



# 程序示例说明书

JHCap2 SDK 程序示例包含 C#, VB.net, VC, VB, QT 等语言的代码，演示如何使用回调函数使用单相机或者使用线程方式使用单相机，多相机，调整相机参数，以及如何与 OpenCV 等集成使用。可以作为使用 SDK 的参考，或者作为具体视觉应用开发的基础框架使用。

程序示例说明书 .....	I
1 CSharp .....	1
1.1 OneCamera_CSharpDemo .....	1
1.1.1 功能介绍 .....	1
1.1.2 代码示例 .....	2
1.2 OneCamera_CSharpDemo_Thread .....	3
1.2.1 功能介绍 .....	3
1.2.2 代码示例 .....	4
1.3 TwoCamera_CSharpDemo .....	6
1.3.1 功能介绍 .....	6
1.3.2 代码示例 .....	7
2 VB.net .....	9
2.1 OneCamera_VBNetDemo .....	9
2.1.1 功能介绍 .....	9
2.1.2 代码示例 .....	10
2.2 OneCamera_VBNetDemo_Thread .....	11
2.2.1 功能介绍 .....	11
2.2.2 代码示例 .....	12
2.3 TwoCamera_VBNetDemo .....	14
2.3.1 功能介绍 .....	14
2.3.2 代码示例 .....	15
3 VC .....	17
3.1 OneCamera_VCDemo .....	17
3.1.1 功能介绍 .....	17
3.1.2 代码示例 .....	18
3.2 OneCamera_VCDemo_Thread .....	19
3.2.1 功能介绍 .....	19
3.2.2 代码示例 .....	21
3.3 TwoCamera_VCDemo .....	23
3.3.1 功能介绍 .....	23
3.3.2 代码示例 .....	24
3.4 FourCamera_VCDemo .....	27
3.4.1 功能介绍 .....	27
3.4.2 代码示例 .....	27



3.5 OpenCV .....	29
3.5.1 功能介绍 .....	29
3.5.2 代码示例 .....	29
4 VB .....	31
4.1 OneCamera_VBDemo .....	31
4.1.1 功能介绍 .....	31
4.1.2 代码示例 .....	32
5 QT .....	34
5.1 OneCamera_QtDemo .....	34
5.1.1 功能介绍 .....	34
5.1.2 代码示例 .....	35
6 Demo .....	37
5.1 Light12bit .....	37
5.1.1 功能介绍 .....	37
5.1.2 功能实现说明 .....	37
5.1.3 代码示例 .....	38
5.1 LineScan .....	39
5.1.1 功能介绍 .....	39
5.1.2 功能实现说明 .....	40
5.1.3 代码示例 .....	40

