KEYENCE 基恩士

LJ-X8000A

简易图像数据获取DLL

使用说明书

1、应用本DLL的使用场景及功能

2、运行环境

3、文件构成

4、引用方法

5、数据结构与类型、CallBack函数

6、库及函数一览

7、建议使用流程

1、应用此DLL的使用场景

1-1 使用场景

此DLL针对LJ-X8000A的标准通讯库（LJX8IF.DLL)的二次封装

针对LJ-X8000A经常使用的功能进行了简化，可以简单快速得从LJ-X8000A控制器取得高度/图像数据及轮廓信息，以期大幅缩短客户初次使用时导入LJ-X8000A的时间

由于仅针对通常使用场景，对于特殊场合（如超大图像、超低延迟时间、实时获取当前轮廓数据、第三方软件特殊数据格式要求等），建议客户使用标准LJ-X8000通讯库进行修改实现。

1-2 DLL内封装的功能

①从控制器获取批处理轮廓数据、浓淡数据，返回数据类型为Byte数组或UShort数组

②同时与最多六台控制器进行高速轮廓传输

③将获取到的数据保存到PC硬盘上的指定位置，保存格式为Tiff、Bmp或Csv。

2、运行环境

操作系统	Windows 10（Home/Pro/Enterprise） Windows 7（SP1 或更高版本）（Home Premium/Professional/Ultimate）
CPU	intel®Core™ i3 处理器同级规格
内存容量	8 GB 以上
硬盘可用空间	10 GB 以上
接口	配备下列任意一项。 Ethernet 1000BASE-T/100BASE-TX *1

