DS-4000 系列
视音频压缩/解码板卡
系统 SDK 编程指南
（for Linux）
V5.1

杭州海康威视数字技术股份有限公司
http://www.hikvision.com
技术热线：400-700-5998

2009-10

非常感谢您购买我公司的产品，如果您有什么疑问或需要请随时联系我们。

本手册适用于 DS-4000 系列视音频压缩/解码板卡。

本手册可能包含技术上不准确的地方、或与产品功能及操作不相符的地方、或印刷错误。我司将根据产品功能的增强而更新本手册的内容，并将定期改进或更新本手册中描述的产品或程序。更新的内容将会在本手册的新版本中加入，恕不另行通知。

目 录

1 产品简介

海康威视 DS-4000 系列是面向数字监控行业而推出的专用板卡，采用了高性能的视频压缩技术标准 H.264 及 OggVorbis（相当于 G.722）或 G.711 的音频编码标准，完全依靠硬件实现了视频及音频的实时编码（CIF 格式 25 帧 PAL / 30 帧 NTSC）并精确同步，实现了动态码率、可控帧率、帧模式选择、动态图像质量控制，音频预览、视频丢失报警等功能，能独立调整各通道参数，性能稳定而且可靠。与 MPEG-I 产品相比，在保持同等图像质量的前提下，能大大节省存储空间、并非常适合宽带网或窄带网的传输，是新一代数字监控产品的最佳选择。

海康威视 DS-4000 系列板卡 SDK 分为三部分，分别为系统 SDK、网络 SDK、播放 SDK，本文档专门描述系统 SDK，其他 SDK 请参照我公司相关文档。系统 SDK 是专门为该系列板卡设计的本地录像软件接口程序，以动态连接库的形式提供给应用软件开发者，并同时附有演示程序（HikVision H.264 System Demo）及其源码，能有效地缩短应用软件的开发周期。

在使用过程中，特别提醒软件开发人员，DS-4000 系列板卡可以在编码的同时修改除码流类型（复合流、纯视频流、音频流）外的所有参数，包括分辨率、码流、帧结构等。譬如在压缩过程中可改变帧率（SetIBPMode）、量化系数（SetDefaultQuant）、分辨率、码流、帧结构而无须停止、启动压缩。播放器会自动识别帧率、分辨率等参数，按当前压缩帧率、分辨率播放且声音图像播放保持正常。

通过动态修改量化系数（I、B、P）可控制压缩码率，当码率太高时，加大量化系数；当码率太低时，减少量化系数。当然，在量化系数满足的情况下，不必再降低量化系数。

DS-4000 系列压缩卡的运动检测独立于压缩，不进行压缩也可以进行运动检测。可动态改变帧率非常有价值，在无运动时按低帧率录像，运动时按高帧率录像，记录在同一个文件内，可大大节省硬盘空间。

DS-42xx 系列板卡包含 DS-4216HC、DS-4208HFV、DS-4216HFV　3 种型号，性能稳定，功耗低：

DS-4216HC 提供 16 路 4CIF/2CIF 非实时编码或者 16 路 CIF/QCIF 实时编码，同时支持 16 路 CIF/QCIF 子通道编码；

DS-4208HFV 提供 8 路 4CIF/2CIF/CIF/QCIF 实时编码，同时支持 8 路 CIF/QCIF 子通道编码；

DS-4216HFV 提供 16 路 4CIF/2CIF/CIF/QCIF 实时编码，同时支持 16 路 CIF/QCIF 子通道编码。

DS-41xx 系列板卡包含 DS-4108HCV、DS-4116HCV　2 种型号，采用 DM 648 DSP：

DS-4108HCV 包含 1 个 DSP，DS-4116HCV 包含 2 个 DSP， 每个 DSP 支持 8 路 DCIF/2CIF/CIF/QCIF，或者 4 路 4CIF 分辨率音视频压缩，每张板卡支持 1 路模拟视频矩阵输出和 1 路模拟音频矩阵输出功能，DS-4100 系列板卡的音频实时监听功能不再需用 4 针线连接板卡和声卡音频输入口。

DS-40xx 系列板卡包含 HC、HC+、HCS、HF、HS 等型号，采用 DM642DSP：

DS-4004HC/HC+支持 4 路的 DCIF/2CIF/CIF/QCIF 实时编码压缩，也支持 2 路的 4CIF 实时编码压缩。若需要作为 4CIF 编码录像，应用程序可以从 DS-4004HC 的 4 个编码通道中任意选取两个通道设置为 4CIF 分辨率，然后对这两个通道进行录像，此时，DS-4004HC/HC+卡的另外两个通道的图像可以作为视频预览或者不予以显示；

DS-4008HC/HC+板卡支持 8 路的 DCIF/2CIF/CIF/QCIF 实时编码压缩，也支持 4 路的 4CIF 实时编码压缩。若需要作为 4CIF 编码录像，应用程序可以从 DS-4008HC/HC+的 8 个编码通道(编码通道为 0，1，2，3，4，5，6，7)中的前面 4 个通道(0,1,2,3)任意选取两个通道设置为 4CIF 分辨率，再从后面 4 个编码通道(4,5,6,7)

中任意选取两个通道设置为 4CIF 分辨率，然后对这选中的四个通道进行录像；

DS-4016HC 板卡支持 16 路的 DCIF/2CIF/CIF/QCIF 实时编码压缩，也支持 8 路的 4CIF 实时编码压缩。若需要作为 4CIF 编码录像，应用程序可以从 DS-4016HC 的 16 个编码通道(编码通道为 0~15)中的每 4 个通道（0、1、2、3 或者 4、5、6、7 或 8、9、10、11 或 12、13、14、15）中任意选取两个通道设置为 4CIF 分辨率，然后对这选中的八个通道进行录像；

对于 DS-4004HC、DS-4008HC、DS-4016HC 板卡，通过子通道编码，可以把每一个通道全部设置为 4CIF 分辨率（SetSubEncoderPictureFormat），这样每一个通道就都可以实现 4CIF 编码，然后通过函数 StartSubVideoCapture 实现每个通道的 4CIF 分辨率录像。在一般场景下，每路图像都可以达到 15 帧以上。

DS-4016HCS：16 路视音频压缩卡，支持 16 路 CIF/QCIF 音视频实时压缩，不支持 4CIF、2CIF、DCIF 分辨率，不支持双编码。

DS-4004HF、DS-4008HF：全 D1 编码卡，每个通道均可进行 4CIF 实时编码。

DS-4008HS、DS-4016HS：一芯八路视音频压缩板卡，每个 DSP 支持 8 路 CIF/QCIF 音视频实时压缩，不支持 4CIF、2CIF、DCIF 分辨率，不支持双编码。

2 SDK版本更新

V5.1（2009-8-12）

- 全面兼容 DS-42xx 系列、DS-41xx 系列、DS-40xx 系列板卡
- 为解决某些主板上不带 cd_in 口的问题，DS-41xx 、42xx 系列板卡可以支持不接 4 帧音频线直接进行音频预览功能

DS-42xxHF

- 增加本卡模拟视频输出功能，支持视频单画面模拟输出（调用函数 SetEncoderVideoExtOutput）。
- 增加本卡模拟音频输出功能（调用函数 SetEncoderAudioOutput）

DS-4216HC

- 增加了 2CIF 以及 4CIF 主通道非实时编码，DS-4216HC 卡启动 4CIF 主编码时，无子通道编码

5.0 _1524 版本（2009-3-24）

更新

- 支持 DS-4004HF 板卡。
- DS-4108HCV 和 DS-4116HCV 分别支持 8 路和 16 路非实时 4CIF 子通道编码。
- 优化 DS-41xx 系列板卡的编码性能。

修正 bug

- 修改移动侦测区域设置出错的问题。
- 修改 LOGO 功能设置某些关键色无效的问题。
- 修改 DS-40xx 系列板卡 JPEG 抓图有白线的问题。

5.0 版本（2008-10-6）

更新

- 支持 DS-4100HCV 系列板卡（DS-4108HCV、DS-4116HCV）。
- 5.0 版本 SDK 兼容 DS-41xx 系列、DS-4000HC/HC+/HCS/HF/HS/MD/系列板卡，不再兼容 DS-4000H 系列板卡。
- 新增音频矩阵输出功能，可将任意编码通道或者解码通道的音频数据输出到任意模拟音频输出口上，本功能适用于 HCV 卡和 MD 卡。
- HCV 卡视频矩阵输出功能使用函数 SetEncoderVideoExtOutput 实现，与 MD 卡本地矩阵输出功能相同，音频矩阵输出使用新增函数 SetEncoderAudioOutput 或者 SetEncoderAudioOutputExt 实现。

修正 bug

- 解决了移动侦测区域判断出错的问题
- 解决了 HS 卡全屏预览反复启停时错位的问题。

新增 API 函数（具体请参看函数说明）

SetEncoderAudioOutput

SetEncoderAudioExtOutput

SetDecoderAudioExtOutput

SetAudioPCIPreview

4.3版本(2008-4-13)

更新

- 编解码通道上限扩展到 256。
- DS-40xxMD 卡增加解码视频捕获功能，HW_SetDecoderVideoCapture() 。
- 本地矩阵输出功能增加支持帧率控制功能，SetEncoderVideoExtOutput() 。
- DS-40xxMD 卡解码延时降低。
- DS-40xxHS 卡支持 YUV 抓图、JPEG 抓图、原始视频捕获。
- DS-40xxHS 卡支持矩阵输出。
- 支持纯音频流编码。
- 完善了对视频信号检测的判断。
- 在 HC、HC+、HF 卡上增加了色度串扰的处理。
- 编解码性能提升。

修正 bug

- 修正退出应用程序时，界面已经关闭，但 SDK 可能还没有彻底退出，此时如果再启动应用程序，可能会导致死机的问题。
- 调整 DS-40xxMD 卡启动后默认的音频输出为关闭状态，之前版本为输出前两路音频。
- 修正从 4.2 版本开始，GetSoundLevel()在 HCS 卡的前 12 路上无法正确执行的问题。
- 修正如果用户采用多线程来输入码流，DS-40xxMD 卡可能会出现多路图像混叠情况。
- 修正 DS-40xxMD 卡回放时文件尾部数据可能无法解码的问题。
- 修正抓图问题。抓 BMP 时，可能导致图像错位；抓 JPEG 时，可能会返回超时，并且无法恢复。
- 修正 DS-40xxMD 卡解码 N 制 QCIF 花屏问题。
- 修正 DS-40xxMD 卡解码花屏。

新增函数接口（具体请参看函数说明）

RegisterDecoderVideoCaptureCallback()

HW_SetDecoderVideoCapture()

4.2版本(2006-8-15)

更新

- DS-40xxHC 卡增加新的 api 函数 StartVideoPreviewEx(), 以扩充 StartVideoPreview()函数功能,提供了更多种显示选择
- DS-40xxMD 卡增加播放多编码图像功能
- DS-40xxMD 卡增加新的 api 函数: HW_SetDecoderPostProcess(), 功能是防止解码时显示图像出现闪烁的情况
- DS-40xxMD 卡增加新的 api 函数: HW_InputDataByFrame(), 功能与函数 HW_InputData()类似, 而且支

持 I 帧数据完整地输入

- 更新了 MD 卡解码器，包括 I 帧的解码器，解决 MD 卡实时显示的图像问题，并且完善和改进了图像的质量
- 提高了程序的线程安全性

修正 bug

- DS-40xxMD 卡在关闭主机解码视频(HW_StopDecChanVgaDisplay)时存在的内存泄漏
- 修正 DS-40xxMD 在播放图像时可能出现的 ErrorCodeNoSpare 错误
- 修正出现大 I 帧时,MD 卡程序可能出现停止解码或死掉的情况
- 修正 DS-40xxHC 在只有子通道以混合流的方式记录时 PC 机会自动关机的现象
- 修正一些 api 函数输出错误

4.1版本(2005-10-25)

更新

- 支持全新推出的 DS-40xxHC+ 卡
- 编码性能优化，全面提升图像质量，特别是 4CIF 的图像质量有很大提高
- DS-40xxMD 卡增加文件索引功能，支持按照时间或帧号定位功能，并可以获取录像文件的起止时间

修正 bug

- DS-40xxMD 卡无法解码某些子通道的录像文件
- DS-40xxMD 矩阵输出时可能会出现图像错误
- DS-40xxMD 卡回放小文件时，可能会误报文件结束
- DS-40xxHC 原始图像流的帧率控制无效(Ver:4.0)
- 录像音频的音量偏小(Ver:3.0-4.0)

4.0 版本(2005-07-25)