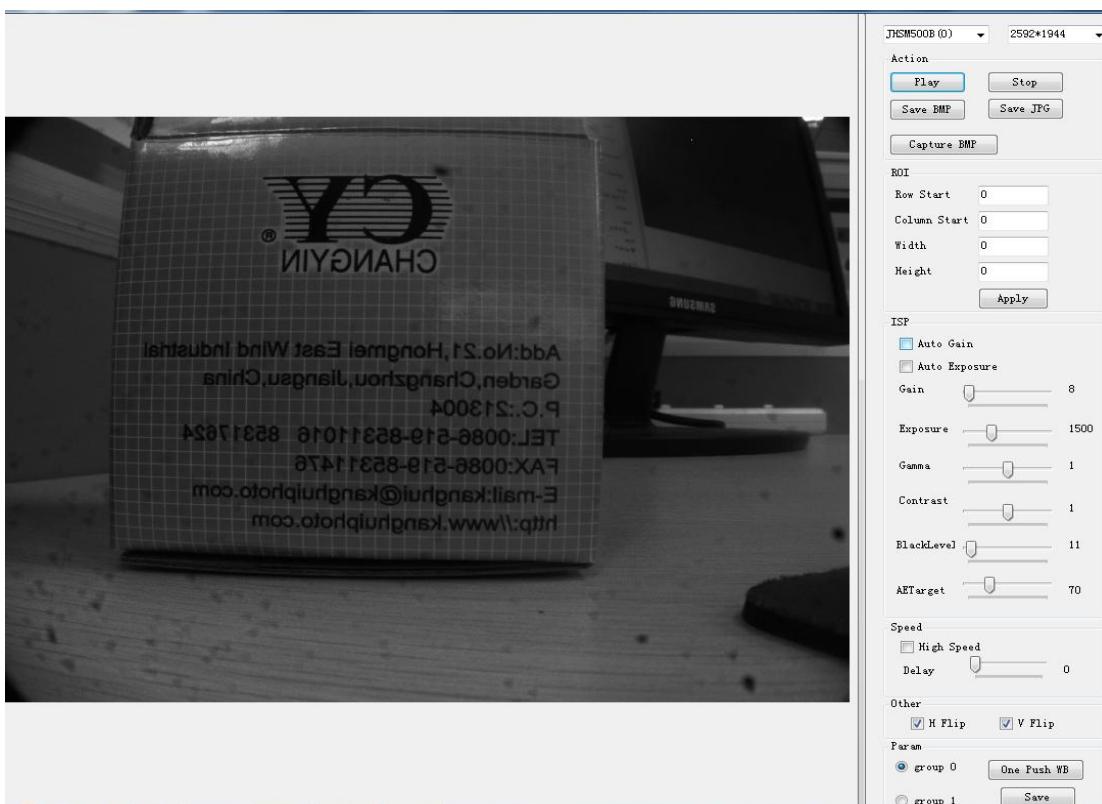


# 1 CSharp

## 1.1 OneCamera\_CSharpDemo

### 1.1.1 功能介绍

OneCamera\_CSharpDemo 这个示例实现的功能主要包括：对单个相机采集的图像进行显示，保存图片，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度，相机黑电平，相机自动曝光目标值），设置数据传输模式，设置延时时间，水平和垂直镜像操作，实现一键白平衡，参数保存等。



### 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 保存图片：可对界面显示的图像信息进行截取并保存，点击 SaveBMP 把当前显示画面保存为一张 BMP 图，点击 SaveJPG 把当前显示画面保存为一张 JPG 图，点击 Capture BMP 保存内存数据为一张 BMP 图。

3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。



4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2；Blacklevel 对应为相机黑电平值，范围为 0~255；AE Target 对应为自动曝光目标值，范围为 0~255。同时只有使用自动曝光功能时才需调节自动曝光目标值，当设置好目标值后，自动曝光的稳定值会不断接近目标值。

5 数据传输模式：通过选定 High Speed 控件就能把数据传输模式设为高速，反之为低速。

6 设置延时时间：通过滑动 Delay 对应的滑动条可以设置延时时间

7 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

8 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

9 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可进行两组参数之间的切换，点击 Save 可对对应的参数组进行保存。

### 1.1.2 代码示例

#### 初始化获取相机列表

```
int m = 0;  
JHCap.CameraGetCount(ref m); //获取已连接的相机个数  
StringBuilder name = new StringBuilder();  
StringBuilder model = new StringBuilder();  
//依次获取相机分辨率名字  
for (int i = 0; i < m; i++)  
{  
    JHCap.CameraGetName(i, name, model);  
    ... //获取相机列表后，显示相机列表  
}  
... //可根据需要添加其他功能代码
```

#### 相关 API 函数：

**CameraGetCount**  
**CameraGetName**

#### 获取分辨率列表并为相机设置分辨率

```
int reso_width = 0, reso_height = 0, reso_count = 0,  
JHCap.CameraGetResolutionCount(m_index, ref reso_count); //获取相机m_index分辨率种数  
//依次获取相机m_index分辨率  
for (int j = 0; j < reso_count; j++)  
{  
    JHCap.CameraGetResolution(m_index, j, ref reso_width, ref reso_height);  
    ... //获取相机分辨率列表后，显示分辨率列表
```



```
}

... //可根据需要添加其他功能代码
JHCap.CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨率
为第index组分辨率*/
```

相关 API 函数：

**CameraGetResolutionCount**  
**CameraGetResolution**  
**CameraSetResolution**

## 图像显示

启动

```
JHCap.CameraPlay(g_index, pictureBox1.Handle, callback);
//在句柄为pictureBox1.Handle的窗口显示相机g_index1的画面, callback为回调函数
```

回调函数

```
//回调函数定义
private static int Callback(IntPtr pImageBuffer, int width, int height, int format)
{
    //用户可添加代码处理获取的图像, 图像信息存放在pImageBuffer中
    return 0;
}
```

