式；占不支持
bufp	保存 CALLER ID 信息，信息最大长度为 MAX_CALLID_DATA_LEN。

特别说明：

当 infotype=CLIDINFO_GENERAL 时，返回的 bufp 格式如下图所示：



警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  int chdev; /* channel descriptor */
  ......
  /* Open Channel */
  if ((chdev = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
        /* process error */
  }
  char CallerID[MAX_CALLID_DATA_LEN];
  if (ISX_ago_gtextcallid(chdev, CLIDINFO_GENERAL, CallerID) == -1) {
        /* process error */
  }
  ......
```

```
}
```

相关信息

- **ISX_ag_open()**

2.6 FXS 操作函数

▪ *ISX_ags_genring()*

函数名称：	int ISX_ags_genring(dev, len, mode, ulOperIndex)	
输入参数：	**int dev**	模拟线(FXS 口)通道设备句柄
	unsigned short len	振铃次数
	unsigned short mode	同步异步模式
	ULONG ulOperIndex	操作编号
返回值：	异步模式：0 成功，-1 失败 同步模式： -1：失败 MSMM_RNGOFFHK：遇到摘机而中止振铃 MSMM_RNGSTOP：因调用 ISX_ags_stopfn()函数而中止振铃 MSMM_TERM：振铃次数已到而中止	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXS 操作函数	
模式：	同步、异步	

描 述

ISX_ags_genring( )函数对指定的模拟线路进行振铃，该函数只适用于FXS口的模拟线路。在振铃过程中，若遇到摘机则中止振铃；当振铃次数达到len参数指定的次数后，振铃也会自动中止。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
len	指定振铃的次数，若为 0xFFFF，则不限定振铃次数，直到遇到摘机或者调用 ISX_ags_stopfn()函数才中止。
mode	同步/异步模式： EV_SYNC ：同步 EV_ASYNC ：异步
ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**MSEV_RING**或者**MSEV_NORING**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include "srllib.h"
#include "voclib.h"
```

```
#include "aglib.h"
int dev1;
int rc; /* Return code */

if ((dev1 = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
  printf( "Cannot open the channel");
  exit(1);
}
/*
* Continue processing
*/
/* Generate ringing for 10 cycles in sync mode*/
if ((rc =ISX_ags_genring(dev1,10,EV_SYNC)) == -1) {
  /* process error */
}
/* If timeout, process the condition */
if (rc=MSMM_TERM) {
  printf("Station not responding");
}
/*
* Continue Processing
*/
/* Done processing - close device */
if (ISX_ag_close(dev1) == -1) {
  printf("Cannot close device");
  exit(1);
}
```

相关信息

- **ISX_ags_stopfn()**

ISX_ags_SendCallerID()

函数名称：	int ISX_ags_SendCallerID(dev, OrigAddr, rfu)	
输入参数：	**int dev**	模拟线(FXS 口)通道设备句柄
	char* OrigAddr	主叫号码
	void* rfu	保留，目前无意义。
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXS 操作函数	
模式：	同步	

描 述

ISX_ags_SendCallerID( )函数对指定的模拟线路在振铃的过程中通过FSK方式发送主叫号码信息。该函数仅和ISX_ags_genring()配合使用，仅当收到第一个**MSEV_RINGS**事件时，调用该函数发送主叫号码。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄

OrigAddr	主叫号码，ASCII 字符串，最大长度为 MAX_CALLID_DATA_LEN。
rfu	保留，目前无意义。

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include "srllib.h"
#include "voclib.h"
#include "aglib.h"
int dev1;
int rc; /* Return code */

if ((dev1 = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
  printf( "Cannot open the channel");
  exit(1);
}
ISX_ag_SetEvtMsk(dev1, MSMM_RINGS, AGACT_ADDMSK);
/*
* Continue processing
*/
/* Generate ringing for 10 cycles in async mode*/
if ((rc =ISX_ags_genring(dev1,10,EV_ASYNC)) == -1) {
  /* process error */
}