更新

- 支持新的板卡：DS4016HCS、DS4004MD、DS4002MD（矩阵解码卡）支持
- DS4016HCS：16 路视音频压缩卡。支持 16 路 CIF 实时压缩，支持 CIF/QCIF 分辨率，不支持 4CIF、2CIF、DCIF 分辨率，不支持双编码。新增加了 WatchDog 和报警输入、输出功能。
- DS4002MD： 4 路解码、2 路输出矩阵解码卡。 基于 DS4002MD 可以实现视频矩阵和硬件解码功能（请参考《DS400xMD、数字视频矩阵方案》）。
- DS4004MD： 8 路解码、4 路输出矩阵解码卡。产品功能和 2 块 DS4002MD 相同。
- 增加新的移动侦测接口 SetupMotionDetectionEx()，提供了更灵活的功能，并且简化了用户的工作量；对于移动侦测的操作应用程序仅需调用 3 个接口函数：SetupMotionDetectionEx()，StartMotionDetection()和 StopMotionDetection()。
- 在应用程序以新的接口函数实现移动侦测功能时，SDK 不再返回移动侦测帧，而仅仅
- 是通过函数 SetupMotionDetectionEx() 所调用的回调函数 MotionDetectionCallback() 的参数 bMotionDetected 告知应用程序视频是否处于移动状态。
- 增加新的 OSD 接口 SetOsdDisplayModeEx()，最多支持 8 行 OSD 字符。同时，修改 OSD 参数时无需重新启停。 修正 OSD 时钟不准确问题。OSD 时钟始终以主机时钟为准，同时 SetupDateTime() 函数不再有效，用户无须自行校时。

- SDK 内部增加了异常检测、恢复机制，增强系统稳定性，无需用户干预。
- 在 DS4016HCS 上实现了 WatchDog 功能,接口函数为 SetWatchDog()，只要打开任意一块 DS4016HCS 的 WatchDog 功能，就可以实现对上层软件和系统中所有压缩板卡的运行状态监控。
- 在 DS4016HCS 上增加了报警输入、输出功能，当配合报警卡使用时，一块 DS4016HCS 支持 16 路报警输入和 4 路报警输出，同时增加 RS485 串口，并提供了简单、实用的串口操作 API。
- 增加新的接口函数 GetJpegImage()，支持 JPEG 方式抓图，抓取的图像质量动态可调。
- 采用新的预览实现方式，在 sdk 中将 SDL 模块剥离，以便无预览和 server 应用的方便实现。Sdk 提供预览原始数据流，完全由应用程序实现图像的预览，可采用定时器或 semaphore 控制预览。在新的 demo：dsdemo 仍采用 SDL 模块实现，但 SDL 的功能并不太尽人意，如客户有更好的选择，现提供了原始数据流，更便于实现不同的预览方式，以达到 linux 下也有好的预览。在 SDL1.2.8 版本，及某些显卡（如 NVIDA）下，SDL 可实现硬件加速功能，可以达到低 cpu 利用率和高显示水准的效果。在 dsdemo 中默认将硬件加速功能打开，即已将 SDL_VIDEO_YUV_HWACCEL 环境变量设为 1 了，如果出现预览图像乱闪的现象，表明所使用系统不支持硬件加速，可尝试升级 SDL 及显卡驱动，如还不行，那只能将该功能关闭，即将 SDL_VIDEO_YUV_HWACCEL 设为 0。

DS400xMD 卡介绍：实现**视频矩阵**和**硬件解码**功能

1 片 DS4004MD 即相当于 2 片 DS4002MD 卡

1、解码功能

- 每块 DS4002MD 可做 4 路解码。
- 支持的码流格式：海康威视 H、HC 系列板卡；海康威视 M、ME、ATM、HC、DVS 系列嵌入式设备。
- 音、视频输出：
  a) 音频输出：2 路，可在 4 个解码通道中任选 2 路输出。
  b) 视频输出：2 路，每路视频输出最多可以划分为 16 个窗口。视频输出功能请参考“视频矩阵部分”的说明。
  c) 音频预览：每块 DS4002MD 支持 1 路音频预览输出。
- 软件：从 4.0 版 SDK 开始提供对 DS400xMD 的支持。
  a) 支持 HC 卡和 MD 卡在 1 台 PC 内混插。
  b) 在一个 SDK 内同时支持 HC 卡和 MD 卡。
- 目前 1 台 PC 最多支持 16 块 DS4002MD 卡，即最多支持 64 路解码，32 路视频输出。
- 基本解码性能（值为每解码 1 路视频大约要占用的 DSP 资源）：
  a) CIF：12%（512Kb）；16%（2Mb）
  b) 2CIF：30%（1Mb）
  c) DCIF：28%（768Kb）
  d) 4CIF：50%（1.5Mb）；60%（3Mb）

  ※上述测试文件为定码率下的稳定图像。

  ※目前对解码器的进一步优化正在进行中，其性能在以后的版本中会不断的得到提升。

2、矩阵功能

视频矩阵功能可以概括为：

- 视频输入端：由 HC 卡实时采集的视频、MD 卡解码后视频（本地文件或网络实时流）。
- 视频输出端：MD 的视频输出通道。视频输出支持画面分割，每路视频输出最多可划分为 16 窗口，视频矩阵以窗口为单位进行图像切换。
- 矩阵控制：对于 1 台 PC 中的所有 HC 卡和 MD 卡，HC 卡的每个编码通道和 MD 卡的每个

解码通道，都可以把本通道的视频输出到任意一块 MD 卡的任意一路显示通道中的任意一个窗口进行显示。

矩阵的基本参数：

- 每块 DS4002MD 支持 2 路矩阵输出，每路输出为 4CIF 分辨率。
- HC 卡的每个编码通道可以同时支持 1 路显卡预览和 1 路矩阵输出。
- MD 卡的每个解码通道可以同时支持 1 路显卡输出和 2 路矩阵输出。
- 每路视频输出都支持画中画功能，每个窗口的位置动态可调。
- 每路视频输出总的窗口面积之和不能超过 4CIF＋QCIF，即最大可以实现一个 4CIF 的全屏输出＋1 个 QCIF 的画中画输出。

3 数据类型及结构体定义

3.1 数据类型定义

3.1.1 帧类型定义

PktError	非法帧数据
PktSysHeader	系统头
PktIFrames	I 帧包
PktPFrames	P 帧包
PktBBPFrames	BBP 帧包
PktAudioFrames	音频帧包
PktMotionDetection	动态监测包
PktSFrames	为 I 帧捕获时传送的帧类型
PktSubIFrames	双编码时，子通道 I 帧
PktSubPFrames	双编码时，子通道 P 帧
PktSubBBPFrames	双编码时，子通道 BBP 帧
PktSubSysHeader	双编码时，子通道系统头

3.1.2 视频标准定义

StandardNone	无视频信号
StandardNTSC	NTSC 制式
StandardPAL	PAL 制式

3.1.3 流类型

STREAM_TYPE_VIDEO	视频流
STREAM_TYPE_AUDIO	音频流
STREAM_TYPE_AVSYNC	音视频同步流

3.2 数据结构定义

3.2.1 特殊功能能力定义

```
typedef struct tagChannelCapability{
    UCHAR bAudioPreview;        音频预览
    UCHAR bAlarmIO;             报警信号
    UCHAR bWatchDog;            看门狗
}CHANNEL_CAPABILITY, *PCHANNEL_CAPABILITY;
```

3.2.2 帧数据统计

```
typedef struct tagFramsStatistics{
    ULONG   VideoFrames;        视频帧
    ULONG   AudioFrames;        音频帧
    ULONG   FramesLost;         丢失帧
    ULONG   QueueOverflow;      缓存溢出
    ULONG   CurBps;             当前码流（kb/s）
}FRAMES_STATISTICS, *PFRAMES_STATISTICS;
```

3.2.3 显示区域

```
typedef struct tagRect{
    short RectTop;
    short RectBottom;
    short RectLeft;
    short RectRight;
}RECT;
```

3.2.4 版本信息

```
typedef struct tagVersion{
    ULONG DspVersion, DspBuildNum;          DSP 版本及 BUILD 号
    ULONG DriverVersion, DriverBuildNum;    驱动版本及 BUILD 号
    ULONG SDKVersion, SDKBuildNum;          SDK 版本及 BUILD 号
}VERSION_INFO, *PVERSION_INFO;
```

3.2.5 视频预览定义

```
typedef struct _Preview_Config{
        ULONG    imageSize;      输出值：返回的图像尺寸
        ULONG    w;              输入值：想要显示图像的宽度
                                 输出值：实际显示图像的宽度
        ULONG    h;              输入值：想要显示图像的高度
                                 输出值：实际显示图像的高度
        char*dataAddr;           输出值：预览图像数据的基地址
        sem_t*   SyncSem;        输入值：the semphore for have new image coming
        sem_t*     ChangeSem;    输入值：the semphore for the image size changed，  现无效
                                        sdk 内部已经处理消化掉了这个信号量
}PREVIEWCONFIG, *PPREVIEWCONFIG;
```

3.2.6 运动检测数据说明

DS40xxHC/HF 现在提供运动强度信息来处理运动检测，设置移动侦测区域时以 32*32 像素块为单位，按 4CIF（704*576）分辨率计算，每行有 22 个块（704/32），PAL 时 18 行（576/32），NTSC 时 15 行（480/32），与编码格式无关。经过测试，这种方法比 H 卡提高了灵敏度和可靠性，并简化了返回的数据，返回的值是 18 个 DWORD，对应屏幕高度 576/32=18 行（PAL）,每个 DWORD 的 0-21 位对应屏幕宽度 704/32=22，其中 1 为运动,0 为静止。

4.0 版本的 SDK 新增了接口函数 SetupMotionDetectionEx()，提供了更灵活的功能，并且简化了用户的工作量。

4 函数调用顺序简介

A.

设置默认的视频制式	**SetDefaultVideoStandard()**

B.

初始化板卡	**InitDSPs()**

C.

获取编码通道总个数	**GetTotalChannels()**
打开通道	**ChannelOpen()**
注册获取原始图像数据流的回调函数	**RegisterImageStreamCallback()**
启动视频图像预览	**StartVideoPreview ()**

D.

//设置 OSD	
设置 OSD 显示模式(此函数支持 2 行 OSD 显示)	**SetOsdDisplayMode()**
设置 OSD 显示模式(此函数最多支持 8 行 OSD 显示)	**SetOsdDisplayModeEx ()**
设置 OSD 显示	**SetOsd()**
//设置 Logo	
将 24 位 bmp 文件转成 yuv 格式的数据	**LoadYUVFromBmpFile()**
设置 LOGO 显示模式	**SetLogoDisplayMode()**
设置 LOGO 图像位置及数据	**SetLogo()**
//设置遮挡	
设置屏幕遮挡	**SetupMask()**

E.