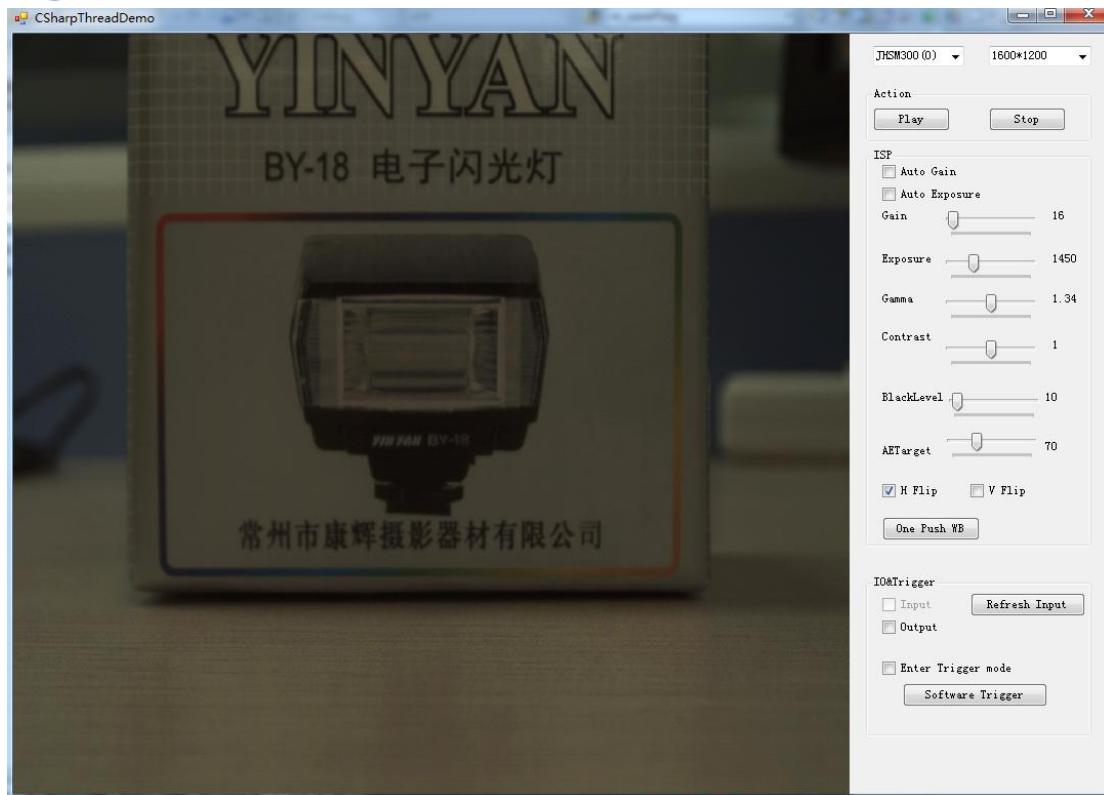
相关 API 函数：

**CameraPlay**

## 1.2 OneCamera\_CSharpDemo\_Thread

### 1.2.1 功能介绍

OneCamera\_CSharpDemo\_Thread 这个示例实现的功能主要包括：对单个相机采集的图像运用线程的方式进行显示，设置相机参数（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度，相机黑电平和自动曝光目标值），水平和垂直镜像操作，实现一键白平衡，相机 GPIO 设置，软件触发拍摄图像等。



### 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2；Blacklevel 对应为相机黑电平值，范围为 0~255；AE Target 对应为自动曝光目标值，范围为 0~255。同时只有使用自动曝光功能时才需调节自动曝光目标值，当设置好目标值后，自动曝光的稳定值会不断接近目标值。

3 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

4 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

5 GPIO 设置：选定 IO&Trigger 模块的 Output 或者 Input 控件，再点击 Refresh Input 按钮就能设置相机的 GPIO，其中 Output 对应为 Out，Input 对应为 In。

6 通过软件触发拍摄图像：选定 Enter Trigger Mode 就能使相机进入触发模式，点击 Software Trigger 按钮就能通过软件触发去拍摄图像。

### 1.2.2 代码示例

#### 初始化获取相机列表

```
int m = 0;  
JHCap.CameraGetCount(ref m); // 获取已连接相机的个数
```



```
StringBuilder name = new StringBuilder();
StringBuilder model = new StringBuilder();
//依次获取相机分辨率名字
for (int i = 0; i < m; i++)
{
    JHCap.CameraGetName(i, name, model);
    ...//获取相机列表后，显示相机列表
}
... //可根据需要添加其他功能代码
```

