视频模块

类

```
class Capture;
```
摄像头类

```
class VideoWriter;
```
录像类

```
struct Size {
    int width;
    int height;
};
```
分辨率类

函数

摄像头类：

```
ErrorCode Capture::getError();
```
功能：获取上次函数运行的错误码
返回：错误码

```
bool Capture::isConnectCapture();
```
功能：检查设备是否已经连接
返回：已连接返回 true，未连接返回 false

```
vector<Size> Capture::getCaptureResolution(int captureType);
```
功能：返回对应摄像头的分辨率列表
返回：分辨率列表
参数：captureType 为摄像头类型（0 卡证 1 人脸）

```
bool Capture::openCapture(int captureType, Size resolution);
```
功能：打开摄像头
返回：打开成功返回 true，打开失败返回 false
参数：captureType 为摄像头类型（0 卡证 1 人脸）
resolution 为分辨率

double Capture::getFPS();
功能：获取摄像头当前帧率
返回：当前打开的摄像头的 FPS，未打开时返回-1

const unsigned char* Capture::getFrame();
功能：获取当前帧图像
返回：当前帧的图像，使用 24 位 RGB 格式（8-8-8）存储图像

录像类：
ErrorCode VideoWriter::getError();
功能：获取上次函数运行的错误码
返回：错误码

bool VideoWriter::startWrite(const char* filename, double fps);
功能：开始录像（保存为 MP4 格式）
返回：录像成功返回 true，录像失败返回 false
参数：filename 为文件名，fps 为帧率

bool VideoWriter::writeFrame(const unsigned char* frame);
功能：为录像传入一帧数据
返回：传入成功返回 true，传入失败返回 false
参数：frame 为图像数据，使用 24 位 RGB 格式（8-8-8）存储

void VideoWriter::endWrite();
功能：结束录像

卡证读取

类

class Inductor;
读卡器类

```
enum class InductorType {
    IDCard,         //身份证
    ICCard,         //IC 卡
    MagneticCard,       //磁条卡
    BusCard,        //深圳通
```

```
    All         //所有
};
```
读卡器种类

```
enum IDCardInfo {
    ChipID,         //芯片 ID
  Name,         //姓名
  Sex,          //性别
  Ethnic,           //民族
  Birthday,         //生日
  Address,          //地址
  CardNumber,           //卡号
  IssuingAgency,        //签发机关
  EffectiveDate,        //有效期始
  ExpirationDate,       //有效期止
    HeadPortrait        //头像（返回 bmp 文件路径）
};
```
身份证信息

```
enum ICCardInfo {
    ChipID,         //芯片 ID
    CardNumber,         //卡号
    …           //其他能读到的信息
    RecordTotal         //记录总数
};
```
IC 卡信息

```
enum BankCardRecordInfo {
    DealTime,           //交易时间
    DealMoney,          //交易金额
    TerminalCode,           //终端代码
    CountryCode,        //国家代码
    DealType            //交易类型
};
```
银行卡记录信息

```
enum MagneticCardInfo {
    ChipID,         //芯片 ID
    …           //其他能读到的信息
};
```
磁条卡信息

```
enum BusCardInfo {
    ChipID,         //芯片 ID
```

```
    CardNumber,        //卡号
    Balance,        //余额
    …          //其他能读到的信息
    RecordTotal        //记录总数
};
```
深圳通信息

```
enum BusCardRecordInfo {
    DealTime,          //交易时间
    DealMoney,         //交易金额
    TerminalCode,            //终端代码
    DealType           //交易类型
};
```
深圳通记录信息

函数

读卡器类

```
ErrorCode Inductor::getError();
```
功能：获取上次函数运行的错误码

返回：错误码

```
bool Inductor::init(CardKind cardKind = InductorType::All);
```
功能：初始化读卡器模块

返回：初始化成功返回 true，初始化失败返回 false

参数：cardKind 为读卡器的种类，默认为 All，即全部初始化

```
bool Inductor::setInductor(bool isOpen, InductorType cardKind);
```
功能：打开/关闭对应读卡器设备

返回：打开成功返回 true，打开失败返回 false

参数：isOpen 为输入指令，为 true 时打开读卡器，为 false 时关闭读卡器

cardKind 为读卡器的种类

```
int Inductor::getInductor(InductorType cardKind);
```
功能：获取对应读卡器设备状态

返回：对应读卡器模块未初始化返回-1，已经初始化未打开返回 0，已经打开返回 1

参数：cardKind 为读卡器的种类

```
bool Inductor::readCard (InductorType cardKind);
```
功能：命令对应读卡器设备读取卡证信息

返回：读卡成功返回 true，读卡失败返回 false

参数：cardKind 为读卡器的种类

std::string Inductor::getCardInfo (InductorType cardKind, int index);

功能：获取对应读卡器设备已经读取的卡证信息

返回：卡证中对应的字段信息，若没有此字段信息返回空值

参数：cardKind 为读卡器的种类

index 为字段索引，依照读卡器设备类型，输入枚举 IDCardInfo、ICCardInfo、MagneticCardInfo、BusCardInfo 中的数值

std::string Inductor::getCardRecordInfo (InductorType cardKind, int sequence, int index);

功能：获取对应读卡器设备已经读取的卡证记录信息

返回：卡证中对应的字段信息，若没有此字段信息返回空值

参数：cardKind 为读卡器的种类

sequence 为信息的序列号（第几条记录）

index 为字段索引，依照读卡器设备类型，输入枚举 BankCardRecordInfo、BusCardRecordInfo 中的数值

其他

类

```
class enum ErrorCode {
    UnknownError = -1,          //未知错误
    OK = 0,         //无错误
    Error_NotOpen_Uninit = …,       //错误：无法打开设备，对应模块未初始化
    Warning_NotOpen_AlreadyOpen = …,            //警告：无法打开设备，该设备已经打开
    …           //其他可能的错误
};
```

执行结果类