*1 不保证连接到 LAN 和使用路由器连接时的操作状况。

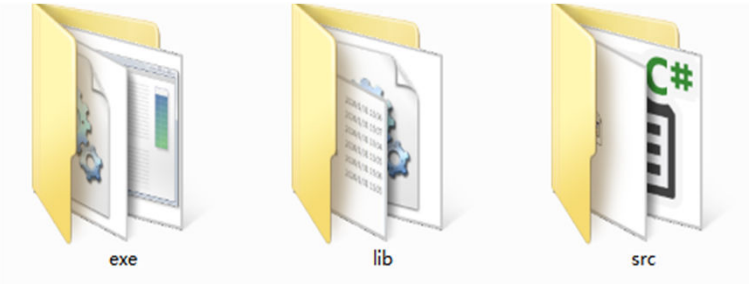
2-1 运行环境

①Microsoft C Runtime Library（通常系统已经预装，如果运行报错，请至微软官网下载）

②Microsoft .NET Framwork 4.5或以上（通常系统已经预装，如果运行报错，请至微软官网下载）

注：仅支持64bit Windows操作系统

3、文件构成



1、exe

包含了编译后的DLL使用样例程序可执行exe文件及对应的DLL文件，打开后可直接运行

如果提示错误，请参照使用环境要求进行使用环境确认

2、lib

包含了LJXADriverFor.Net.dll及LJX8IF.dll,仅支持64bit系统

3、src

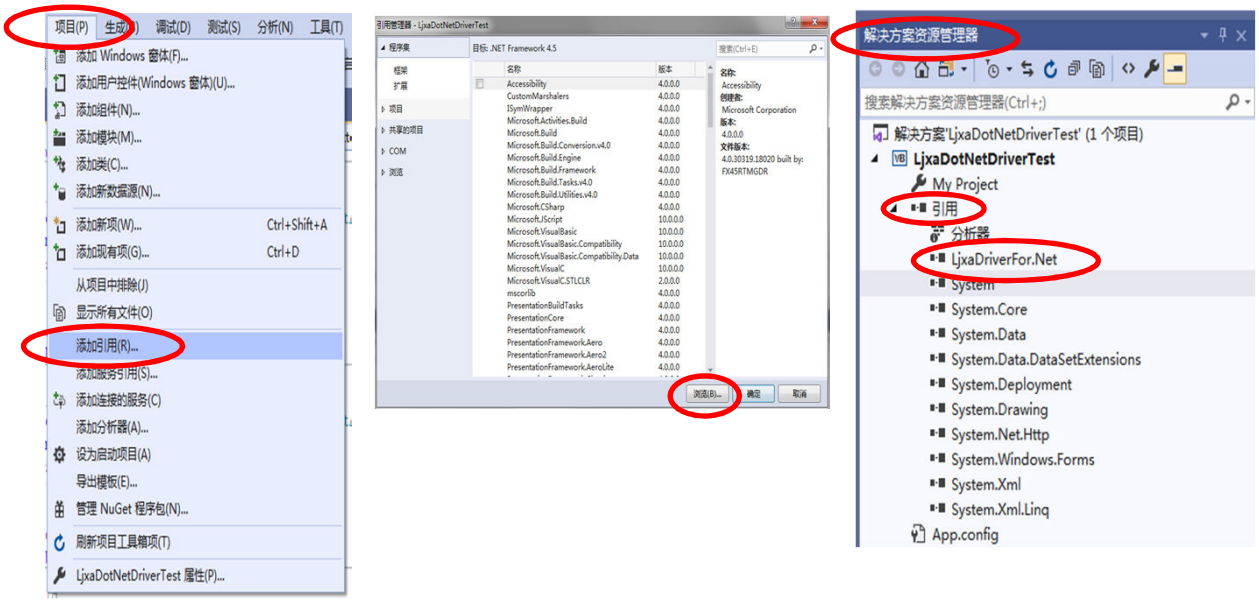
LJXADriverFor.Net的使用样例程序源代码

3、VisualStudio中引用此DLL的方法

如下图所示，在**项目**选项下单击**添加引用**，在弹出的引用管理器窗口中，单击**浏览**

在弹出的文件管理器窗口中，找到LJXADriverFor.Net.DLL文件，单击确定添加引用

查看程序的引用列表，确保LjxaDriverFor.Net已添加引用，如下最右图



5、数据结构、CallBack函数

5-1 结构体

CommunicationSetting　　连接控制器时使用的结构体

所在库：LjxaSample.DriverDotNet.CommunicationSetting

属性名称：	类型	只读/读写	默认值	描述
CallbackFrequency	UInt16	R/W	200	内部处理轮廓数据的频率，不建议修改
StartPosition	UInt16	R/W	2	开始发送轮廓的位置，0→从最早轮廓开始，1→从上次发送停止的位置，2→从下一条轮廓开始，建议使用2
YImageSize	Int	R/W	0	需要的扫描行数，下限为50，上限与设定内容有关（42000~60000），连接控制器时会自动修改控制器内批处理设定
UseLuminanceData	Bool	R/W	false	是否需要浓淡数据（此设定不会修改控制器内的浓淡输出设定）
UseExternalBatchStart	Bool	R/W	false	是否使用外部批处理开始信号（如外部端子），不使用时，在获取开始时会自动发送批处理开始指令
EthernetSetting	LJXA_EthernetSet	R/W	false	包括IP地址、端口号等通信信息，通常不用手动初始化，在对此结构进行初始化时会内

结构体初始化：

VB.Net

```
Sub CommunicationSetting.New(IPAddress As Byte(), CommandPortNo As UShort,
HighSpeedPortNo As UShort, YImageSize As Integer, UseExternalBatchStart As Boolean,
UseLuminanceData As Boolean) (+ 1 重载)
```

C#

```
CommunicationSetting.CommunicationSetting(byte[] IPAddress, ushort CommandPortNo,
ushort HighSpeedPortNo, int YImageSize, bool UseExternalBatchStart, bool
UseLuminanceData) (+ 1 重载)
```

控制器IP地址为192.168.0.1，指令端口24691，高速通信端口24692，目标图像行数3000行

需要使用浓淡图像，不使用外部批处理开始信号时，初始化函数如下：

VB.NET

```
Dim _ComSet As CommunicationSetting = New CommunicationSetting
(New Byte() {192, 168, 0, 1}, 24691, 24692, 3000, False, True)
```

C#

```
CommunicationSetting _Comset = new CommunicationSetting
(new byte[] { 192, 168, 0, 1 },24691, 24692, 3000, false, true);
```