/*
* Continue processing
*/

/* Event process */
int nAgHdl = ISX_sr_getevtdev();
case MSEV_RINGS:
{
  int nRingNum = *(int *)ISX_sr_getevtdatap();
  if(nRingNum == 1){
        printf("recv first ring");
        ISX_ags_SendCallerID(nAgHdl, “13632366794”);
  }
  break;
}
```

相关信息

- **ISX_ags_stopfn()**
- **ISX_ags_genring()**

ISX_ags_genringCallerID()

函数名称：	int ISX_ags_genringCallerID(dev, len, mode, Cadid, OrigAddr, rfu, ulOperIndex)

输入参数：	**int dev**	模拟线(FXS 口)通道设备句柄
	unsigned short len	振铃次数
	unsigned short mode	同步异步模式
	unsigned short Cadid	铃声节奏，保留，目前无意义。
	char* OrigAddr	主叫号码
	void* rfu	保留，目前无意义。
	ULONG ulOperIndex	操作编号
返回值：	异步模式：0 成功，-1 失败 同步模式： -1：失败 MSMM_RNGOFFHK：遇到摘机而中止振铃 MSMM_RNGSTOP：因调用 ISX_ags_stopfn()函数而中止振铃 MSMM_TERM：振铃次数已到而中止	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXS 操作函数	
模式：	同步、异步	

描 述

ISX_ags_genringCallerID( )函数对指定的模拟线路进行振铃，该函数只适用于FXS口的模拟线路。在振铃过程中，若遇到摘机则中止振铃；当振铃次数达到len参数指定的次数后，振铃也会自动中止。在振铃的过程中通过FSK方式发送主叫号码信息。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
len	指定振铃的次数，若为 0xFFFF，则不限定振铃次数，直到遇到摘机或者调用 ISX_ags_stopfn()函数才中止。
mode	同步/异步模式： EV_SYNC ：同步 EV_ASYNC ：异步
Cadid	铃声节奏，保留，目前无意义。
OrigAddr	主叫号码，ASCII 字符串，最大长度为 MAX_CALLID_DATA_LEN。
rfu	保留，目前无意义。
ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**MSEV_RING**或者**MSEV_NORING**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include "srllib.h"
#include "voclib.h"
#include "aglib.h"
int dev1;
int rc; /* Return code */

if ((dev1 = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
  printf( "Cannot open the channel");
  exit(1);
}
/*
* Continue processing
*/
/* Generate ringing for 10 cycles in sync mode*/
if ((rc =ISX_ags_genringCallerID(dev1,10,EV_SYNC,0,”13302292010”)) ==
-1) {
  /* process error */
}
/* If timeout, process the condition */
if (rc=MSMM_TERM) {
  printf("Station not responding");
}
/*
* Continue Processing
*/
/* Done processing - close device */
if (ISX_ag_close(dev1) == -1) {
  printf("Cannot close device");
  exit(1);
}
```

相关信息

- **ISX_ags_stopfn()**

ISX_ags_stopfn()

函数名称：	int ISX_ags_stopfn(dev, funcid, mode)	
输入参数：	**int dev**	模拟线(FXS 口)通道设备句柄
	unsigned int funcid	要停止的函数 ID 号
	unsigned short mode	同步异步模式
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	FXS 操作函数	

模式：	同步、异步

描 述

ISX_ags_stopfn( )函数停止异步函数操作。

参数	描述
dev	由**ISX_ag_open( )**返回的模拟线通道设备句柄
funcid	要停止的函数 ID 号，目前仅仅支持停止振铃，即：MTF_RING
mode	同步/异步模式： EV_SYNC ： 同步 EV_ASYNC ： 异步