设置主通道的编码分辨率格式：	**SetEncoderPictureFormat()**
设置主通道编码流类型：	**SetStreamType()**
设置编码图像质量：	**SetDefaultQuant()**
设置编码帧结构、帧率：	**SetIBPMode()**
设置码流的最大比特率：	**SetupBitrateControl()**
设置码流控制模式：	**SetBitrateControlMode()**
设置图像亮度、对比度、饱和度：	**SetVideoPara()**

F. 移动侦测方式 1

设置移动侦测灵敏度：	**AdjustMotionDetectPrecision()**
设置移动侦测区域及个数：	**SetupMotionDetection()**
启动移动侦测：	**StartMotionDetection()**

移动侦测分析：	**MotionAnalyzer()**

G．移动侦测方式 2

设置移动侦测：	**SetupMotionDetectionEx()**
启动移动侦测：	**StartMotionDetection()**

H．抓图及图像保存函数

获取原始图像：	**GetOriginalImage()**
图像保存为 BMP 文件：	**SaveYUVToBmpFile()**
抓取 JPEG 格式图像：	**GetJpegImage()**

I．音频幅度获取及现场声音监听

获取现场声音音量幅度：	**GetSoundLevel()**
设置现场声音监听：	**SetAudioPreview()**

J．获取视频、SDK 及板卡相关信息

获取视频信号输入情况：	**GetVideoSignal()**
获取 SDK 版本号：	**GetSDKVersion()**
获取视频参数：	**GetVideoPara()**
获取板卡的型号和序列号：	**GetBoardInfo()**
获取帧统计信息：	**GetFramesStatistics()**
获取板卡的详细信息：	**GetBoardDetail()**
获取 DSP 的详细信息：	**GetDspDetail()**

K．启动录像(编码压缩数据)

启动主通道数据截取：	**StartVideoCapture()**

L．启动原始图像数据流的截取

启动获取原始图像数据流：	**SetImageStream()**

M．子通道的参数设置以及录像

设置子通道编码流类型：	**SetSubStreamType ()**
设置子通道的编码分辨率格式：	**SetSubEncoderPictureFormat()**
切换至子通道:	**SetupSubChannel(, 1)**
//其它参数设置方式与主通道相同，可以设置与主通道不同的编码量化系数，帧率等等	
切换回主通道:	**SetupSubChannel(, 0)**
启动子通道数据截取：	**StartSubVideoCapture()**

N．退出

停止获取原始图像数据流：	**SetImageStream()**
停止移动侦测：	**StopMotionDetection()**
停止主通道数据截取：	**StopVideoCapture()**

停止子通道数据截取：	**StopSubVideoCapture()**
停止视频图像预览：	**StopVideoPreview()**
关闭通道：	**ChannelClose()**
卸载 DSP：	**DeInitDSPs()**

目前，SDK 函数之中除了 SetStreamType()和 SetSubStreamType ()不能在板卡编码录像过程中动态设置以外，其它视频参数，譬如 OSD、Logo、分辨率、帧率、码流、图像量化系数等参数都可以在编码录像的过程之中动态调整。

5 编码卡函数说明

板卡初始化及卸载

5.1 GetLastErrorNum

int GetLastErrorNum();

说明： 获取最近一次的 SDK 运行的错误号。

返回： 在 ds40xxsdk.h 中定义的错误号。

5.2 InitDSPs

int InitDSPs();

说明： 初始化每块板卡，应在应用软件启动时完成。如果返回值为 0，则有两种可能的结果。第一种：当系统中有编码卡时，表明初始化失败，可能是没找到相应 DSP 软件模块，会返回 0 值。第二种：用此函数来初始化只有解码卡的系统，也会返回 0 值，但其实系统已经初始化成功。故建议用户使用 HW_InitDecDevice()函数来初始化解码卡，以避免混淆。其对应的接口为 DeInitDSPs。

返回： 返回 0（如上所诉，两种可能之一），其他返回值是可用的编码通道数。

5.3 DeInitDSPs

int DeInitDSPs();

说明： 关闭每块板卡上功能，应在程序退出时调用。

返回： 成功时返回 0，否则返回-1。

通道打开及关闭

5.4 ChannelOpen

int ChannelOpen(int ChannelNum, STREAM_READ_CALLBACK StreamReadCallBack);

参数： int ChannelNum 通道号（从 0 开始）

STREAM_READ_CALLBACK StreamReadCallBack 用户定义的流处理回调函数：

typedefvoid(*STREAM_READ_CALLBACK)(int cahnnelNum, char *pBuf, int frameType,int length)

int channelNum	通道号
void * pBuf	帧数据区
int frameType	帧类型
int length	帧数据长度

用户主要在这个回调函数中处理数据，包括录象，运动检测等……每次 StartVideoCapture()之后,DSP 会送上来一个系统头 PktSysHeader，每个录象文件必须以系统头开始，否则无法回放。

说明： 打开通道，注册流处理回调函数，获取相关的操作句柄，与通道相关的操作必须使用该句柄。

返回： 成功时返回有效句柄，失败时返回-1。

5.5 ChannelClose

int ChannelClose(int channelHandle)

参数： int channelHandle 通道句柄

说明： 关闭通道，释放相关资源。

返回： 成功时返回 0，否则返回-1。

取得板卡相关信息

5.6 GetTotalChannels

int GetTotalChannels()

返回： 获取系统内可使用的通道个数。如果返回小于系统中安装的通道数，表明有一 DSP 初始化失败。

5.7 GetTotalDSPs

int GetTotalDSPs()

返回： 获取系统内正确安装的 DSP 个数。如果返回小于系统中 DSP 数，表明有一 DSP 初始化失败。

5.8 GetBoardCount

int GetBoardCount()

返回： 系统中板卡的个数。

说明： 得到系统中板卡的个数。包括 HC、MD 卡。

5.9 GetBoardDetail

int GetBoardDetail (UINT boardNum,DS_BOARD_DETAIL *pBoardDetail)

参数： UINT boardNum 板卡索引

DS_BOARD_DETAIL *pBoardDetail 返回板卡信息

```
typedef struct
{
    BOARD_TYPE_DS type;          //板卡类型
    BYTE sn[16];                 //序列号
    UINT dspCount;               //板卡包含的 DSP 个数
    UINT firstDspIndex;          //板卡上第一个 DSP 在所有 DSP 中的索引
    UINT encodeChannelCount;     //板卡包含的编码通道个数
    UINT firstEncodeChannelIndex; //板卡上第一个编码通道在所有编码通道中的索引
    UINT decodeChannelCount;     //板卡包含的解码通道个数
    UINT firstDecodeChannelIndex; //板卡上第一个解码通道在所有解码通道中的索引
    UINT displayChannelCount;    //板卡包含的视频输出通道个数
    UINT firstDisplayChannelIndex; //板卡上第一个视频输出通道在所有视频输出通道中的索引
    UINT reserved1;
    UINT reserved2;
    UINT reserved3;
    UINT reserved4;
}DS_BOARD_DETAIL;
```

返回： 正确为 0，否则返回-1。
说明： 得到某块板卡的详细信息。

5.10 GetDspDetail

int GetDspDetail (UINT dspNum,DSP_DETAIL *pDspDetail)
参数： UINT dspNum DSP 索引
DS_BOARD_DETAIL * pDspDetail 返回 DSP 信息

```
typedef struct
{
    UINT encodeChannelCount;        //DSP 包含的编码通道个数
    UINT firstEncodeChannelIndex;   //DSP 上第一个编码通道在所有编码通道中的索引
    UINT decodeChannelCount;        //DSP 包含的解码通道个数
    UINT firstDecodeChannelIndex;   //DSP 上第一个解码通道在所有解码通道中的索引
    UINT displayChannelCount;       //DSP 包含的视频输出通道个数
    UINT firstDisplayChannelIndex;  //DSP 上第一个视频输出通道在所有视频输出通道中的索引
    UINT reserved1;
    UINT reserved2;
    UINT reserved3;
    UINT reserved4;
}DSP_DETAIL;
```

返回： 正确为 0，否则返回-1。
说明： 得到某个 DSP 的详细信息。

5.11 GetEncodeChannelCount

int GetEncodeChannelCount()
返回： 系统中所有的编码通道个数。
说明： 返回系统中所有的编码通道个数。

5.12 GetDecodeChannelCount

int GetDecodeChannelCount()
返回： 系统中所有的解码通道个数。
说明： 返回系统中所有的解码通道个数。

5.13 GetDisplayChannelCount

int GetDisplayChannelCount()
返回： 系统中所有的显示（视频输出）通道个数。
说明： 返回系统中所有的显示（视频输出）通道个数。

5.14 GetBoardInfo

int GetBoardInfo(int hChannelHandle, UINT *BoardType, char *SerialNo);
参数： int hChannelHandle, 通道句柄
UINT*BoardType, 板卡型号
Char *SerialNo 板卡 ID 号，内容为板卡序列号的 ASCII 的数值，次序为 SerialNo[0] 对应最高位，SerialNo[11]对应最低位。比如卡号为“４０００１００２３４５”的值

对应为 4,0,0,0,0,1,0,0,2,3,4,5 的整形数组。

返回： 正确为 0，否则返回-1。

说明： 获取板卡硬件信息。

板卡型号如下：

DS400XM	0,
DS400XH	1,
DS4004HC	2,
DS4008HC	3,
DS4016HC	4,
DS4001HF	5,
DS4004HF	6,
DS4002MD	7,
DS4004MD	8,
DS4016HCS	9,
DS4002HT	10,
DS4004HT	11
DS4108HCV	
DS4116HCV	
DS5016HC	
DS4208HFV	
DS4216HC	
DS4216HFV	

5.15 GetCapability

int GetCapability(int hChannelHandle, CHANNEL_CAPABILITY *Capability);

参数： int hChannelHandle, 通道句柄

CHANNEL_CAPABILITY *Capability 见 3.2.1 节

返回： 正确为 0，否则返回-1。

说明： 获取板卡的特殊功能信息。

启停视频预览

5.16 StartVideoPreview

int StartVideoPreview(int channelHandle, PREVIEWCONFIG* pPreviewConf, UINT useSyncSem);

参数： int channelHandle 通道句柄

PREVIEWCONFIG* pPreviewConf 祥见视频预览定义

UINT useSyncSem 1 – 用 semaphore 方式控制预览，需自行维护一个 semaphore，由 pPreviewConf-〉SyncSem 传入 sdk，sdk 在有传完一帧新的预览图像后会用该 semaphor 发消息给应用程序。

说明： 启动视频预览。按返回的 pPreviewConf 所指定的宽高创建 SDL 平面，并从 pPreviewConf 所传回的数据地址拷贝预览图像。

返回： 成功时返回 0，否则返回-1。

5.17 StopVideoPreview

int StopVideoPreview(int channelHandle)

参数： int channelInfo 通道句柄

说明： 停止视频预览。

返回： 成功时返回 0，否则返回-1。

设置及获取视频参数

5.18 SetVideoPara

int SetVideoPara(int channelHandle int Brightness, int Contrast, int Saturation, int Hue)

参数：	int channelHandle	通道句柄
	int Brightness,	亮度值（0—255）
	int Contrast,	对比度（0—127）
	int Saturation,	饱和度（0—127）
	int Hue	色调 （0—255）

说明： 设置视频参数。

返回： 成功时返回 0，否则返回-1。

5.19 GetVideoPara

int GetVideoPara(int channelHandle VideoStandard_t * VideoStandard, int *Brightness, int *Contrast, int *Saturation, int *Hue)

参数：	int channelHandle	窗口句柄
	VideoStandard_t * VideoStandard	视频标准（见 1.2）
	int *Brightness,	亮度指针值 （0—255）
	int *Contrast,	对比度指针值（0—127）
	int *Saturation,	饱和度指针值（0—127）
	int *Hue	色调指针值 （0—255）

说明： 得到视频参数。

返回： 成功时返回 0，否则返回-1。

获取SDK信息

5.20 GetSDKVersion

int GetSDKVersion(PVERSION_INFO versionInfo)

参数： PVERSION_INFO VersionInfo 指向 VERSION_INFO 的参数指针

说明： 获取当前使用的 SDK 版本号。

返回： 由 16 位（BCD 码）组成，高 8 位为主版本，低 8 位次版本号， 其后为 32 位的 BUILD 号，用于软件升级时标明该版本的最后修改时间。

设置及获取编码流类型

5.21 SetStreamType

int SetStreamType(int channelHandle, int StreamType)

参数： int channelHandle 通道句柄

int StreamType 流类型，见以下宏定义

宏定义：

```
# define     STREAM_TYPE_VIDEO     1        //视频流
# define     STREAM_TYPE_AUDIO     2         //音频流
# define     STREAM_TYPE_AVSYNC    3        //音视频同步流
```

说明： 设置流类型。

返回： 成功时返回 0，否则返回-1。

5.22 GetStreamType

int GetStreamType(int channelHandle int *StreamType);

参数： int channelHandle 通道句柄

int *StreamType 指向流类型指针

说明： 获得流类型。

返回： 成功时返回 0，否则返回-1。

5.23 SetSubStreamType

int SetSubStreamType (HANDLE hChannelHandle, int Type)

参数： int hChannelHandle, 通道句柄

int type 流类型

返回： 正确为 0，否则返回-1。

说明： 设置子通道流类型。

5.24 GetSubStreamType

int GetSubStreamType(int hChannelHandle, int*StreamType)

参数： int hChannelHandle 通道句柄

int*StreamType 指向流类型指针

返回： 正确为 0，否则返回-1。

说明： 获得子通道流类型。

启动及停止录像

5.25 StartVideoCapture

int StartVideoCapture(int channelHandle)

参数： int channelHandle 通道句柄

说明： 启动数据截取， SDK 向用户注册的流处理回调函数发送数据流，用户可以在流处理函数中处理数据。

返回： 成功时返回 0，否则返回-1。

5.26 StopVideoCapture

int StopVideoCapture(int channelHandle);

参数： int channelHandle 通道句柄

说明： 停止数据截取。

返回： 成功时返回 0，否则返回-1。

5.27 StartSubVideoCapture

int StartSubVideoCapture(int channelHandle)

参数： int channelHandle 通道句柄

说明： 启动某个通道的子通道的录像

返回： 成功时返回 0，否则返回-1。

5.28 StopSubVideoCapture

int StopSubVideoCapture(int channelHandle)