5-2 Callback函数

此DLL使用回调函数来通知主程序当前DLL高速通信状态，可以主程序处理图像时异步获取数据

AcquireFinishCallback　DLL高速通信状态变化时，调用此函数通知主程序进行处理

所在库：LjxaSample.DriverDotNet.LJX8000A.AcquireFinishCallback

函数定义：

VB.NET
```
Delegate Sub LjxaSample.DriverDotNet.LJX8000A.AcquireFinishCallback(DeviceID As
Integer, Notify As Integer)
```

C#
```
delegate void LjxaSample.DriverDotNet.LJX8000A.AcquireFinishCallback
(int DeviceID, int Notify)
```

使用方法：

需要在主程序内定义域上述输出输入参数一致的函数，并在连接设备时将函数地址作为参数导入

Callback函数定义示例：

VB.NET
```
Private Sub AcquiredFinish(DeviceID As Integer, Notify As Integer)
  '需要实现的函数内容，输入参数DeviceID为控制器的ID编号
  'Notify为高速通信状态变化的指示值
End Sub
```

C#
```
 private void AcquiredFinish(int DeviceID,int Notify)
{
  //需要实现的函数内容，输入参数DeviceID为控制器的ID编号
  //Notify为高速通信状态变化的指示值
}
```

6、库及函数一览

6-1 命名空间

根命名空间：LjxaSample

包含两个子命名空间

①DriverDotNet　包含二次封装后的图像数据采集函数、图像保存函数及对应的数据结构

②LJX8IF　　LJ-X8000A标准DLL封装的函数引出及对应数据结构，备用库

此处仅针对DriverDotNet命名空间内的函数进行说明

6-2 DriverDotNet

此DLL中DriverDotNet封装了LJX8000A库，所有函数为非共享函数，实现前需要创建实例之后使用，如下是VB.NET和C#中的实例化示例：

以下示例中创建了名称为**LJXA**的LjxaSample.DriverDotet.LJX8000A的实例

VB.NET
```
Dim LJXA As LjxaSample.DriverDotNet.LJX8000A = New LjxaSample.DriverDotNet.LJX8000A
```

C#
```
LjxaSample.DriverDotNet.LJX8000A LJXA = new LjxaSample.DriverDotNet.LJX8000A();
```

6-2-1 LjxaSample.DriverDotNet.LJX8000A

LJX8000A内包含的库和函数列表如下

库：SaveReceiveDataAsFile，用于将接收到的数据保存到PC上的库，详情请参照6-2-2

LjxaSample.DriverDotNet.LJX8000A的函数列表：

（以下所有函数返回类型都是布尔型，返回True时执行成功，返回False执行失败

①OpenDevice　　与控制器建立通信，并为控制器定义ID

②CloseDevice　　断开与控制器的通信，并释放对应ID

③AcquireStart　打开控制器的高速通信，DLL进入等待数据状态

④AcquireStop　　发送批处理结束信号，并暂停高速通信

⑤GetHeightData　读取获取到的高度数据

⑥GetLuminanceDa读取获取到的浓淡数据

⑦GetProfileInfc读取轮廓信息

OpenDevice的函数定义　　与控制器建立通信，并为控制器定义ID

VB.NET

```
Function LJX8000A.OpenDevice(CurrentDeviceID As Byte, _ethernetSetting As CommunicationSetting,
_acquiredFinishCallback As LJX8000A.AcquireFinishCallback) As Boolean
```

C#

```
bool LJX8000A.OpenDevice(byte CurrentDeviceID, CommunicationSetting _ethernetSetting,
LJX8000A.AcquireFinishCallback _acquiredFinishCallback)
```

输入参数	描述
CurrentDeviceID	分配控制器的编号，范围为0~5
_ethernetSetting	通讯相关的设定，参照5-1 CommunicationSetting
_acquiredFinishCallback	Callback函数的地址

CloseDevice的函数定义　　断开与控制器的通信，并释放对应ID

VB.Net

```
bool LJX8000A.CloseDevice(byte CurrentDeviceID)
```

C#

```
Function LJX8000A.CloseDevice(CurrentDeviceID As Byte) As Boolean
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5

AcquireStart的函数定义　　打开控制器的高速通信，DLL进入等待数据状态

VB.Net
```
Function LJX8000A.AcquireStart(CurrentDeviceID As Byte) As Boolean
```

C#
```
bool LJX8000A.AcquireStart(byte CurrentDeviceID)
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5