警 告

无。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数

例 子

```
#include "srllib.h"
#include "voclib.h"
#include "aglib.h"

int chdev1 ;

if ((chdev1 = ISX_ag_open(0, 0, 0, -1, NULL)) == -1) {
  printf( "Cannot open the channel");
  exit(1);
}

/* ring the station five times */
if (ISX_ags_genring(chdev1, 5, EV_ASYNC)== -1){
  printf("Error Message = %s", ISX_ATDV_ERRMSGP(chdev1));
  exit(1);
}

/* 2 seconds later, ringing has not completed and station 2
* has not gone off-hook. However, there is a need to abort the
* ringing on station 2. Issue the abort command
*/
if (ISX_ags_stopfn(chdev1, MTF_RING, EV_SYNC) == -1) {
  printf("Error Message = %s", ISX_ATDV_ERRMSGP(chdev1));
  exit(1);
}
```

相关信息

- **ISX_ags_genring()**

2.7 Tone 操作函数

▪ *ISX_ag_StartSendTone()*

函数名称：	int ISX_ag_StartSendTone(dev, iToneId, tptp, iPrdDev, mode, ulOperIndex)	
输入参数：	**int dev**	模拟线通道设备句柄
	const int iToneId	TONE 音 ID 号
	DV_TPT *tptp	中止播放条件表指针
	int iPrdDev	PRD 设备句柄号
	unsigned short mode	同步/异步模式
	ULONG ulOperIndex	操作编号
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	TONE 操作函数	
模式：	同步、异步	

描 述

ISX_ag_StartSendTone( )函数用于发送忙音、回铃音、催挂音、拨号音等。调用该函数，必须把模拟线通道与DSP语音通道绑定；有关与DSP语音通道绑定请参见ISX_ag_open()函数。

参数	描述
dev	指定有效的语音通道设备句柄。
iToneId	TONE 音的 ID 号，可以为： TONEID_BUSY：忙音 TONEID_DIAL：拨号音 TONEID_RINGBACK：回铃音 TONEID_HOWLER：催挂音
tptp	中止播放条件表（**DV_TPT**）指针，该参数指定播放操作的中止条件，详细请参见**DV_TPT**结构说明。**注：**除**DV_TPT** 的中止条件外，当播到文件尾、或用户调用了**ISX_dx_stopch()**函数等，也可以使播放操作中止，详细参见**ISX_ATDX_TERMMSK()**、**ISX_ATAG_TERMMSK()**函数说明里的中止原因列表。
iPrdDev	PRD 设备句柄。由于目前发送 TONE 音是通过 DSP 语音通道进行文件播放来实现，因此该参数指定 TONE 音文件所在的 PRD 设备句柄。
Mode	指定同步/异步模式： EV_ASYNC：异步模式 EV_SYNC：同步模式（默认）

ulOperIndex	操作编号，若采用异步操作模式，当操作完成时产生**AGEV_SENDTONE**或**AGEV_ERROR**时，可以通过ISX_sr_getevtoperindex()函数获得该编号。

注：

1. TPT 条件受到绑定的 DSP 语音通道的历史按键和当前按键控制。
2. 可以使用 ISX_ag_StopSendTone(dev)函数来停止播放，但 dev 必须是 ISX_ag_StartSendTone(dev)函数中的模拟线通道句柄。
3. 可以使用 ISX_dx_stopch(dev)函数来停止播放，但 dev 必须是绑定的 DSP 语音通道设备句柄。
4. 播放过程中，若有按键，可以通过 ISX_dx_getdig(dev)函数获得按键情况，但 dev 必须是绑定的 DSP 语音通道设备句柄。

错 误

如果函数调用失败则返回－1，请调用**ISX_ATDV_LASTERR( )**和**ISX_ATDV_ERRMSGP( )**提取失败原因；可能的失败原因如下：

EDX_BADPARM：无效的参数
EDX_BADIOTT：无效的 DX_IOTT 参数
EDX_BADTPT：无效的 DX_TPT 参数
EDX_BUSY：设备忙