参数： int channelHandle 通道句柄

说明： 停止某个通道的子通道的录像

返回： 成功时返回 0，否则返回-1。

设置编码图像质量及编码帧结构、帧率

5.29 SetIBPMode

int SetIBPMode(int channelHandle int KeyFrameIntervals, int BFrames, int PFrames, int FrameRate);

参数：

int channelHandle	通道句柄
int KeyFrameIntervals,	关键帧间隔（默认值为 25）
int BFrames,	B 帧数（默认值为 2）
int PFrames,	P 帧数
int FrameRate	帧率（默认值为 25）

说明： 设置帧结构、关键帧间隔、B 帧数目、帧率，其中关键帧间隔不能小于 12，B 帧数目可以为 0、1 或 2，P 帧暂时设为无效，帧率范围 1-25(PAL)、1-30(NTSC)，可在运行时设定。

关键帧间隔解释：

关键帧为编码码流中采用帧内压缩的图像帧，其特点是图像清晰度好，但数据量大，通常作为帧间编码的原始参考帧。关键帧间隔是连续的帧间编码的帧个数，因 H264(MPEG4)编码是有损压缩，关键帧的个数会影响图像质量，因此关键帧的间隔需要合理设计。

返回： 成功时返回 0，否则返回-1。

注 意： 42 系列板卡不支持 B 帧编码，但是 B 帧参数仍需设置为 0 或者 2。

5.30 SetDefaultQuant

int SetDefaultQuant(int channelHandle int IQuantVal, int PQuantVal, int BQuantVal);

参数： int channelHandle 通道句柄
int IQuantVal, I 帧量化系数
int PQuantVal, P 帧量化系数
int BQuantVal B 帧量化系数

说明： 设置图像量化系数，用于调整图像质量，系数越低质量越好，取值范围（12-30），一般取值法为：取 I 帧和 P 帧一样大，B 帧比它们大 3 到 5，例如：15，15，20 和 18，18，23，这里系统默认值为 18，18，23；若要设定高清晰度图像，推荐值为：12 12 17 并可在运行时设定。

注： 量化系数解释：量化系数是强烈影响 MPEG 标准中编码图像质量和码率的参数，当量化系数越低，图像质量就会越高，码率也就越高，反之，图形质量就会越低，码率也就越低。

返回： 成功时返回 0，否则返回-1。

设置编码的分辨率格式

5.31 SetEncoderPictureFormat

int SetEncoderPictureFormat(int channelHandle PictureFormat_t pictureFormat);

参数： int channelHandle 通道句柄
PictureFormat_t pictureFormat 编码图象大小（4CIF、2CIF、DCIF、CIF 、QCIF）

说明： 设置当前主通道的编码格式。

返回： 成功时返回 0，否则返回-1。

注 意： 42 卡不支持 DCIF 编码

5.32 SetSubEncoderPictureFormat

int SetSubEncoderPictureFormat(int channelHandle, PictureFormat_t pictureFormat)

参数： int channelHandle 通道句柄
PictureFormat_t pictureFormat 编码图象大小（4CIF、2CIF、DCIF、CIF 、QCIF）

说明： 此函数是设置双编码模式时子通道的编码格式。

返回： 成功时返回 0，否则返回-1。

注 意： 42 卡子通道支持 CIF/QCIF 编码

设置码流及码流控制模式

5.33 SetupBitrateControl

int SetupBitrateControl(int channelHandle, int MaxBps)

参数： int channelHandle 通道句柄
int MaxBps 最大波特率（100000 以上）

说明： 可用来设置最大波特率，MaxBps 设为 0 时码率控制无效，当设置为某一最大波特率时，当编码码流将超过该值时，DSP 会自动调整编码参数来保持不超过最大波特率，当码流低于最大波特率时，DSP

不进行干涉，调整误差为<10%。

5.34 SetBitrateControlMode

int SetBitrateControlMode(int channelHandle BitrateControlType_t brc)

参数： int channelHandle 通道句柄

BitrateControlType_t brc 码率控制模式，分为 brCBR,恒定码率，brVBR 变码率。

说明： 配合 SetupBitrateControl 函数使用，但设定为 brVBR 同时又设置了指定的码率，那么编码系统会按照设置的码率作为上限来控制码率，就是说码率只会在该上限下工作，一般情况下码率会自动回落到最低的状态（由设定的图像质量和关键帧间隔决定），能最大地降低带宽和存储空间。设定为 brCBR 时，则按照指定的码率输出数据，不会自动回落到低码率的状态。

返回： 成功时返回 0，否则返回-1。

设置及获取视频信号制式、状况、视频信号输入位置调整

5.35 SetVideoStandard

int SetVideoStandard(int channelHandle VideoStandard_t VideoStandard)

参数： int channelHandle 通道句柄

VideoStandard_t VideoStandard 视频标准

说明： 设置视频标准，在某一制式的摄像头已经接好的情况下启动系统时可不必调用该函数，如果没有接摄像头的情况下启动系统然后再接 NTSC 制式的摄像头则必须调用该函数，或者中途调换不同制式的摄像头也必须调用该函数。

返回： 成功时返回 0，否则返回-1。

5.36 SetDefaultVideoStandard

int SetDefaultVideoStandard(VideoStandard_t VideoStandard)

参数： int channelHandle 通道句柄

VideoStandard_t VideoStandard 视频标准

说明： 设置系统默认的视频制式。默认为 PAL。系统中所有视频输入通道中如果无视频输入，则该通道会按照默认的制式处理。所有的视频输出通道的制式，在系统启动时也会按照默认的制式处理。

注：该函数只能在系统初始化（InitDSPs）之前运行，否则无效。

5.37 SetVideoDetectPrecision

int SetVideoDetectPrecision(int hChannelHandle,unsigned int value)

参数： int hChannelHandle 通道句柄

int value 灵敏度

说明： 设置视频信号检测的灵敏度。Value 范围：0－100。默认值为 20。如果视频信号的强度比较弱，或者信号通、断的切换比较频繁，为了避免“无视频信号”的提示影响图像，可以更改视频信号检测的灵敏度，值越大，精度越低，“无视频信号”出现的频率越低。

返回： 成功时返回 0，否则返回-1。

5.38 GetVideoSignal

int GetVideoSignal(int channelHandle)

参数： int channelHandle 通道句柄

说明： 获得接入视频信号情况，用于检测视频丢失报警。

返回： 返回 0 时表明视频信号正常，否则有错误。

5.39 SetInputVideoPosition

int SetInputVideoPosition(int hChannelHandle, unsigned int x, unsigned int y)

参数： int channelhandle 通道句柄

unsigned int x 坐标的 X 轴，缺省值为 8

unsigned int y 坐标的 Y 轴，缺省值为 2

说明： 设置视频输入位置，某些摄像头预览可能在左边出现黑边(x,y)为系统所处理图像的左上角在摄像机输入的原始图像中的坐标。x 必须为 2 的整数倍。(x,y)坐标的参数范围和摄像机的型号有关，如果指定的值和摄像机的输入不匹配，可能会导致图像静止或水平、垂直方向滚动。此函数请慎重调用。

返回： 成功时返回 0，否则返回-1。

注意： 42 卡不支持输入位置的调整。

设置OSD、LOGO及视频遮挡

5.40 SetOsdDisplayMode

int SetOsdDisplayMode(int hChannelHandle, int brightness, int translucent, int twinkInterval,unsigned short *format1, unsigned short *format2);

参数： int channelHandle 通道句柄

int Brightness OSD 显示亮度，255 最亮，0 最暗；42 系列板卡 OSD 亮度只支持黑（16）和白（235），亮度参数值大于 128 时为白色，小于 128 时为黑色。

unsigned short bTranslucent OSD 图像是否做半透明处理；

int TwinkleInterval 当值为 1 时，的亮度根据背景的亮度来调整，但背景较亮时，OSD 亮度自动调低，但背景较暗时，OSD 亮度自动调亮。原来的闪烁功能关闭；

unsigned char *Forma1, Format2 描述字符叠加的位置和次序的格式串，具体定义如下：

USHORT X， USHORT Y， CHAR0， CHAR1， CHAR2，… CHARN， NULL

其中 X，Y 是该字串在标准 CIF 图像的起始位置，X 必须是 8 的倍数，Y 可以在图象高度内取值即（0-287）PAL 、(0-239)NTSC；42 卡的 Y 值需要按照 16 对齐，如果取值未对齐会作自动调整；CHARN 也是 USHORT 型的参数，可以是 ASCII 码也可以是汉字，当想要显示当前时间时，可以指定为固定的时间定义值，其值如下：

_OSD_YEAR4	四位的年显示，如 2002
_OSD_YEAR2	两位的年显示，如 02
_OSD_MONTH3	英文的月显示，如 Jan
_OSD_MONTH2	两位阿拉伯数字的月显示，如 07
_OSD_DAY	两位的阿拉伯数字的日显示，如 31
_OSD_WEEK3	英文的星期显示，如 Tue
_OSD_CWEEK1	中文的星期显示，如星期二
_OSD_HOUR24	24 小时的时钟显示，如 18
_OSD_HOUR12	12 小时的时钟显示，如 AM09 或 PM09

_OSD_MINUTE　　两位分钟的显示
_OSD_SECOND　　两位秒的显示

在格式字符串的最后必须以 NULL（0）结尾，否则会显示错误的内容。字符串和时间显示可以在 FORMAT1 也可以在 FORAMT2，也可以混合在一起，但不得超过一行 4CIF 图像的宽度。

要显示位置在 16，19 的字符串“办公室”的格式字符串如下：

unsigned short　Format[] = {16, 19, ‘办’,’公’,’室’, ‘\0’};

要显示位置在 8， 3 的时间字符串可以如下：

unsigned short Format[]={8, 3, _*OSD*_YEAR4, ‘ 年 ’,_OSD_MONTH2,’ 月 ’,_OSD_DAY, ‘日’,_OSD_HOUR24,’:’, _*OSD*_MINUTE, ‘:’, _OSD_SECOND, ‘\0’};

如果只想显示其中一行，则将起始的字符串定义如下：

unsigned short FormatNoDisplay[]={0, 0, ‘\0’};

5.41 SetOsd

int SetOsd(int channelHandle, int Enable);

参数 int channelHandle　　通道句柄
　　int Enable　　使能(0/1)

说明：　设置 OSD 显示，在当前的系统时间年月日星期时分秒叠加在预览视频之上，并可作透明处理。

返回：　成功时返回 0，否则返回-1。

5.42 SetupDateTime

int SetupDateTime(int channelhandle, SYSTEMTIME *now)

参数：　int　channelHandle　　通道句柄
　　SYSTEMTIME *　now　　需要设置的时间指针值，如为 NULL，即本地校时。

说明：　这个功能在 ver4.0 版本已无效。

返回：　成功时返回 0，否则返回-1。

5.43 SetOsdDisplayModeEx

int SetOsdDisplayModeEx(int hchannelHandle, int brightness, int Translucent, int Param, int nLineCount, unsigned short **Format)

参数：　int　hChannelHandle　通道句柄
　　int　brightness　OSD 显示高度，255 最亮，0 最暗。42 系列板卡 OSD 亮度只支持黑（16）和白（235），亮度参数值大于 128 时为白色，小于 128 时为黑色。
　　BOOL　Translucent　OSD 图象是否做半透明处理；
　　int　param　Bit 0：　是否自动进行颜色翻转；
　　Bit 16-23：　垂直放大倍数
　　Bit 24-31：　水平放大倍数

在 OSD 字符中，ASCII 字符的标准分辨率为 8×16，汉字的标准分辨率为 16×16。

由于在编码之前需要对原始图像进行缩小才能产生编码所需的分辨率，此时就需要先把 OSD 字符放大以后再叠加在 4CIF 的原始图像上，这样才能保证在缩小后的编码图像上能够看清 OSD 字符。

如果不指定放大倍数（默认设置），则系统会根据该通道录像的分辨率自动设置，这样在任何分辨率下都可以保证回放时能够看清 OSD 内容，但是这会导致 OSD 的大小和位置在原始图像中不固定。

为了避免上面的现象，用户可以指定 OSD 的大小。例如，如果应用程序想以 CIF、DCIF、2CIF、4CIF 的分辨率录像，这时候可以将放大系数设为 2、2，此时 OSD 的位置始终固定，但在不同的编码分辨率下，OSD 字符的分辨率也不同，所以需要特别注意。如果此时使用 QCIF 录像，则 OSD 字符会变得模糊不清（因