AcquireStop的函数定义　　发送批处理结束信号，并暂停高速通信

VB.Net
```
Function LJX8000A.AcquireStop(CurrentDeviceID As Byte) As Boolean
```

C#
```
bool LJX8000A.AcquireStop(byte CurrentDeviceID)
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5

GetProfileInfo的函数定义　　读取轮廓信息

VB.Net
```
Function LJX8000A.GetProfileInfo(CurrentDeviceID As Byte, ByRef ProfInfo As
LjxaSample.LJX8IF.LJX8IF_PROFILE_INFO) As Boolean
```

C#
```
bool LJX8000A.GetProfileInfo(byte CurrentDeviceID, ref LjxaSample.LJX8IF.LJX8IF_PROFILE_INFO
ProfInfo)
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5
ProfInfo	输出的轮廓信息，ProfInfo的内容如下（结果输出）

已获取轮廓的轮廓信息。

```
Typedef  struct  {
    BYTE     byProfileCount;          // 固定为 1
    BYTE     reserve1;
    BYTE     byLuminanceOutput;       // 有、无亮度 （1：有亮度、 0：无亮度）
    BYTE     reserve2;
    WORD     wProfileDataCount;       // 轮廓的数据数
                                        （初始设定分别为 LJ-X：3200、
                                        LJ-V：800）
    BYTE     reserve3[2];
    LONG     lXStart;                 // 第 1 点的 X 坐标。
    LONG     lXPitch;                 // 轮廓数据的 X 方向间隔。
}       LJX8IF_PROFILE_INFO;
```

以 0.01 μm 为单位存储 lXStart、lXPitch。

GetHeightData的函数定义　　读取获取到的高度数据

此函数与GetLuminanceData均有4个重载函数，以下是各个重载的定义

VB.Net

```
Function LJX8000A.GetHeightData(CurrentDeviceID As Byte, ByRef HeightDatas As Byte()) As Boolean (+ 3 多个重载)
Function LJX8000A.GetHeightData(CurrentDeviceID As Byte, ByRef HeightDatas As UShort()) As Boolean (+ 3 多个重载)
Function LJX8000A.GetHeightData(CurrentDeviceID As Byte, ByRef HeightDatas As Byte(), ByRef Width As Integer, ByRef XPit
As Integer) As Boolean (+ 3 多个重载)
Function LJX8000A.GetHeightData(CurrentDeviceID As Byte, ByRef HeightDatas As UShort(), ByRef
Width As Integer, ByRef XPitch As Integer) As Boolean (+ 3 多个重载)
```

C#

```
bool LJX8000A.GetHeightData(byte CurrentDeviceID, ref byte[] HeightDatas) (+ 3 多个重载)
bool LJX8000A.GetHeightData(byte CurrentDeviceID, ref ushort[] HeightDatas) (+ 3 多个重载)
bool LJX8000A.GetHeightData(byte CurrentDeviceID, ref byte[] HeightDatas, ref int Width, ref int XPitch)
bool LJX8000A.GetHeightData(byte CurrentDeviceID, ref ushort[] HeightDatas, ref int Width, ref int
XPitch) (+ 3 多个重载)
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5
HeightDatas	输出的轮廓数据，byte数组或ushort数组（结果输出）
Width（可选）	单条轮廓的数据点数（即图像的宽度）（结果输出）
Xpitch（可选）	X方向的点间距（单位为0.01um）（结果输出）

注意点：

①DLL内部缓存的数据为byte类型，读取为ushort型时，需要额外时间进行处理

转换参考时间：3,200,000 数据点/10ms，具体时间与PC的性能相关

②未收到回调函数通知已获取完成的情况下，不能读取缓存数据

未通知情况下访问缓存可能会出现线程间冲突导致异常发生

③Byte数组中实际为16bit数据，按照little_endian方式排列的byte数据

GetLuminanceData的函数定义 读取获取到的浓淡数据

此函数与GetHeightData均有4个重载函数，以下是各个重载的定义

VB.Net