例 子

```
#include <srllib.h>
#include <voclib.h>
#include <aglib.h>
main()
{
  UCHAR ucIsxNo = 0;
  UCHAR ucAgBrd = 4;
  UCHAR ucAgChannel = 1;
  int iRetVal;

  UCHAR ucDspBrd = 1;
  USHORT usDspChannel = 0;
  int dspdev = ISX_dx_open(DT_DSP_CH, ucIsxNo, ucDspBrd, usDspChannel, NULL);

  int agdev = ISX_ag_open(ucIsxNo, ucAgBrd, ucAgChannel, dspdev, NULL);

  DV_TPT tpt[1];
  tpt[0].tp_type   = IO_EOT;
  tpt[0].tp_termno = DX_MAXDTMF;
  tpt[0].tp_length = 1;
  tpt[0].tp_flags  = TF_MAXDTMF;
  int iPrdDev = ISX_dx_open(DT_PRD, 0, 0, 0, NULL);
  ISX_ag_StartSendTone(agdev, TONEID_BUSY, tpt, iPrdDev, EV_ASYNC, 1);
  Sleep(10*1000);
  ISX_ag_StopSendTone(agdev, EV_SYNC);
   ...
}
```

相关信息

- **ISXE_dx_PlayMultiFiles()**
- **ISX_ATDX_TERMMSK()**
- **ISX_ATAG_TERMMSK()**
- **DV_TPT**
- ISX_ag_StopSendTone()

▪ *ISX_ag_StopSendTone()*

函数名称：	short ISX_ag_StopSendTone(dev, mode)	
输入参数：	**int chdev**	模拟线通道设备句柄
	unsigned short mode	同步异步模式
返回值：	0 成功 -1 失败	
头文件：	srllib.h voclib.h aglib.h	
类别：	TONE 操作函数	
模式：	同步、异步	