为 QCIF 要对图像进行 1/4 缩小，而对 OSD 字符只进行了 2 倍的放大)。具体配置详见下表：

水平放大倍数	垂直放大倍数	适合的录像分辨率	说明
1	1	4CIF	其它分辨率下会模糊
1	2	2CIF	小于 2CIF 时无法分辨
2	2	CIF、DCIF	QCIF 时无法分辨
4	4	QCIF	在其它分辨率下字符会很大
任意系数为 0		自动设置（默认值）	
其它无效值		按水平 2、垂直 2 处理	

注意：　因为字符的位置会随着不同的录像分辨率而改变，在位置改变后，某些 OSD 字符的位置可能会超出图像的范围，此时这些字符将无法显示，但系统并不会返回错误。

int　　nLineCount　　OSD 显示的行数，最多为 8

unsigned short　　**Format　　多行 OSD 字符串显示，描述字符叠加的位置和次序的格式串，其中每一行的第一元素 X 和第二元素 Y 是该字串在标准 4CIF 图象的起始位置，X 必须是 16 的倍数，Y 可以在图象高度内取值即（0-575）PAL 、（0-479）NTSC,，42 卡的 Y 值需要按照 16 对齐，如果取值未对齐会作自动调整；可以是 ASCII 码也可以是汉字，当想要显示当前时间时，可以指定为固定的时间定义值，其值如下：

_OSD_YEAR4　　四位的年显示，如 2002
_OSD_YEAR2　　两位的年显示，如 02
_OSD_MONTH3　　英文的月显示，如 Jan
_OSD_MONTH2　　两位阿拉伯数字的月显示，如 07
_OSD_DAY　　两位的阿拉伯数字的日显示，如 31
_OSD_WEEK3　　英文的星期显示，如 Tue
_OSD_CWEEK1　　中文的星期显示，如星期二
_OSD_HOUR24　　24 小时的时钟显示，如 18
_OSD_HOUR12　　12 小时的时钟显示，如 AM09 或 PM09
_OSD_MINUTE　　两位分钟的显示
_OSD_SECOND　　两位秒的显示

在格式字符串的最后必须以 NULL（0）结尾，否则会显示错误的内容。

返回：　正确为 0，否则返回-1。

说明：　此函数为 SetOsdDisplayMode 的扩展，SetOsdDisplayModeEx 函数支持最多 8 行 OSD 字符串的示。

5.44 LoadYUVFromBmpFile

int LoadYUVFromBmpFile(char *FileName, unsigned char *yuv, int BufLen, int *Width, int *Height)

参数：　char *FileName　　文件名
unsigned char *yuv　　YUV422 格式的图像指针
int BufLen　　yuv 缓存大小
int *Width　　返回 yuv 图像的宽度
int *Height　　返回 yuv 图像的高度

说明：　把 24 位 bmp 文件转成 yuv 格式的数据，其中 BMP 位图的长宽须为 16 的倍数，最大支持 128*128 像素。

返回：　成功时返回 0，否则返回-1。

5.45 SetLogoDisplayMode

int SetLogoDisplayMode(int channelHandle unsigned short ColorKeyR, unsigned short ColorKeyG, unsigned short ColorKeyB, unsigned short bTranslucent, int TwinkleInterval)

参数： int channelHandle 通道句柄

unsigned short ColorKeyR, ColorKeyG, ColorKeyB

LOGO 图像该颜色将在显示时完全透明

unsigned short bTranslucent LOGO 图像是否做半透明处理

int TwinkleInterval 闪烁的时间设置，由 16 进制数表示为 0xXXYY，其中 XX 为显示的秒数，YY 为停止显示的秒数，XXYY 均为 0 时正常显示

说明： 可通过该函数来设置 LOGO 的显示模式。

返回： 成功时返回 0，否则返回-1。

5.46 SetLogo

int SetLogo(int channelHandle int x, int y, int w, int h, unsigned char *yuv)

参数： int channelHandle 通道句柄

int x, 左上角位置 x(0-703)

int y, 左上角位置 y(0-575)

int w, 宽度(0-128) (此宽度必须与原始 Logo 图像宽度一致)

int h, 高度(0-128)

unsigned char *yuv YUV422 格式的图像指针

说明： 设置 logo 屏幕图像位置及数据；用户程序可调用 LoadYUVFromBmpFile 从 24 位 bmp 文件中获取 YUV 数据（见 4.44）,透明处理由 DSP 完成。

返回： 成功时返回 0，否则返回-1。

5.47 StopLogo

int StopLogo(int channelHandle)

参数： int channelHandle 通道句柄

说明： 停止 LOGO 显示；

返回： 成功时返回 0，否则返回-1。

5.48 SetupMask

int SetupMask(int channelHandle, RECT *rectList, int iAreas)

参数： int channelHandle 通道句柄

RECT * rectList 矩形框数组

int iAreas, 矩形框个数

说明： DS40xxHC 提供的启动屏幕遮挡函数，最多可设置 3 2 个，矩形框的范围是(0,0,703,575)，矩形框的宽度为 16 对齐，高度为 8 对齐

返回： 成功时返回 0，否则返回-1。

5.49 StopMask

int StopMask(int channelHandle)

参数： int channelHandle 通道句柄

说明： DS40xxHC 提供的停止屏幕遮挡函数。

返回： 成功时返回 0，否则返回-1。

移动侦测

5.50 AdjustMotionDetectPrecision

int AdjustMotionDetectPrecision(int channelHandle int iGrade, int iFastMotionDetectFps, int iSlowMotionDetectFps)

参数： int channelHandle 通道句柄

int iGrade 运动分析灵敏度等级（0-6），（默认值为 2）

int iFastMotionDetectFps, 高速运动检测的帧间隔，取值范围 0-12，0 表示不作高速运动检测，通常值为 2，（默认值为 2）

int iSlowMotionDetectFps 低速运动检测，适合于慢速运动情况，通常取值范围 13 以上，当取值为 0 时不作低速运动检测。（默认值为 200）

iFastMotionDetectFps、 iSlowMotionDetectFps 两个参数同时为 0 的时候不做移动侦测 ，如果 iFastMotionDetectFps 取值为 0 则默认为 1，如果 iSlowMotionDetectFps 取值小于 iFastMotionDetectFps 则默认为 15

说明： 调整运动分析灵敏度（iGrade），可于编码时动态调整运动侦测的灵敏度，决定 DSP 全局运动分析的灵敏度，与 MotionAnalyzer 的 iThreshold 不同，后者主要用于主机分析指定区域的运动统计结果，等级 0 最灵敏，6 最迟钝；推荐值为 2。

将运动分析灵敏度等级参数 iGrade 和 0x80000000 做“或”操作，会对移动侦测启用自适应分析。（自适应分析不适用于 42 系列板卡）

返回： 成功时返回 0，否则返回-1。

5.51 SetupMotionDetection

int SetupMotionDetection(int hChannelHandle, RECT *RectList, int iAreas)

参数： int hChanelHandle 通道句柄

RECT *RectList 矩形框数组

int iAreas, 矩形框个数(最大个数为 100)

说明： 设置运动检测区域，当收到运动信息的数据帧（PktMotionDetection）时,调用 MotionAnalyzer, MotionAnalyzer 会根据在 SetupMotionDetection 中的设置来分析每个需要检测的区域，当某区域的阀值(MotionAnalyzer 的 iThreshold)到达时，会在返回的区域数组（MotionAnalyzer 的 iResult）标明最后的判断；矩形框范围是（0，0，703，575）

返回： 成功时返回 0，否则返回-1。

5.52 StartMotionDetection

int StartMotionDetection(int channelHandle)

参数： int channelHandle 通道句柄

说明： 启动运动检测，运动检测信息会通过数据流传送，用户程序发现是 PktMotionDetection 帧类型时，可调用 MotionAnalyzer 来处理运动信息，结果由 MotionAnalyzer 在 iResult 中返回。也可以按照 SDK 提供的数据格式来自己分析，运动信息格式参见 2.5 节。注意：运动检测与编码相互独立，用户程序可在不启动编码的情况下进行运动检测。

返回： 成功时返回 0，否则返回-1。

5.53 StopMotionDetection

int StopMotionDetection(int channelHandle)

参数： int channelHandle 通道句柄

说明： 停止运动检测。

返回： 成功时返回 0，否则返回-1。

5.54 MotionAnalyzer

int MotionAnalyzer(int channelHandle char *MotionData, int iThreshold， int *iResult)

参数： int channelHandle 通道句柄

char *MotionData 运动矢量指针

int iThreshold 判断运动的一个区域阀值（0-100）

int *iResult 按照区域阀值指定的强度分析后的结果，是一个数组，大小在 SetupMotionDetection 的 numberOfAreas 指定，如果某数组单元的值大于零则表明有该区域有该值表明的运动强度

说明： 动态监测分析，运动检测由 DSP 完成，DSP 送出的 IPktMotionData 帧就是已经分析好的运动信息，区域的运动分析由主机完成，数据源由码流中的 PktMotionData 帧提供，结果在 iResult 中说明。用户软件可使用由码流提供的运动强度信息来自己分析或调用该函数来进行区域分析，运动强度数据结构在 2.5 说明。

返回： 成功时返回 0，否则返回-1。

5.55 SetupMotionDetectionEx

int SetupMotionDetectionEx(int hChannelHandle,int iGrade,int iFastMotionDetectFps,int iSlowMotionDetectFps, UINT delay,RECT *RectList, int iAreas,MOTION_DETECTION_CALLBACK MotionDetectionCallback,int reserved)

参数： int hChannelHandle 通道句柄

int iGrade 运动分析灵敏度等级（0-6）

int iFastMotionDetecFps 高速运动检测的帧间隔，取值范围 0-12，0 表示不作高速运动检测，通常值为 2，

int iSlowMotionDetectFps 速运动检测，适合于慢速运动情况，通常取值范围 13 以上，当取值为 0 时不作低速运动检测。

iFastMotionDetectFps、 iSlowMotionDetectFps 两个参数同时为 0 的时候不做移动侦测 ，如果 iFastMotionDetectFps 取值为 0 则默认为 1，如果 iSlowMotionDetectFps 取值小于 iFastMotionDetectFps 则默认为 15

UINT delay 上一次产生移动侦测后的延时时间

RECT *RectList 矩形框数组

int iAreas 矩形框个数

MOTION_DETECTION_CALLBACK MotionDetectionCallback 回调函数,

int reserved 保留参数

回调函数说明

MotionDetectionCallback (ULONG channelNumber,

Int bMotionDetected,

Void *context)

参数： ULONG channelNumber 通道号；

int bMotionDetected 移动侦测标志，若当前通道所设置的移动侦测区域内产生移动侦测，则 bMotionDetected 被标志为 1，若当前通道检测到当前设置区域自上一次产生移动侦测以后的 delay 秒时间之内没有产生移动侦测，则 bMotionDetected 被标志为 0

void *context 设备上下文

参数详细说明：

int iGrade 运动分析灵敏度等级（0-6），可于编码时动态调整运动侦测的灵敏度，决定 DSP 全局运动分析的灵敏度，等级 0 最灵敏，6 最迟钝；推荐值为 2。将运动分析灵敏度等级参数 iGrade 和 0x80000000 做“或”操作，会对移动侦测启用自适应分析。（自适应分析不适用于 42 系列板卡）

UINT delay 画面静止之后的延时时间，单位为秒，若在该延时时间内没有产生移动侦测，则将回调函数 MotionDetectionCallback 之中的参数 bMotionDetected 标志为 0，若在该延时时间之内，在当前所设置的区域内产生移动侦测，则 bMotionDetected 被标志为 1，并且在产生移动侦测之后的 delay 时间内，DSP 不会对在此时间段之内的视频帧进行移动侦测分析，因此 DSP 和主机都省却了在此时间段对产生的视频运动进行频繁判断和分析。直至超过了此 delay 秒延时时间，DSP 才会对此时刻的视频进行判断，若产生了移动侦测，则回调函数中的 bMotionDetected 被再次标志为 1，否则标志为 0。

说明：设置运动检测的扩展。

返回：成功时返回 0，否则返回-1。

设置现场音频监听及获取现场声音音量幅度

5.56 SetAudioPreview

int SetAudioPreview(int channelHandle int bEnable)

参数： int channelHandle 通道句柄

int bEnable 使能(0/1)

说明： 设置音频预览，打开或关闭，系统只能有一路打开。

注意： 此函数接口只对 40xx 系列板卡有效，41xx 系列板卡的音频预览功能请使用 SetPCIAudioPreview 函数，当 41xx 系列板卡与 40xx 系列板卡混插时，可通过 GetBoardInfo 函数来判断音频通道所属卡的类型。然后各自调用之前提到的两个函数接口。