```
Function LJX8000A.GetLuminanceData(CurrentDeviceID As Byte, ByRef LuminanceDatas As Byte()) As Boolean (+ 3 多个重载)
Function LJX8000A.GetLuminanceData(CurrentDeviceID As Byte, ByRef LuminanceDatas As UShort()) As Boolean (+ 3 多个重载)
Function LJX8000A.GetLuminanceData(CurrentDeviceID As Byte, ByRef LuminanceDatas As Byte(), ByRef Width As Integer, ByRef XPitch As Integer) As Boolean (+ 3 多个重载)
Function LJX8000A.GetLuminanceData(CurrentDeviceID As Byte, ByRef LuminanceDatas As UShort(), ByRef Width As Integer, ByRef XPitch As Integer) As Boolean (+ 3 多个重载)
```

C#

```
bool LJX8000A.GetLuminanceData(byte CurrentDeviceID, ref byte[] LuminanceDatas) (+ 3 多个重载)
bool LJX8000A.GetLuminanceData(byte CurrentDeviceID, ref ushort[] LuminanceDatas) (+ 3 多个重载)
bool LJX8000A.GetLuminanceData(byte CurrentDeviceID, ref byte[] LuminanceDatas, ref int Width, ref int XPitch)
bool LJX8000A.GetLuminanceData(byte CurrentDeviceID, ref ushort[] LuminanceDatas, ref int Width, ref int XPitch) (+ 3 多个重载)
```

输入参数	描述
CurrentDeviceID	目标控制器编号，范围为0~5
LuminanceDatas	输出的轮廓数据，byte数组或ushort数组（结果输出）
Width（可选）	单条轮廓的数据点数（即图像的宽度）（结果输出）
Xpitch（可选）	X方向的点间距（单位为0.01um）（结果输出）

注意点：

①DLL内部缓存的数据为byte类型，读取为ushort型时，需要额外时间进行处理

转换参考时间：3,200,000 数据点/10ms，具体时间与PC的性能相关

②未收到回调函数通知已获取完成的情况下，不能读取缓存数据

未通知情况下访问缓存可能会出现线程间冲突导致读取异常发生

③Byte数组中实际为16bit数据，按照little_endian方式排列的byte数据

④GetLuminanceData的使用方法与GetHeightData一致

⑤控制器的浓淡输出为false或连接控制器时指定不使用浓淡数据时，返回数据为空数组

6-2-2 LjxaSample.DriverDotNet.LJX8000A.SaveReceiveDataAsFile

①枚举类型　　ImageSaveType　　用于保存图像时的格式选择枚举项，包含内容如下

1） Tiff　　2)Bmp565　　3)LjxHeightBmp　　4)LjxLuminanceBmp

其中，LjxHeightBmp和LjxLuminanceBmp为3D图像处理控制器LJ-X8000专用的图像格式，一般情况下不使用

②函数列表

SaveAsImage	将高度/浓淡数据保存为图像格式（Tiff、Bmp565、LjxBmp）文件
SaveAsCsv	将高度/浓淡数据保存为Csv格式文件
SaveProfileInfo	将轮廓信息保存为Csv格式文件

SaveAsImage的函数定义　　将高度/浓淡数据保存为图像格式（Tiff、Bmp565、LjxBmp）文件

此函数与SaveAsCsv有2个重载函数，以下是各个重载的定义

VB.NET

```
Function LJX8000A.SaveReceiveDataAsFile.SaveAsImage(Datas As UShort(), Width As Integer, filePath As String, type As
LJX8000A.SaveReceiveDataAsFile.ImageSaveType, ZResolutionIn As Double, ZResolutionOut As Double) As Boolean (+ 1 重载)
Function LJX8000A.SaveReceiveDataAsFile.SaveAsImage(Datas As UShort(), Width As Integer, filePath As String, type As
LJX8000A.SaveReceiveDataAsFile.ImageSaveType) As Boolean (+ 1 重载)
```

C#