描 述

ISX_ag_StopSendTone( )函数停止由ISX_ag_StartSendTone()启动的发送TONE音操作。

参数	描述
dev	模拟线通道设备句柄。
mode	同步/异步模式： EV_SYNC ： 同步 EV_ASYNC ： 异步

警 告

无

错 误

无

例 子

参见 ISX_ag_StartSendTone()函数的例子。

相关信息

- **ISX_ag_StartSendTone()**

第二章 数据结构

AG_CAP

```
typedef struct ag_cap {
   unsigned short ca_nbrdna;    /* # of rings before no answer. */
   unsigned short ca_stdely;    /* Delay after dialing before analysis. */
   unsigned short ca_cnosig;    /* Duration of no signal time out delay. */
   short          ca_lcdly;     /* Delay after dial before lc drop connect */
   unsigned short ca_lcdly1;    /* Delay after lc drop con. before msg. */
   unsigned short ca_hedge;     /* Edge of answer to send connect message. */
   unsigned short ca_cnosil;    /* Initial continuous noise timeout delay. */
   unsigned short ca_lo1tola;   /* % acceptable pos. dev of short low sig. */
   unsigned short ca_lo1tolb;   /* % acceptable neg. dev of short low sig. */
   unsigned short ca_lo2tola;   /* % acceptable pos. dev of long low sig. */
   unsigned short ca_lo2tolb;   /* % acceptable neg. dev of long low sig. */
   unsigned short ca_hi1tola;   /* % acceptable pos. dev of high signal. */
   unsigned short ca_hi1tolb;   /* % acceptable neg. dev of high signal. */
   unsigned short ca_lo1bmax;   /* Maximum interval for shrt low for busy. */
   unsigned short ca_lo2bmax;   /* Maximum interval for long low for busy. */
   unsigned short ca_hi1bmax;   /* Maximum interval for 1st high for busy */
   unsigned short ca_nsbusy;    /* Num. of highs after nbrdna busy check. */
   unsigned short ca_logltch;   /* Silence deglitch duration. */
   unsigned short ca_higltch;   /* Non-silence deglitch duration. */
   unsigned short ca_lo1rmax;   /* Max. short low  dur. of double ring. */
   unsigned short ca_lo2rmin;   /* Min. long low  dur. of double ring. */
   unsigned short ca_intflg;    /* Operator intercept mode. */
   unsigned short ca_intfltr;   /* Minimum signal to qualify freq. detect. */
   unsigned short ca_1strngbak; /*在没应答之前，等待第一个回铃音最大延时时间*/
   unsigned short rfu2;         /* reserved for future use */
   unsigned short rfu3;         /* reserved for future use */
   unsigned short rfu4;         /* reserved for future use */
   unsigned short ca_hisiz;     /* Used to determine which lowmax to use. */
   unsigned short ca_alowmax;   /* Max. low before con. if high >hisize. */
   unsigned short ca_blowmax;   /* Max. low before con. if high <hisize. */
   unsigned short ca_nbrbeg;    /* Number of rings before analysis begins. */
   unsigned short ca_hi1ceil;   /* Maximum 2nd high dur. for a retrain. */
   unsigned short ca_lo1ceil;   /* Maximum 1st low dur. for a retrain. */
   unsigned short ca_lowerfrq;  /* Lower allowable frequency in hz. */
   unsigned short ca_upperfrq;  /* Upper allowable frequency in hz. */
   unsigned short ca_timefrq;   /* Total duration of good signal required. */
   unsigned short ca_rejctfrq;  /* Allowable % of bad signal. */
   unsigned short ca_cnct_t;    /* FXO 口用来判断 CR_CNCT 是否接通的最大时间*/
   unsigned short ca_ansrdgl;   /* Silence deglitching value for answer. */
   unsigned short ca_mxtimefrq;  /* max time for 1st freq to remain in bounds */
   unsigned short ca_lower2frq;  /* lower bound for second frequency */
   unsigned short ca_upper2frq;  /* upper bound for second frequency */
   unsigned short ca_time2frq;   /* min time for 2nd freq to remains in bounds */
   unsigned short ca_mxtime2frq; /* max time for 2nd freq to remain in bounds */
   unsigned short ca_lower3frq;  /* lower bound for third frequency */
   unsigned short ca_upper3frq;  /* upper bound for third frequency */
   unsigned short ca_time3frq;   /* min time for 3rd freq to remains in bounds */
   unsigned short ca_mxtime3frq; /* max time for 3rd freq to remain in bounds */
   unsigned short ca_dtn_pres;   /* Length of a valid dial tone (def=1sec) */
   unsigned short ca_dtn_npres;  /* Max time to wait for dial tone (def=5sec)*/
   unsigned short ca_dtn_deboff; /* The dialtone off debouncer (def=100ms) */
   unsigned short ca_pamd_failtime; /*Wait for AMD/PVD after cadence
                                      break(default=4sec)*/
   unsigned short ca_pamd_minring;  /* min allowable ring duration (def=1.9sec)*/
```

```
  unsigned char ca_pamd_spdval; /* Set to 2 selects quick decision (def=1) */
  unsigned char ca_pamd_qtemp;  /* The Qualification template to use for PAMD */
  unsigned short ca_noanswer;   //time before no answer after first ring(default=30sec)
  unsigned short ca_maxintering; // Max inter ring delay before connect (8 sec)
} AG_CAP;
```

描 述

AG_CAP用来描述Call Analysis的参数。

域描述

AG_CAP 的很多域是不支持的，下面只罗列目前所支持的域：

ca_dtn_npres

等待拨号音的长度（单位为 10ms），默认为 5 秒。

ca_cnct_t

FXO 口用来判断 CR_CNCT 是否接通的最大时间(单位:10ms)，默认为 500ms。

ca_1strngbak

在没应答之前，等待第一个回铃音最大延时时间（单位：10ms），默认为 30 秒。

例 子

无。

第三章 事件列表

▪ *MOEV_DIAL*

FXO口模拟线呼出结束事件，结束原因可以通过**ISX_ATAG_TERMMSK()**函数获得。呼出操作函数：**ISX_ago_dial()**。只有采用异步模式而且不启动Call Progress Analysis才会收到该事件。

