# AVRecord主要流程

## 文件格式模块(统一 格式转换 接口)

typedef int (\*FOnWriteOneFrame)(HANDLE hWnd,void \* pBuf, DWORD nSize);

typedef struct{

HANDLE hWnd;//文件处理句柄

int nType;//0:未知, 1:asf, 2:avi, 3:mp4 ...

int nStat;//0:空闲, 1:正在录制, 2:正在停止录像

int (\*OnSendOneDataPack)(HANDLE hWnd,void \* pBuf);//FileFormatSetCallBack时设置

}TRecord;

//文件格式转换回调

int FileFormatOnReord(void \* context, DWORD nIP, WORD nChannel, void \* pBuffer,

DWORD nSize, BYTE Flag)

{

//TRecord \*pRecd=(TRecord \*)context;

//write file

}

void FileFormatInit(TRecord \*pRecd)

{

memset(pRecd, 0, sizeof(TRecord));

}

BOOL FileFormatNew(TRecord \*pRecd, int nType)

{

if(nType==1){

pRecd->nType=nType;

pRecd->hWnd=InitAsfRecord();

return pRecd->hWnd?TRUE:FALSE;

}

return FALSE;

}

void FileFormatSetAudioInfo(TRecord \*pRecd, DWORD wFormatTag, DWORD nChannels,

DWORD SampleRate, DWORD BitRate)

{

asfSetAudioFormat....

}

void FileFormatSetVideoInfo(TRecord \*pRecd, DWORD nWidth, DWORD nHeight, DWORD

dwCompressID)

{

asfSetVideoSize....

}

/\* FileFormatSetCallBack

NetStreamCallBackFun:转码回调函数, 转成文件流, 如: FileFormatOnReord

FOnWriteOneFrame:写数据帧回调函数, 如asf SendOneFrame

\*/

void FileFormatSetCallBack(TRecord \*pRecd, NetStreamCallBackFun pFun,

FOnWriteOneFrame pFunWrite, void \* context ,DWORD nIP, WORD nchannel)

{

pRecd-> OnWriteOneFrame =pFunWrite;

//check nType

asfSetCallBackFun(pRecd->hWnd, pFun, context, nIP, nChannel);

}

BOOL FileFormatStartRecord(TRecord \*pRecd, BOOL Seekable, DWORD RecordFlag)

{

//asfStartRecord....

recd.nStat=1;

}

void FileFormatStopRecord(TRecord \*pRecd)

{

recd.nStat=0;

//StopRecord....

//检测回写文件

}

int FileFormatWritePack(TRecord \*pRecd, void \*buf,)

{

return pRecd->OnWriteOnePack(pRecd->hWnd , buf);

}

//--------------------------------------------------

## 文件存储模块(统一 写文件 接口)

typedef void (\*RecordMsgCallBack)(int nMsg); //nMsg:0:未使用, 1,启动录像失败(AVRECORD\_xxxx)

typedef int (\*FileOnWrite)(const void \*ptr, DWORD nLen, struct FileStorage \*pFile, int nFlag);

typedef struct FileStorage{

int nType;//0:未知, 1:SD卡, 2:U盘, 3:硬盘 4:ftp ...

FILE \*fp;

FtpClient ftp;

FileOnWrite OnWrite;

char \*sFileName;//保存fileOpen时的文件名(ftpSeek时要使用)

int nFileStat;//0:已经关闭, 1:已经打开/可写, 2:写出错(要检测u盘/sd卡/ftp状态)

DWORD nFileSize;//文件大小

}TFile;

void fileInit(TFile \*pFile)

{

pFile->nType=0;

pFile->fp=NULL;

ftpInit(&ftp);

}

//检测是否可写

BOOL fileCheck(TFile \*pFile)

{

//U盘

//SD卡

//ftp

...