```
Function LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(Datas As UShort(), Width As Integer, filePath As String) As Boolean (+ 1 重载)
Function LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(Datas As UShort(), Width As Integer, filePath As String,
HeightResolution As Double, ZeroValue As UShort) As Boolean (+ 1 重载)
```

输入参数	描述
Datas	需要保存的数据，格式为ushort数组
Width	图像的宽度（每行轮廓的数据点数）
filePath	保存的文件路径及文件名称
type	保存的目标格式，参照枚举类型ImageSaveType
ZResolutionIn(可选)	保存格式为LjxBmp时调整高度比例系数的差异造成的数
ZResolutionOut（可选）	保存格式为LjxBmp时调整高度比例系数的差异造成的数

注意点：

①输入的数据为Ushort数组，Byte数组不可以

②Tiff和Bmp565像素数据格式为Ushort，Bmp565分为了RGB三通道（各5bit、6bit、5bit）

SaveAsCsv的函数定义　　将高度/浓淡数据保存为Csv格式文件

此函数与SaveAsImage有2个重载函数，以下是各个重载的定义

VB.NET

```
Function LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(Datas As UShort(), Width As Integer, filePath As String) As Boolean (+ 1 重载)
Function LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(Datas As UShort(), Width As Integer, filePath As String, HeightResolution As Double) As Boolean (+ 1 重载)
```

C#

```
bool LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(ushort[] Datas, int Width, string filePath) (+ 1 重载)
bool LJX8000A.SaveReceiveDataAsFile.SaveAsCsv(ushort[] Datas, int Width, string filePath, double HeightResolution) (+ 1 重载)
```

输入参数	描述
Datas	需要保存的数据，格式为ushort数组
Width	图像的宽度（每行轮廓的数据点数）
filePath	保存的文件路径及文件名称
HeightResolution（可选）	高度分辨率，需要以mm或um单位保存数值时输入

注意点：

①输入的数据为Ushort数组，Byte数组不可以

②Csv内需要保存为实际高度数值时，需要输入HeightResolution，计算方法如下

保存值 =（原始数值 - 32768）* HeightResolution

SaveProfileInfo的函数定义　　将轮廓信息保存为Csv格式文件

VB.NET

```
Function LJX8000A.SaveReceiveDataAsFile.SaveProfileInfo(Profinfo As LjxaSample.LJX8IF.LJX8IF_PROFILE_INFO, filePath As String) As Boolean
```

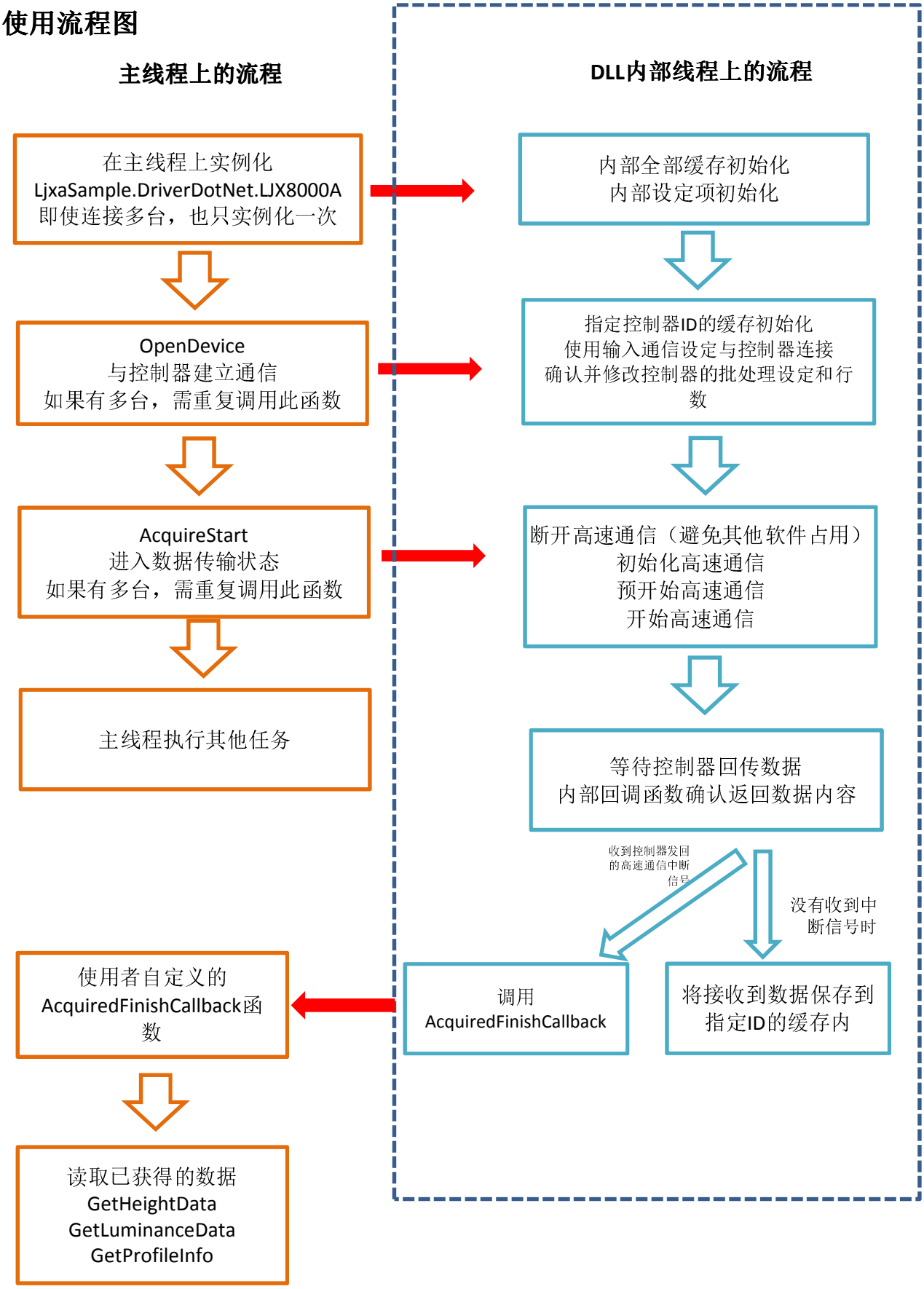
C#