▪ *MOEV_CALLP*

FXO口模拟线呼出结束事件，结束原因可以通过**ISX_ATAG_TERMMSK()**函数获得。呼出操作函数：**ISX_ago_dial()**。只有采用异步模式而且启动Call Progress Analysis才会收到该事件。

▪ *AGEV_ERROR*

FXO 口模拟线摘机挂机、发送 TONE 音等操作失败事件，具体是哪种操作引起的失败，可通过调用 ISX_sr_getevtopertype()函数获得。详细的错误描述可以通过调用 ISX_ATDV_ERRMSGP()函数获得。只有采用异步模式操作才会收到该事件。

▪ *MSEV_RING*

FXS口模拟线振铃成功结束事件，结束原因可以通过**ISX_ATAG_TERMMSK()**函数获得，也可以通过事件的事件数据来获得。若振铃失败则产生**MSEV_NORING**事件。振铃操作函数：**ISX_ags_genring()**。只有采用异步模式才会收到该事件。该事件的带有事件数据，该事件数据为DX_CST结构。例如：

```
case MSEV_RING:
{
    DX_CST *cstp;
    cstp = (DX_CST *)ISX_sr_getevtdatap();
    if(cstp->cst_data == MSMM_RNGOFFHK){
        printf("Solicited off hook");
    }
    else if(cstp->cst_data == MSMM_TERM){
        printf("Ring termination");
    }
    else if(cstp->cst_data == MSMM_RNGSTOP){
        printf(" User stopped ringing by stopfn() cmd");
    }
    break;
}
```

▪ *MSEV_RINGS*

FXS振铃通知事件。用户可以通过对该事件进行计数。振铃操作函数：**ISX_ags_genring()**。只有采用异步模式才会收到该事件。该事件的带有事件数据，事件数据为振铃计数。例如：

```
/*使能事件*/
ISX_ag_SetEvtMsk(nAgHdl, MSMM_RINGS, AGACT_ADDMSK);

/*事件处理*/
int nAgHdl = ISX_sr_getevtdev();
case MSEV_RINGS:
{
    int nRingNum = *(int *)ISX_sr_getevtdatap();
    if(nRingNum == 1){
        printf("recv first ring");
        ISX_ags_SendCallerID(nAgHdl, “13632366794”);
```

```
    }
    break;
}
```

注意：可以通过**ISX_ag_SetEvtMsk()**函数禁止接收此类事件，系统默认是禁止接收该事件。

▪ *MOEV_SETHOOK*

FXO口模拟线摘机挂机成功结束事件，若摘机挂机操作失败则会产生**AGEV_ERROR**事件。摘机挂机操作：**ISX_ago_sethook()**。只有采用异步模式才会收到该事件。该事件的带有事件数据，该事件数据为DX_CST结构。例如：

```
case MOEV_SETHOOK:
{
    DX_CST *cstp;
    cstp = (DX_CST *)ISX_sr_getevtdatap();
    if(cstp->cst_data == AG_OFFHOOK){
        printf("Off hook termination");
    }
    else if(cstp->cst_data == AG_ONHOOK){
        printf("On hook termination");
    }

    break;
}
```

▪ *MSEV_NORING*

FXS口模拟线振铃失败结束事件。该事件不带事件数据。若是成功结束则产生**MSEV_RING**事件。振铃操作函数：**ISX_ags_genring()**。

▪ *MSEV_SIGEVT*

FXS 摘机挂机通知事件。具体是摘机（MSMM_OFFHOOK）、挂机（MSMM_ONHOOK）还是拍插簧（MSMM_HOOKFLASH）则由事件数据指出；事件数据为 DX_CST 结构。例如：

```
case MSEV_SIGEVT:
{
    DX_CST *cstp;
    cstp = (DX_CST *)ISX_sr_getevtdatap();
    if(cstp->cst_data == MSMM_ONHOOK){
        printf("On hook event");
    }
    else if(cstp->cst_data == MSMM_OFFHOOK){
        printf("Off hook event");
    }
    else if(pCst->cst_data == MSMM_HOOKFLASH) {
        printf("hook flash event");
    }

    break;
}
```