}

//打开文件

int fileOpen(TFile \*pFile, char \*sFileName, int nType)

{

//本地，SD卡，U盘，网络存储

}

//文件定位

int fileSeek(TFile \*pFile, DWORD offset, int nFlag)

{

//本地，SD卡，U盘，网络存储

}

//写文件

int fileWrite(TFile \*pFile, void \*buf, DWORD nLen, int nFlag)

{

//本地，SD卡，U盘，网络存储

}

//关闭文件

void fileClose(TFile \*pFile)

{

//本地，SD卡，U盘，网络存储

}

//--------------------------------------------------

## 调度模块(统一 启动/关闭 录像接口)

typedef struct{

TFile hFile;//保存文件

TRecord hRecord;//文件格式转换

char szTime[64];

DWORD starttime;//开始录像时间(录像中重新录像时,会重装计算)

DWORD nRecordedTime;//已录制时间(毫秒)

DWORD nDuration;//记录时间(1/1000秒)

int nRecordFlag;//1:声音, 2: 视频, 3:声音+视频

int nFileFormat;

int nStorageType;

int nFileStat;//(同 nFileStat)0:已关闭, 1:正在写文件, 2:写文件出错,(要检测u盘/sd卡/ftp状态)

int nReaderId;//

DWORD nDiskNo;//DMS\_NET\_SINGLE\_HD.dwHDNo

int nStreamType;

int nTrigerType; //TRIGER\_XXX

char sFileName[1024];

RecordMsgCallBack OnCallBack;

}TAVRecord;

//模块入口

int FtpModeInit()

{

gs\_hWndRecord = dms\_sysnetapi\_open(DMS\_NETPT\_DMS\_CMS);//10:CGI

if(gs\_hWndRecord<0){

printf("Open platform error!\n");

return 0;

}

if(AVRecordInit(&gs\_avrecord)){

gs\_nRecordRun=1;

pthread\_create(&gs\_tidMainRecord, NULL, OnMainRecord, NULL);

pthread\_create(&gs\_tidDelOldFile, NULL, OnCheckAndDelOldFile, NULL);

}else{

printf("AVRecord Init error!\n");

}

return 0;

}

void \* **OnMainRecord** (void \*param)

{

while(1)

{

//检测计划是否录像

StartRecord(TRIGER\_SCHED, gs\_nChannel, 0);

}

}

//注册回调函数

RecordMsgCallBack()

{

}

GetRecordAttr(nCmd, void \*lpParam, nSize)

{

}

SetRecordAttr(nCmd, void \*lpParam, nSize)

{

nCmd:

ftp属性

NAS属性

录像计划属性

}

//检测是否正在录像，如果正在录像（是指storagetype也相关，不同的storagetype独立判断），只是把duration取最大值(毫秒)，并且异或filetype。

//返回0为成功，其他值为错误码

int StartRecord(int nTrigerType, int nChannel, DWORD nDuration)

{

GetRecordAttr();//获取录像配置信息

//获取 文件类型

switch(nFileType){

asf:

avi:

...

}

//获取 存储类型

switch(nMedium){

sd:

usb:

ftp:

....

}

//确定保存文件名及路径

fileCheck()

//收集所有必须的录像信息(如:时间长度，存储试等

if(AVRecordStart(&gs\_avrecord)==0){

pthread\_t tid;

pthread\_create(&tid, NULL, OnRecordThread, NULL);

return 0;//成功

}

//打开失败

AVRecordFree(&gs\_avrecord);

return -1;

}

//所有信息都收集好了才调用 AVRecordStart

int AVRecordStart(TAVRecord \*pRecd)

{

读取码流信息，通道等

fileOpen(&pRecd->hFile, pRecd->sFileName, "w+", pRecd->nStorageType);

FileFormatNew()

dms\_sysnetapi\_MBUF\_AddReader()

FileFormatSetAudioInfo(&pRecd->hRecord, DMS\_WAVE\_FORMAT\_G711\_ADPCM, 1, wBitsPerSample, 16);

FileFormatSetVideoInfo(&pRecd->hRecord, nWidth, nHeight, dwCompressID);

FileFormatSetCallBack(&pRecd->hRecord, FileFormatOnReord, pRecd, 0, gs\_nChannel);

AVRecordSetMsgCallBack(pRecd, RecordMsgCallBackFun);//设置回调函数

FileFormatStartRecord(&pRecd->hRecord, 1, pRecd->nRecordFlag, pRecd->nDuration);//通知OnRecordThread,开始录像

return blRet?0:-1;

}

/\* OnRecordThread

主录像线程

\*/

static void \*OnRecordThread(void \*param)