返回： 成功时返回 0，否则返回-1。

5.57 GetSoundLevel

int GetSoundLevel(int channelHandle)

参数： int channelHandle 通道句柄

说明： 获取当前通道的现场声音幅度，注意当无声音输入时因背景躁声的原因返回值并不为 0。

返回： 成功时返回声音幅度，失败返回-1。

启动及停止获取原始图像数据流

5.58 RegisterImageStreamCallback

int RegisterImageStreamCallback(IMAGE_STREAM_CALLBACK pFunc, void* context)

参数： IMAGE_STREAM_CALLBACK 回调函数

回调函数说明

(*IMAGE_STREAM_CALLBACK)(UINT channelNumber ,void *context)

UINT channelNumber 通道号

void *context 调用回调函数时提供的上下文

说明： 注册用户获取原始图像数据流的函数，用户可以获取实时的 YUV420 格式的预览数据。

返回： 成功时返回声音幅度，失败返回-1。

5.59 SetImageStream

int SetImageStream(int hChannelHandle, int bStart, unsigned int fps, unsigned width, unsigned height, unsigned char* imageBuf)

参数：

int	hChannelHandle		通道句柄
int	bStart		1:启动捕获，0:停止捕获
unsigned int	fps		帧率
unsigned int	width		图象的宽度(必须是 4CIF 宽度的 1/8，1/4，1/2 或原始大小 704)
unsigned int	height		图象的高度(必须是 4CIF 高度的 1/8，1/4，1/2 或原始大小 576)
char	*imageBuffer		抓图后存数据的地址

说明： 用户通过这个函数启动或停止获取原始图象数据流，此函数依赖主机的处理速度。

返回： 成功时返回声音幅度，失败返回-1。

注意： 42 系列板卡不支持原始数据流捕获。

抓图及图像保存函数

5.60 GetOriginalImage

int GetOriginalImage(int channelHandle char *ImageBuf, int *Size);

参数：

int	channelHandle	通道句柄
char	*ImageBuf	原始图像指针
int	*Size	原始图像的大小（注：调用前是 imagebuf 的大小，调用后是实际图像所使用的字节数）

说明： 获得原始图像，DS40xxHC/HF 原始图像是标准的 4CIF 图像格式(包括 QCIF 编码)，用户程序可调用 SaveYUVToBmpFile 来生成 24 位的 bmp 文件。

返回： 成功时返回 0，否则返回-1。

注 意：

DS-42xxHC 卡的原始图像是 CIF 图像格式；DS-42xxHFV 卡的原始图像的 4CIF 图像格式；

DS041xxHCV 卡的原始图像是 4CIF 图像格式；

DS-40xxHS 的原始图像是 CIF 图像格式；

DS-40xxHC/HC+/HF/HCS 的原始图像是 4CIF 图像格式(包括 QCIF 编码)；

DS-40xxH 的原始图像是 CIF 图象格式。

5.61 SaveYUVToBmpFile

int SaveYUVToBmpFile(char *FileName, unsigned char *yuv, int Width, int Height);

参数： char *FileName 文件名
unsigned char *yuv YUV422 格式的图像指针
int Width yuv 图像的宽度
int Height yuv 图像的高度

说明： 把 yuv 图像转成 bmp 文件，如果是 DS40xxHC/HF 卡抓图则 Width 为 704，Height 为 576(PAL)或 480(NTSC)，如果是 DS400xH 卡抓图则 Width 可能为 352 或 176，Height 为 288、240、144 或 120，要根据缓冲区的大小来判断。

返回： 成功时返回 0，否则返回-1。

5.62 GetJpegImage

int GetJpegImage(int hChannelHandle,unsigned char *ImageBuf,unsigned long *Size,unsigned int nQuality)

参数： int channelHandle 通道句柄
unsigned char* ImageBuf 图像指针
unsigned long *Size 图像的大小（注：调用前是 imagebuf 的大小，调用后是实际图像所使用的字节数）
unsigned int nQuality JPEG 图片质量(1 到 100，1 最差，100 最好)

说明： 抓取 JPEG 格式图像。

返回： 成功时返回 0，否则返回-1。

注 意：

DS-42xxHC 卡的 JPEG 抓图是 CIF 图像格式；DS-42xxHFV 卡的 JPEG 抓图像的 4CIF 图像格式。
DS041xxHCV 卡的 JPEG 抓图是 4CIF 图像格式；
DS-40xxHC/HC+/HF/HCS 的 JPEG 抓图是 4CIF 图像格式(包括 QCIF 编码)；
DS-40xxHS 的 JPEG 抓图是 CIF 图像格式；
DS-40xxH 的 JPEG 抓图是 CIF 图象格式。

双编码

5.63 SetupSubChannel

int SetupSubChannel(int channelHandle int iSubChannel);

参数： int channelHandle 通道句柄
int iSubChannel 子通道号

说明： 配合双编码模式使用。当设置某个通道为双编码模式时，如主通道编码 CIF，子通道编码 QCIF，这时可对主/子通道分别设置某些参数。OSD、LOGO 等参数对主/子通道是一样的；在关键帧间隔、设置帧率、量化系数、变码流/定码流模式、码流大小等参数时应调用此函数分别对主/子通道进行设置，缺省是对主通道进行设置

双编码功能说明：

对一路视频图像进行两路视频编码，两路视频可以有不同的码流类型、不同分辨率、不同码率等。3.0 版本对双编码功能做了增强，子通道的所有参数都可以任意设置。双编码中主通道和子通道唯一的区别在于：子通道占用的系统资源比主通道少，优先级比主通道低。当系统忙时，会尽量保证主通道编码，并先从子通道开始丢帧。由于占用资源少，因此可以利用子通道来实现多路高分辨率的非实时编码。例如：可以把 DS4004HC 中的 4 个子通道全部设置为 4CIF 分辨率（SetSubStreamType），而不使用主通道编码，这样就可以实现 4 路 4CIF 编码。在一般场景下，每路图像都可以达到 15 帧以上。

返回： 成功时返回 0，否则返回-1。

5.64 GetSubChannelStreamType

int GetSubChannelStreamType(void *DataBuf, int FrameType);

参数： void *DataBuf 输入数据缓存区

int FrameType 输入帧类型

说明： 配合双编码模式使用，当设置双编码模式时，子通道 0 编码 CIF，子通道 1 编码 QCIF，启动录像后，DSP 会向上送两种数据流，CIF 和 QCIF，调用此函数得到数据流类型，供应用程序使用。这个函数的目的只是为了向下兼容，实际上你可以通过参数 FrameType 自行判断。

返回： 0 其他数据

1 主通道数据流的文件头

2 子通道数据流的文件头

3 主通道数据流的视频帧类型

4 子通道数据流的视频帧类型

5 数据流的音频帧

获取帧统计信息

5.65 GetFramesStatistics

int GetFramesStatistics(int hChannelHandle, PFRAMES_STATISTICS framesStatistics)

参数： int channelHandle 通道句柄

PFRAMES_STATISTICS framesStatistics 帧统计信息（见 2.2 节）

说明： 获取帧统计信息。

返回： 成功时返回 0，否则返回-1。

强制设定I帧

5.66 CaptureIFrame

int CaptureIFrame(int channelHandle)

参数： int channelHandle 通道句柄

说明： 将目前编码帧强制设定为 I 帧模式，可从码流中提取该帧单独用于网络传送。

返回： 成功时返回 0，否则返回-1。

设置反隔行变换及强度

5.67 SetDeInterlace

int SetDeInterlace(int hChannelHandle,UINT mode,UINT level)

参数： int channelHandle通道句柄

UINT mode： 0:该通道不进行反隔行变换,此时 level 参数无效；
1：使用默认算法（系统默认值）。42 卡不支持 mode=1

UINT Level： 0－10，反隔行变换的强度逐渐加强，0 最弱，对图像的损失最小，10 最强，对图像的损失最大。默认值为 5。

说明： 是否执行反隔行变换，以及反隔行变换的强度

返回： 成功时返回 0，否则返回-1。

复位DSP

5.68 ResetDSP

int ResetDSP(int DspNumber)

参数： int DspNumber DSP 的索引号

说明： 现无效

设置看门狗

5.69 SetWatchDog

int SetWatchDog(unsigned int boardNumber,int bEnable)

参数： int boardnumber board 的索引号

说明： 设置看门狗。DS-4016HCS 提供 4pin 复位接口，用户需要把主机机箱的 Reset 接线连接到板卡上的 2pin 复位接口，板卡上的另外 2pin 接口连接到主板的 Reset，这样就可以实现对上层软件和系统中所有压缩板卡的运行状态监控。

返回： 成功时返回 0，否则返回-1。

（注：其他未说明函数为专用卡用户使用，海康威视提供其他文档说明。）

V4.2 New add

5.70 StartVideoPreviewEx

int StartVideoPreviewEx(int hChannelHandle, PREVIEWCONF* pPreviewConf, UINT useSyncSem, UINT mode)

参数： hChannelHandle, pPreviewConf, useSyncSem 等参数的设置请参考函数 StarrtVideoPreview 说明。

Mode 可取以下值：NORMAL SIZE，D1_SIZE，DCIF_SIZE，QCIF_SIZE，MINI_SIZE，

其具体定义如下：

```
#define NORMAL SIZE     0    /* just not change */
#define D1_SIZE         1    /* 704 X 576 */
#define DCIF_SIZE       2    /* 528 X 288 */
#define CIF_SIZE        3    /* 352 X 288 */
#define QCIF_SIZE       4    /* 176 X 144 */
#define MINI_SIZE       5    /* 88 X 72 */
```

其中当 mode 值为 NORMAL_SIZE 时，功能和 StartVideoPreview 一致，即图象在内存中长宽根据实际

显示区域决定，而选用其他参数时，图像在内存的大小不变. 需要注意的是所有通道中内存中显示宽度为 D1_SIZE 的通道必须小于 4. 同时为了缓解 PCI 总线压力，应用程序会降低图像质量来减小所传输的数据大小。

6 解码卡API

MD 卡函数说明

如无其他说明，函数返回值统一为 0 表示函数调用成功返回； -1 表示函数调用失败返回，具体失败原因请调用 GetLastErrorNum（）得到错误代号，比对头文件了解。

6.1 HW_InitDecDevice

int HW_InitDecDevice(long *pDeviceTotal)

说　　明：　初始化设备。

输出参数：　pDeviceTotal 初始化成功的解码路数。

6.2 HW_ReleaseDecDevice

int HW_ReleaseDecDevice()

说　　明：　关闭设备，应在程序退出时调用.

6.3 HW_ChannelOpen

int HW_ChannelOpen(long nChannelNum, int* phChannel)

说　　明：　打开通道，获取相关的操作句柄，与通道相关的操作必须使用该句柄。

输入参数：　nChannelNum　　通道号（从 0 开始）

输出参数：　phChannel　　操作句柄

6.4 HW_ChannelClose

int HW_ChannelClose(int hChannel)

说　　明：　关闭通道，释放相关资源。

输入参数：　hChannel　　通道句柄

6.5 HW_OpenStream

int HW_OpenStream(int hChannel, char* pFileHead, int nHeadSize)

说　　明：　打开流接口（类似于打开文件）。

输入参数：　**hChannel**　　通道句柄

　　pFileHead　　文件头数据

　　nHeadSize　　文件头长度

6.6 HW_CloseStream

int HW_CloseStream(int hChannel)

说　　明：　关闭数据流。

输入参数：　hChannel　　通道句柄

6.7 HW_InputData

int HW_InputData(int hChannel,char* pBuf, int nSize)

说　　明：　输入数据，打开流之后才能输入数据。
输入参数：　hChannel　　通道句柄
　　pBuf　　缓冲区地址
　　nSize　　缓冲区大小
返 回 值：　成功返回 nsize，失败返回-1。

6.8 HW_OpenFile

int HW_OpenFile(int hChannel,char* sFileName)
说　　明：　打开文件。
输入参数：　hChannel　　通道句柄
　　sFileName　　文件名

6.9 HW_CloseFile

int HW_CloseFile(int hChannel)
说　　明：　关闭文件。
输入参数：　hChannel

6.10 HW_Play

int HW_Play(int hChannel)
说　　明：　播放开始。
输入参数：　hChannel　　通道句柄

6.11 HW_Stop

int HW_Stop(int hChannel)
说　　明：　播放结束
输入参数：　hChannel　　通道句柄

6.12 HW_Pause

int HW_Pause(int hChannel , ULONG bPause)
说　　明：　播放暂停。
输入参数：　hChannel　　通道句
　　bPause　　1 暂停，0 继续