注意：可以通过**ISX_ag_SetEvtMsk()**函数禁止接收此类事件，系统默认是允许接收该事件。

▪ *MOEV_WTRING*

FXO口模拟线等待振铃结束事件；等待振铃操作函数：**ISX_ago_wtring()**。只有采用异步模式才会收到该事件。

▪ *MOEV_RNGON*

FXO振铃开始通知事件。表示一个新的呼入，无事件数据，影响该事件触发的函数为**ISX_ago_wtring()**。

▪ *MOEV_RNGOFF*

FXO振铃结束通知事件。表示一个呼叫的结束，无事件数据，影响该事件触发的函数为**ISX_ago_wtring()**。

▪ *MOEV_RINGS*

FXO振铃通知事件。用户可以通过对该事件进行计数，当收到一定数量的该事件就可以通过调用**ISX_ago_sethook()**函数进行摘机，这样的操作就类似**ISX_ago_wtring()**函数的行为了。事件数据为振铃计数，当事件数据为 1 时表示一次新的呼叫。例如：

```
char szCallId[MAX_CALLID_DATA_LEN];
/*事件处理*/
case MOEV_RINGS:
{
    int nRingNum = *(int *)ISX_sr_getevtdatap();
    if (nRingNum == 1) {
        //表示一次新的呼叫
    } else if ( nRingNum == 2 ) {
        int iRetVal = ISX_ago_gtcallid(iHdl, szCallId);
        printf("szCallId = '%s' ", szCallId);
    }
    break;
}
```

注意：可以通过**ISX_ag_SetEvtMsk()**函数禁止接收此类事件，系统默认是禁止接收该事件。

▪ *MOEV_CALLERID*

FXO CallerID产生通知事件。该事件为主动事件，用于通知用户获取CallerID时机已到，当收到该事件后可通过调用**ISX_ago_gtcallid()**或**ISX_ago_gtextcallid()**函数获取，例如：

```
char szCallId[MAX_CALLID_DATA_LEN];
/*事件处理*/
case MOEV_CALLERID:
{
    int iRetVal = ISX_ago_gtcallid(iHdl, szCallId);
    printf("szCallId = '%s' ", szCallId);
    break;
}
```

注意：可以通过**ISX_ag_SetEvtMsk()**函数禁止接收此类事件，系统默认是禁止接收该事件。

▪ *AGEV_DIGITS*

按键或 TONE 通知事件。具体是哪个按键值或 TONE 则由事件数据指出；事件数据为 DX_CST 结构。DX_CST::cst_event 域表示事件属性，DX_CST::cst_data 表示事件数据。

cst_event 可能出现的值如下：

DE_DIGITS ： Digit Received
DE_TONEON ： Tone ON Event Received
DE_TONEOFF ： Tone OFF Event Received

当 cst_event 为 TONE 时，**cst_data** 可能出现的值如下：

TONEID_BUSY ：忙音
TONEID_RINGBACK ：回铃音

例如：

```
case AGEV_DIGITS:
{
      DX_CST *cstp;
      cstp = (DX_CST *)ISX_sr_getevtdatap();
      if(cstp->cst_event == DE_DIGITS)
          printf("receive DTMF: %c", cstp->cst_data);
      else if(cstp->cst_event == DE_TONEON)
          printf("receive TONE: %d", cstp->cst_data);

      break;
}
```

注意：可以通过**ISX_ag_SetEvtMsk()**函数禁止接收此类事件，系统默认是禁止接收该事件。

- ***AGEV_SENDTONE***

发送TONE音成功结束事件，结束原因可以通过**ISX_ATAG_TERMMSK()**函数获得。若发送TONE音失败则产生**AGEV_ERROR**事件。振铃操作函数：**ISX_ag_StartSendTone()**。只有采用异步模式才会收到该事件。

- ***MOEV_RONHOOK***

主动事件，该事件指示远端已经挂机，该事件无事件数据。