```
bool LJX8000A.SaveReceiveDataAsFile.SaveProfileInfo
(LjxaSample.LJX8IF.LJX8IF_PROFILE_INFO Profinfo, string filePath)
```

输入参数	描述
Profinfo	输入轮廓信息
filePath	保存的文件路径及文件名称

7、建议使用流程

7-1 使用流程图



7-2 参考的代码（C#）

```
using namespace LjxaSample.DriverDotNet
```
使用 LjxaSample.DriverDotNet命名空间

```
private LJX8000A LJXA = new LJX8000A();
```
实例化LJX8000A，实例名称：

```
private void OpenDeviceAndAcquireStart()
{
    CommunicationSetting _CommunicationSetting0
    = new CommunicationSetting(new byte[] { 192, 168, 0, 1 }, 24691, 24692, 1000, false, true);
    LJXA.OpenDevice(0, _CommunicationSetting, AcquiredFinish);
    LJXA.AcquireStart(0);
    //CommunicationSetting _CommunicationSetting1
    //= new CommunicationSetting(new byte[] { 192, 168, 0, 2 }, 24691, 24692, 1000, false, true);
    //LJXA.OpenDevice(1, _CommunicationSetting1, AcquiredFinish);
    //LJXA.AcquireStart(1);
}
```
指定IP地址：192.168.0.1，端口号24691，高速端口24692，行数1000行

打开设备连接，编号为0，指定回调函数为

开始高速通信，并发送批处理开始的信号（如果需要）

需要使用2台控制器时，参考此部分的代码建立通信

```
private void CloseDevice()
{
    LJXA.CloseDevice(0);
    //LJXA.CloseDevice(1);
}
```
清空指定ID的缓存数据，并断开控制器连接

多台控制器时，需要分别断开