6.13 HW_PlaySound

int HW_PlaySound(int hChannel)
说　　明：　打开相应通道的声音，默认情况下声音是关闭的。
输入参数：　hChannel　　通道句柄

6.14 HW_StopSound

int HW_StopSound(int hChannel)
说　　明：　关闭声音。
输入参数：　hChannel　　通道句柄

6.15 HW_SetVolume

int HW_SetVolume(int hChannel, ULONG nVlome)

说　　明：　调节音量。

输入参数：　hChannel　　通道句柄

　　nVlome　　音量大小。0~0xffff。

6.16 HW_StartCapFile

int HW_StartCapFile(int hChannel,char* sFileName)

说　　明：　捕获当前的码流，保存到另外的文件中。

输入参数：　hChannel　　通道句柄

　　sFileName　　文件名。

6.17 HW_StopCapFile

int HW_StopCapFile(int hChannel)

说　　明：　停止捕获码流。

输入参数：　hChannel　　通道句柄

6.18 HW_GetPictureSize

int HW_GetPictureSize(int hChannel,ULONG* pWidth, ULONG* pHeight)

说　　明：　获取当前码流中的原始图像尺寸。

输入参数：　hChannel　　通道句柄

输出参数：　*pWidth　　图像宽

　　*pHeight　　图像高

6.19 HW_GetYV12Image

int HW_GetYV12Image(int hChannel, char* pBuffer, ULONG nSize)

说　　明：　捕获当前显示的图像,YV12 格式。

输入参数：　hChannel　　通道句柄

　　pBuffer　　保存图像的缓冲区，大小应该大于或等于（*pWidth）*（*pHeight）*3/2，YV12 格式每个像素占用 3/2 个 BYTE。

　　nSize　　输入缓冲区 pBuffer 的大小。

6.20　HW_ConvertToBmpFile

int HW_ConvertToBmpFile(char * pBuf, ULONG nSize,ULONG nWidth, ULONG nHeight, char *sFileName, ULONG nReserved)

说　　明：　把 YV12 格式图像转换成位图。

输入参数：　pBuf　　YV12 格式的图像缓冲

　　nSize　　输入缓冲 pBuf 的大小

　　nWidth　　图像的宽

　　nHeight　　图像的高

　　sFileName　　BMP 文件的文件名

　　nReserved　　保留

6.21 HW_GetSpeed

int HW_GetSpeed(int hChannel, long *pSpeed)

说　　明：　得到播放速度。

输入参数：　hChannel　　通道句柄

输出参数：　*pSpeed　　播放速度；范围-4 ~ 4

6.22 HW_SetSpeed

int HW_SetSpeed(int hChannel,long nSpeed)

说　　明：　设置播放速度；-4 时停止播放，调用 HW_Pause(hChannel,0)可单帧播放，每调用一次播放一帧。其他值则是-3 表示 1/8 速，-2 表示 1/4 速，-1 表示 1/2 速，0 表示正常播放，1 表示 2 倍速，2 表示 4 倍速，3 表示 8 倍速，4 表示最大倍速。

输入参数：　hChannel　　通道句柄

　　nSpeed　　播放速度

6.23 HW_SetPlayPos

int HW_SetPlayPos(int hChannel,ULONG nPos)

说　　明：　设置播放位置。

输入参数：　hChannel　　通道句柄

　　nPos　　0 ~ 100 文件位置的百分数

6.24 HW_GetPlayPos

int HW_GetPlayPos(int hChannel,ULONG* pPos)

说　　明：　获取播放位置。

输入参数：　hChannel　　通道句柄

输出参数：　*pPos　　0 ~ 100 文件位置的百分数

6.25 HW_SetJumpInterval

int HW_SetJumpInterval(int hChannel,ULONG nSecond)

说　　明：　设置跳跃时间间隔。

输入参数：　hChannel　　通道句柄

　　nSecond　　跳跃时间间隔，单位秒

6.26 HW_Jump

int HW_Jump(int hChannel,ULONG nDirection)

说　　明：　前跳或后跳。

输入参数：　hChannel　　通道句柄

　　nDirection　　文件跳跃方向，JUMP_FORWARD 向前跳跃；

　　JUMP_BACKWARD 向后跳跃。

6.27 HW_GetVersion

int HW_GetVersion(PHW_VERSION pVersion)

说　　明：　获取版本信息。

输出参数：　pVersion　版本信息，详见下

结构说明：

```
typedef struct {
    ULONG DspVersion, DspBuildNum;
    ULONG DriverVersion, DriverBuildNum;
    ULONG SDKVersion, SDKBuildNum;
}HW_VERSION, *PHW_VERSION;
```

DspVersion, DspBuildNum	DSP 程序的版本号和 Build 号
DriverVersion, DriverBuildNum	驱动程序的版本号和 Build 号
SDKVersion, SDKBuildNum	SDK 的版本号和 Build 号

6.28 HW_GetCurrentFrameRate

int HW_GetCurrentFrameRate(int hChannel,ULONG* pFrameRate)

说　　明：	获得当前播放的帧率。	
输入参数：	hChannel	通道句柄
输出参数：	* pFrameRate	帧率

6.29 HW_GetCurrentFrameNum

int HW_GetCurrentFrameNum(int hChannel,ULONG* pFrameNum)

说　　明：	获得当前播放的帧序号。	
输入参数：	hChannel	通道句柄
输出参数：	*pFrameNum	帧序号

6.30 HW_GetFileTotalFrames

int HW_GetFileTotalFrames(int hChannel,ULONG* pTotalFrames)

说　　明：	获取文件总帧数；	
输入参数：	hChannel	通道句柄
输出参数：	* pTotalFrames	总帧数

6.31 HW_GetFileTime

int HW_GetFileTime(int hChannel, ULONG* pFileTime)

说　　明：	获取文件总时间；	
输入参数：	hChannel	通道句柄
输出参数：	* pFileTime	总时间，单位毫秒（ms）

6.32 HW_GetCurrentFrameTime

int HW_GetCurrentFrameTime(int hChannel,ULONG* pFrameTime);

说　　明：	获取当前播放帧的时间。	
输入参数：	hChannel	通道句柄
输出参数：	* pFrameTime	时间，单位毫秒（ms）

6.33 HW_GetPlayedFrames

int HW_GetPlayedFrames(int hChannel,ULONG *pDecVFrames)

说　　明：	获取已经解码的视频帧数。	
输入参数：	hChannel	通道句柄
输出参数：	*pDecVFrames	已经解码的帧数。

6.34 HW_SetFileEndMsg

int HW_SetFileEndMsg(int hChannel, sem_t* nMsg)

说　　明：　注册一个自定义的消息，当文件结束时被 post。

输入参数：　hChannel　　通道句柄

　　nMsg　　自定义的 semaphore

6.35 HW_SetStreamOpenMode

int HW_SetStreamOpenMode(int hChannel,ULONG nMode)

说　　明：　设置流的调节参数。

输入参数：　hChannel　　通道句柄

　　nMode　　0 ~ 5 , 0 不做任何调节，适合播放文件流（非实时流），1 ~ 5 调节实时流的流畅性和延迟性，值越大表示越流畅但延迟越大。

6.36 HW_GetStreamOpenMode

int HW_GetStreamOpenMode(int hChannel, ULONG *pMode)

说　明：获取当前流的调节参数。

输入参数：　hChannel　　通道句柄

输出参数：　*pMode　　0 ~ 5

6.37 HW_SetAudioPreview

int HW_SetAudioPreview(int hChannel,　UINT bEnable)

说　　明：　声音预览，板卡接线方法和 DS-40xxHC 系列板卡相同。在同一时间只能打开一个通道的声音预览，会自动关闭原来已经打开的声音预览。注意：要使声音预览起作用必须在此之前打开所有的通道（调用 HW_ChannelOpen），而且必须打开此通道的声音（调用 HW_PlaySound）。

输入参数：　bEnable　　如果是 1，打开声音预览，否则关闭声音预览

6.38 HW_StartDecVgaDisplay

int HW_StartDecVgaDisplay(int hChannel, PREVIEWCONFIG* pPreviewConf,UINT useSyncSem)

说明：　开始主机解码视频;按返回的 pPreviewConf 所指定的宽高创建 SDL 平面，并从 pPreviewConf 所传回的数据地址拷贝预览图像。

输入参数：　PREVIEWCONFIG* pPreviewConf　　祥见 2.5 视频预览定义

　　UINT　useSyncSem　1 – 用 semaphore 方式控制视频，需自行维护

一个 semaphore，由 pPreviewConf-〉SyncSem 传入 sdk，sdk 在有传完一帧新的解码图像后会用该 semaphore 发消息给应用程序。

6.39 HW_StopDecChanVgaDisplay

int HW_StopDecChanVgaDisplay(int hChannel)

说明：　关闭主机解码视频

输入参数：　hChannel　　通道句柄

6.40 SetDisplayStandard

int SetDisplayStandard (UINT nDisplayChannel,VideoStandard_t VideoStandard)

参数：　UINT　nDisplayChannel　示通道索引

　　VideoStandard_t　VideoStandard　视频制式

说明：　设置视频输出通道的视频制式

6.41 int SetDisplayRegion

int SetDisplayRegion (UINT nDisplayChannel,UINT nRegionCount,REGION_PARAM *pParam,UINT nReserved)

参数：　UINT　nDisplayChannel　显示通道索引
　　UINT　nRegionCount　区域个数
　　REGION_PARAM *pParam　区域参数
　　UINT　nReserved　保留

```
typedef struct
    {
    UINT left;                  区域左边界
    UINT top;                   区域上边界
    UINT width;                 区域宽度
    UINT height;                区域高度
    UINT r;                     区域背景色红色
    UINT g                      区域背景色绿色
    UINT b;                     区域背景色蓝色
    UINT param;                 区域扩展参数
}REGION_PARAM;
#define MAX_DISPLAY_REGION    16   //每个显示窗口最多可划分为 16 个区域。
```

说明：　将显示通道划分为多个区域。Left、width 需要按 16 对齐，top、height 需要按 8 对齐。

返回：

ErrorCodeNotSupport：DSP 资源不足，无法划分窗口，此时需要缩小窗口大小。每块 MD 卡每一路显示通道最大支持 1 个 4CIF＋2 个 QCIF 窗口，如果该卡的某个显示通道的总面积超过该限制，就会返回 ErrorCodeNotSupport。

ErrorCodeInvalidDevice：nDisplayChannel 溢出

ErrorCodeInvalidArgument：nRegionCount 溢出，参数对齐错误，区域超出范围

其它情况下返回 ERR_KERNEL。

6.42 ClearDisplayRegion

int ClearDisplayRegion (UINT nDisplayChannel, UINT nRegionFlag)

参数：　UINT　nDisplayChannel　显示通道索引
　　UINT　RegionFlag　要清除的区域

说明：　把显示区域清空，显示 SetDisplayRegion 中所设置的背景色。Bit0－Bit15：对应区域 1－16，对应位为 1，则将该区域清空。

注：　如果该区域当前有图像正在显示，则该命令无效。

6.43 SetDisplayRegionPosition

int SetDisplayRegionPosition (UINT nDisplayChannel,UINT nRegion,UINT nLeft,UINT nTop)

参数：　UINT　nDisplayChannel　显示通道索引
　　UINT　nRegion　要调整位置的区域
　　UINT　nLeft, UINT nTop　调整后的位置

说明： 改变某个显示区域的位置。

6.44 FillDisplayRegion

int FillDisplayRegion (UINT nDisplayChannel,UINT nRegion,unsigned char *pImage)

参数：			
	UINT	nDisplayChannel	显示通道
	UINT	nRegion	要填充的区域
	unsigned char *	pImage	YUV420 格式的图像指针

说明： 用自定义的图像填充显示区域。pImage 所指向图象的大小必须和 SetDisplayRegion 中设置的图像大小相同，否则图像无法正常显示。

注： 如果该区域当前有图像正在显示，则该命令无效。

6.45 SetDecoderAudioOutput

int SetDecoderAudioOutput(UINT nDecodeChannel,UINT bOpen,UINT nOutputChannel)

参数：			
	UINT	nDecodeChannel	解码通道索引
	UINT	bOpen	开、关
	UINT	nOutputChannel	要输出的显示通道索引

说明： 设置解码通道的音频输出。将第 nDecodeChannel 路解码音频输出到所在 MD 卡上的第 nOutputChannel 个音频输出通道。每块 DS-4002MD 卡包含 2 路音频输出，因此 nOutputChannel 必须为 0 或 1，每块 DS-4004MD 卡包含 4 路音频输出，因此 nOutputChannel 必须为 0，1，2 或者 3。如果已经有其它的解码通道的音频在 nOutputChannel 输出，则系统会自动先将其停止。