{

while(gs\_avrecord.hRecord.nStat)

{

s32Ret=dms\_sysnetapi\_MBUF\_GetReadPtrPos(gs\_hWndRecord,gs\_avrecord.nRea.....);

//检测是否关闭录像

//检测是否已达到录像时间长度

//写数据

s32Ret=FileFormatWritePack(&gs\_avrecord.hRecord,&readPtr);

//重连检测

if(gs\_avrecord.hFile.nFileStat==2){//asfSendOneFrame失败/ftp断开连接

pthread\_mutex\_lock(&mutexTask);

fileClose()

while(gs\_avrecord.hRecord.nStat){//正在录像

if(!fileCheck(nFileType, gs\_avrecord.sFileName)){

sleep(1);

continue;

}

fileOpen(&gs\_avrecord.hFile, gs\_avrecord.sFileName, ...);//重新打开

break;

}

pthread\_mutex\_unlock(&mutexTask);

break;//正常退出录像状态

}

}

lock()

AVRecordFree(&gs\_avrecord);

unlock()

}

StopRecord(nChannel, trigertype, storagetype,)

{

//关闭所有正在录制的文件

close socket

stop record

fflush();

close file

}

StopAllRecord()

{

//关闭所有正在录制的文件

close socket

stop record

fflush();

close file

}

# 相关结构体

录像信息：

typedef struct

{

DWORD dwSize; /\* 此结构的大小 \*/

DWORD dwChannel;

DWORD dwRecord; /\*是否录像 0-否 1-是\*/

DMS\_RECORDSCHED stRecordSched; //录像时间段，星期一到星期天

DWORD dwPreRecordTime; /\* 预录时间，单位是s，0表示不预录。 \*/

DWORD dwRecorderDuration; /\*录像保存的最长时间\*/

BYTE byRedundancyRec; /\*是否冗余录像,重要数据双备份：0/1, 默认为不启用\*/

BYTE byAudioRec; /\*录像时复合流编码时是否记录音频数据：国外有此法规,目前版本不支持\*/

BYTE byRecordMode; //0:自动模式（按定时录像配置） 1:手动录像模式（全天侯录像），2:禁止所有方式触发录像

BYTE byStreamType; //录像码流类型 0：主码流 1：子码流

BYTE byReserve[8];

}DMS\_NET\_CHANNEL\_RECORD, \*LPDMS\_NET\_CHANNEL\_RECORD;

设备信息:

typedef struct tagDMS\_NET\_DEVICE\_INFO {

DWORD dwSize;

char csDeviceName[DMS\_NAME\_LEN];//DVR名称

DWORD dwDeviceID; //DVR ID,用于遥控器

.....

}

tp上传参数

typedef struct tagDMS\_NET\_FTP\_PARAM

{

DWORD dwSize;

BOOL bEnableFTP; /\*是否启动ftp上传功能\*/

char csFTPIpAddress[DMS\_MAX\_DOMAIN\_NAME]; /\*ftp 服务器，可以是IP地址或域名\*/

DWORD dwFTPPort; /\*ftp端口\*/

char csUserName[DMS\_NAME\_LEN]; /\*用户名\*/

char csPassword[DMS\_PASSWD\_LEN]; /\*密码\*/

WORD wTopDirMode; /\*0x0 = 使用设备ip地址,0x1 = 使用设备名,0x2 = OFF\*/

WORD wSubDirMode; /\*0x0 = 使用通道号 ,0x1 = 使用通道名,0x2 = OFF\*/

BYTE reservedData[28]; //保留

}DMS\_NET\_FTP\_PARAM, \*LPDMS\_NET\_FTP\_PARAM;

本地硬盘信息配置

typedef struct

{

DWORD dwSize;

DWORD dwHDCount; //硬盘数

DMS\_NET\_SINGLE\_HD stHDInfo[DMS\_MAX\_DISKNUM];

}DMS\_NET\_HDCFG, \*LPDMS\_NET\_HDCFG;

# 主框架图

AVRecord

FileStorage

FileFormat

AVI或其它格式

ASF

Other Net FS

FTP

USB/SD/ LocalFile

FtpModeInit()中启动 OnMainRecord()，检测当前录像模式状态，确定当前是否录像。