```
private void AcquiredFinish(int DeviceID, int Notify)
{
//在Callback函数内调用希望的方法
    ushort[] _heightData = new ushort[] { };
    ushort[] _luminanceData = new ushort[] { };
    int _width = 0;
    int _xPitch = 0;
    LJXA.GetHeightData(0, ref _heightData, ref _width, ref _xPitch);
    LJXA.GetLuminanceData(0, ref _luminanceData);
     string _startPath = Application.StartupPath;
     string _filenameH = _startPath + "/DeviceID"+ Convert.ToString(DeviceID) + "_height.tif";
     string _filenameL = _startPath + "/DeviceID"+ Convert.ToString(DeviceID) + "_luminance.tif";
    LJX8000A.SaveReceiveDataAsFile.SaveAsImage(_heightData, _width, , LJX8000A.SaveReceiveDataAsFile.ImageSaveType.Tiff);
    LJX8000A.SaveReceiveDataAsFile.SaveAsImage(_luminanceData, _width, , LJX8000A.SaveReceiveDataAsFile.ImageSaveType.Tiff);
}
```
回调函数，DLL读取数据完成时，会调用此函数

不同ID号的控制器，可以使用同一个回调函数（不同当然也可

读取DLL保存的高度/浓淡数据及宽度，X方向点间隔

将读取结果保存到电脑上，保存路径为程序运行文件夹

ID0 的控制器发回的数据，保存文件名为： DeviceID0_height.tif DeviceID0_luminance.tif

ID1 的控制器发回的数据，保存文件名为： DeviceID1_height.tif DeviceID1_luminance.tif

7-3 参考的代码（VB.NET）

```
Imports LjxaSample.DriverDotNet                                    使用 LjxaSample.DriverDotNet命名空间

private  LJXA  as LJX800A = new LJX8000A()                         实例化LJX8000A，实例名称：LJXA

private sub OpenDeviceAndAcquireStart()                            指定IP地址：192.168.0.1，端口号24691，高速端口24692，行数1000行
    dim _CommunicationSetting0 as CommunicationSetting
    = new CommunicationSetting(new byte[ 192, 168, 0, 1 ], 24691, 24692, 1000, false, true)
    LJXA.OpenDevice(0, _CommunicationSetting, AcquiredFinish)      打开设备连接，编号为0，指定回调函数为

    LJXA.AcquireStart(0)                                           开始高速通信，并发送批处理开始的信号（如果需要）

    'dim _CommunicationSetting1 as CommunicationSetting
    '= new CommunicationSetting(new byte() { 192, 168, 0, 2 }, 24691, 24692, 1000, false, true)
    'LJXA.OpenDevice(1, _CommunicationSetting1, AcquiredFinish)
    'LJXA.AcquireStart(1)                                          需要使用2台控制器时，参考此部分的代码建立通信
end sub

private sub CloseDevice()
    LJXA.CloseDevice(0)                                            清空指定ID的缓存数据，并断开控制器连接
    'LJXA.CloseDevice(1)                                           多台控制器时，需要分别断开
end sub

private sub AcquiredFinish(DeviceID as integer, Notify as integer)     回调函数，DLL读取数据完成时，会调用此函数
    '在Callback函数内调用希望的方法
    dim _heightData as ushort()= new ushort() { }
    dim  _luminanceData as ushort() = new ushort() { }
    dim _width as integer = 0
    dim _xPitch as integer = 0;
    LJXA.GetHeightData(0, _heightData, _width, ref _xPitch)        读取DLL保存的高度/浓淡数据及宽度，X方向点间隔
    LJXA.GetLuminanceData(0, _luminanceData)
    dim _startPath as  string = Application.StartupPath
    dim _filenameH as string = _startPath & "/DeviceID" & Convert.ToString(DeviceID) & "_height.tif"
    dim _filenameL as string = _startPath & "/DeviceID" & Convert.ToString(DeviceID) & "_luminance.tif"
    LJX8000A.SaveReceiveDataAsFile.SaveAsImage(_heightData, _width, , LJX8000A.SaveReceiveDataAsFile.ImageSaveType.Tiff)
    LJX8000A.SaveReceiveDataAsFile.SaveAsImage(_luminanceData, _width, , LJX8000A.SaveReceiveDataAsFile.ImageSaveType.Tiff)
end sub
```

将读取结果保存到电脑上，保存路径为程序运行文件夹

ID0 的控制器发回的数据，保存文件名为： DeviceID0_height.tif　　DeviceID0_luminance.tif

ID1 的控制器发回的数据，保存文件名为： DeviceID1_height.tif　　DeviceID1_luminance.tif