6.46 SetDecoderVideoOutput

int SetDecoderVideoOutput(UINT nDecodeChannel,UINT nPort,UINT bOpen,UINT nDisplayChannel,UINT nDisplayRegion,UINT nReserved)

参数：			
	UINT	nDecodeChannel	解码通道索引
	UINT	nPort	解码通道的输出端口
	UINT	bOpen	开、关
	UINT	nDisplayChannel	要输出的显示通道在所在 DSP 中的索引
	UINT	nDisplayRegion	要输出的显示区域
	UINT	nReserved	保留

说明： 设置解码通道的视频输出（MD 卡内部输出）。把视频图像从 nDecodeChannel 个解码通道的第 nPort 路，送到该解码通道所在的 MD 卡内部的第 nDisplayChannel 个显示通道的第 nDisplayRegion 个区域。此函数中 nPort 必须为 0 或 1（每个解码通道支持 2 路视频输出）。nDisplayChannel 必须为 0,1,2 或者 3（一块 MD 卡最多包含 4 个显示通道），如果 bOpen 为 0，则 nDisplayChannel、nDisplayRegion 无意义。

注： 该函数为 SetDecoderVideoExtOutput 函数的特殊情况，SetDecoderVideoExtOutput 即可以将解码图像输出到其它 MD 卡上显示，也可以在当前的 MD 卡内部显示，当只需在 MD 卡内部显示时，使用 SetDecoderVideoOutput 和 SetDecoderVideoExtOutput 都可以。

6.47 SetDecoderVideoExtOutput

int SetDecoderVideoExtOutput(UINT nDecodeChannel, UINT nPort,UINT bOpen, UINT nDisplayChannel,UINT nDisplayRegion, UINT nReserved)

参数：			
	UINT	nDecodeChannel	解码通道索引
	UINT	nPort	解码通道的输出端口

	UINT	bOpen	开、关
	UINT	nDisplayChannel	要输出的显示通道索引
	UINT	nDisplayRegion	要输出的显示区域
	UINT	nReserved	保留

说明： 设置解码通道的视频外部输出(矩阵输出)。把视频图像从第 nDecodeChannel，解码通道的第 nPort 路，送到第 nDisplayChannel(根据 MD 卡的数目，此)个显示通道的第 nDisplayRegion 个区域。此函数中 nPort 必须为 0 或 1（每个解码通道支持 2 路视频输出）。如果 bOpen 为 0，则 nDisplayChannel、nDisplayRegion 无意义。

6.48 SetEncoderVideoExtOutput

int SetEncoderVideoExtOutput(UINT nEncodeChannel, UINT nPort,int bOpen, UINT nDisplayChannel, UINT nDisplayRegion,UINT nFrameRate);

参数：	UINT	nEncodeChannel	编码通道索引
	UINT	nPort	编码通道的输出端口
	UINT	bOpen	开、关
	UINT	nDisplayChannel	要输出的显示通道索引
	UINT	nDisplayRegion	要输出的显示区域
	UINT	nFrameRate	表示帧率

说明： 此函数适用于编码卡和 MD 卡混插的情况或者具备模拟输出功能的板卡上(如 41 系列，42 系列)，可将编码视频数据以矩阵输出的形式直接输出至 MD 卡或者 41、42 卡的模拟输出口。实现本地视频矩阵功能。将视频图像从第 nEncodeChannel 个编码通道的第 nPort 路，输出到系统中第 nDisplayChannel 个显示通道的第 nDisplayRegion 个显示区域上。

注 意： 42 卡目前支持单画面本卡输出

4.1版本新增函数：

6.49 HW_SetFileRef

int HW_SetFileRef(int hChannel, UINT bEnable,FILE_REF_DONE_CALLBACK FileRefDoneCallback)

参数：	int	nChannel	通道句柄
	UINT	bEnable	开、关
	FileRefDoneCallback		索引创建完成后回调函数
	typedef void (*FILE_REF_DONE_CALLBACK)(UINT nChannel,UINT nSize)		
	UINT	nChannel	通道号
	UINT	nSize	索引大小。(暂时无效，以后可以增加索引导出、导入功能)

说 明： 设置文件索引。请在 HW_OpenFile 之前调用。

6.50 HW_GetFileAbsoluteTime

int HW_GetFileAbsoluteTime(int hChannel, SYSTEMTIME *pStartTime, SYSTEMTIME *pEndTime);

参数：	int hChannel	通道句柄
	SYSTEMTIME* pStartTime	文件开始时间
	SYSTEMTIME* pEndTime	文件结束时间

说 明： 得到文件的起止时间（绝对时间）

注意： SYSTEMTIME 中毫秒参数无效，始终为零。

6.51 HW_GetCurrentAbsoluteTime

int HW_GetCurrentAbsoluteTime(int hChannel,SYSTEMTIME *pTime)

参数： int hChannel 通道句柄

SYSTEMTIEM* pTime 文件当前的绝对时间

说 明： 得到文件当前的绝对时间

注意： SYSTEMTIME 中毫秒参数无效，始终为零。

6.52 HW_LocateByAbsoluteTime

int HW_LocateByAbsoluteTime(int hChannel,SYSTEMTIME time)

参数： int hChannel 通道句柄

SYSTEMTIME time 定位的绝对时间

说 明：按照绝对时间定位文件。只在回放文件、且打开索引后有效，

注意： SYSTEMTIME中毫秒参数无效。文件的起止时间由HW_GetFileAbsoluteTime函数获得。

6.53 HW_LocateByFrameNumber

int HW_LocateByFrameNumber(int hChannel,UINT frmNum);

参数： int hChannel 通道句柄

UINT frmNum 帧号

说 明： 按照帧号定位文件。只在回放文件、且打开索引后有效，文件的所有帧个数由HW_GetFileTotalFrames函数获得。

4.2版本新增函数：

6.54 HW_InputDataByFrame

int HW_InputDataByFrame(int hChannel,char* pBuf,int nSize)

说明： 该函数功能也是输入数据，同样必须在打开流之后才能输入数据，而且支持 I 帧数据完整地输入；实时性更好。

参数： 请参考函数 HW_InputData() (见 5.7)的参数说明。

6.55 HW_SetDecoderPostProcess

int HW_SetDecoderPostProcess(int hChannel, UINT param)

说明： 增加解码后防止图像闪烁的功能

参数： int hChannel 通道句柄

UINT param 0: 关闭防闪烁处理。

1: 打开防闪烁处理。

4.3版本新增函数：

6.56 RegisterDecoderVideoCaptureCallback

int RegisterDecoderVideoCaptureCallback(DECODER_VIDEO_CAPTURE_CALLBACK DecoderVideoCaptureCallback, void *context)

说明： 注册解码视频读取回调函数。

参数： DECODER_VIDEO_CAPTURE_CALLBACK

DecoderVideoCaptureCallback	回调函数
void *context	调用回调函数时提供的上下文

输出参数：无

返 回 值：成功为 0，失败则为错误号。

回调函数说明：

typedef void (*DECODER_VIDEO_CAPTURE_CALLBACK)(UINT nChannelNumber,void *DataBuf, UINT width, UINT height, UINT nFrameNum,UINT nFrameTime, SYSTEMTIME *pFrameAbsoluteTime, void*context)

UINT nChannelNumber	通道句柄
void *DataBuf	缓冲区地址
UINT width	图像宽度
UINT height	图像高度
UINT nFrameNum	捕获的当前帧的序号
UINT nFrameTime	捕获的当前帧的时间，单位毫秒（ms）
SYSTEMTIME *pFrameAbsoluteTime	捕获的当前帧的绝对时间
void *context	调用回调函数提供的上下文

6.57 HW_SetDecoderVideoCapture

int HW_SetDecoderVideoCapture(HANDLE hChannel, BOOL bStart, UINT param)

说明： 设置/停止解码视频数据（yuv420）捕获。

输入参数：

HANDLE hChannel		通道句柄
BOOL	bStart	是否启动解码视频捕获
UINT	param	保留，取值 0

输出参数： 无

返 回 值： 成功为 0，失败则为错误号。

5.0版本新增函数：

DS-4108HCV、DS-4116HCV每张板卡支持 1 路模拟视频矩阵输出功能和 1 路模拟音频矩阵输出功能，视频矩阵输出调用原函数 SetEncoderVideoExtOutput实现，音频矩阵输出调用函数SetEncoderAudioOutput或者SetEncoderAudioExtOutput实现，音频矩阵功能对MD卡同样有效。

6.58 SetEncoderAudioOutput

int SetEncoderAudioOutput(UINT nEncodeChannel, UINT bEnable, UINT nOutputChannel)

说明： HCV、HFV 卡音频输出功能。本函数适用于 41、42 板卡的音频输出功能

输入参数：

UINT	nEncodeChannel；	编码通道索引
int	bEnable；	使能
UINT	nOutputChannel；	音频模拟输出通道索引，HCV 卡每张板卡只有一个音频输出口，所以取值只能为 0

返回值：成功返回 0；失败返回-1。

6.59 SetEncoderAudioExtOutput

int SetEncoderAudioExtOutput(UINT nEncodeChannel, UINT nPort,int bOpen,UINT nOutChannel, UINT nReserved)

说明： HCV 卡编码通道音频输出功能，将音频数据从第 nEncodeChannel 个编码通道的第 nPort 路，输出到系统中第 nOutChannel 个音频输出口上。

输入参数：

UINT	nEncodeChannel；	编码通道索引
UINT	nPort；	二次输出，取值为 0 或 1，每个编码通道最多可供模拟输出 2 次
int	bOpen；	使能
UINT	nOutChannel；	音频模拟输出通道索引，外部输出，以系统中所有音频模拟输出通道顺序排列，取值为 0、1、2、3、4……
UINT	nReserved；	保留参数

返回值：成功返回 0；失败返回-1。

6.60 SetDecoderAudioExtOutput

int SetDecoderAudioExtOutput(UINT nDecodeChannel, UINT nPort,int bOpen, UINT nOutChannel, UINT nReserved)

说明： MD 卡音频矩阵输出功能，将音频数据从第 nDecodeChannel 个解码通道的第 nPort 路，输出到系统中第 nOutChannel 个音频输出口上。

输入参数：

UINT	nDecodeChannel；	解码通道索引
UINT	nPort；	二次输出，取值为 0 或 1，每个解码通道最多可供模拟输出 2 次
int	bOpen；	使能
UINT	nOutChannel；	音频模拟输出通道索引，外部输出，以系统中所有音频模拟输出通道顺序排列，取值为 0、1、2、3、4……
UINT	nReserved；	保留参数

返回值：成功返回 0；失败返回-1。

6.61 SetAudioPCIPreview

int SetAudioPCIPreview(int hChannelHandle, AUDPRE_CONFIG *config, int bEnable)

说明： 在 4100 系列板卡上，开启/关闭通道的音频预览功能。调用此函数，只能获取原始音频数据，用户需要在应用程序中自己播放。

注意： 此函数接口只对 4100 系列板卡有效，4000 系列板卡音频预览请使用 SetAudioPreview 函数。当 4100 系列板卡与 4000 系列板卡混插时，可通过 GetBoardInfo 函数来判断音频通道所属板卡的类型。然后各自调用之前提到的两个函数接口。（具体使用方法可参考 audio.h 头文件中的内容）。

输入参数：

int	hChannelHandle	编码通道句柄
AUDPRE_CONFIG *	config	音频 PCI 预览相关结构指针，输入信号量，关闭时可为 NULL
int	bEnable	当取值为 1 时，开启音频 PCI 预览功能，当取值为 0 时，关闭音频 PCI 预览功能

输出参数：

AUDPRE_CONFIG * config 音频 PCI 预览相关结构指针，返回音频数据的存放首地址

返回值：成功返回 0，失败返回-1。

结构体说明：

```
typedef struct _audio_preview_config
{
    char        *dataAddr; 音频预览数据的存放地址。应用程序在此地址中取得音频数据后，需要自行播放。音频数据由 DSP 每隔 40 毫秒上传一次，长度由下面的 audioSize 字段标识
    sem_t       *syncSem; 音频 PCI 预览同步信号量，该信号量由应用程序分配
    int         audioSize;      由 SDK 返回给用户的信息，表示音频数据长度
    int         freq;           由 SDK 返回给用户的信息，表示音频数据采样频率
    int         samples;        由 SDK 返回给用户的信息，表示音频数据样本长度（字节单位）
    int         channels;       由 SDK 返回给用户的信息，表示音频数据声道数
} AUDPRE_CONFIG, *PAUDPRE_CONFIG;
```

科技呵护未来

First Choice for Security Professionals