相关 API 函数：

**CameraGetCount**  
**CameraGetName**

### 开始采集和显示图像

开启一个后台线程，循环调用 CameraQueryBitmap 可获取图像信息。线程通过 delegate 的方式才能更新界面元素。

```
public delegate void UpdateData(Bitmap bmp);
private void Process()
{
    while (true)
    {
        if (g_work)
        {
            Bitmap bmp = JHCap.CameraQueryBitmap(g_index, JHCap.CAMERA_IMAGE_BMP,
false);
            if (bmp!=null) updateFrame(bmp);
        }
        else
            Thread.Sleep(20);
    }
}

private void updateFrame(Bitmap bmp)
{
    if (this.InvokeRequired)
    {
        this.Invoke(new UpdateData(updateFrame), new object[] { bmp });
    }
    else
    {
        pictureBox1.Image = bmp;
        GC.Collect();
    }
}
```



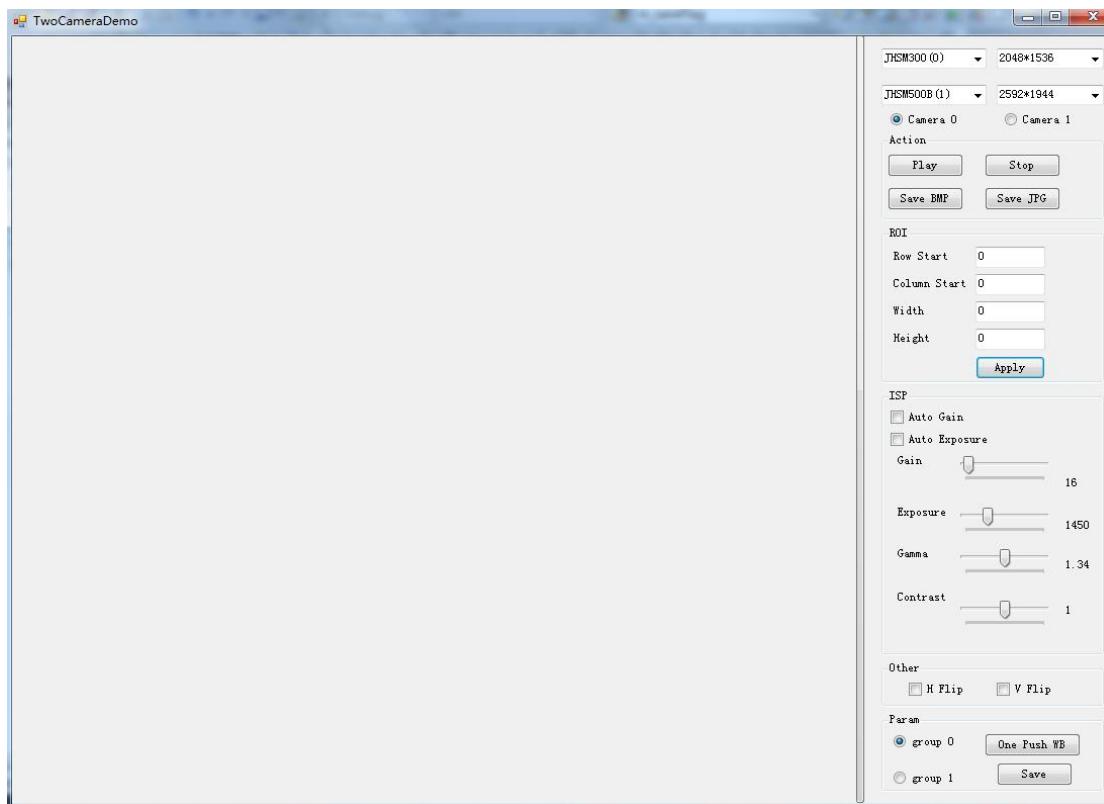
相关 API 函数：

`CameraQueryBitmap`

## 1.3 TwoCamera\_CSharpDemo

### 1.3.1 功能介绍

TwoCamera\_CSharpDemo 这个示例实现的功能主要包括：对两个相机采集的图像进行显示，保存图像，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度），水平和垂直镜像操作，一键白平衡，参数保存等。



### 功能实现说明

1 图像的显示：通过右上方的四个下拉按钮可以分别对两组相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示，通过选定 Camera0 或者 Camera1 对需要操作的相机进行选定。

2 保存图像：选定相机后，点击 Save BMP 按钮能保存 BMP 格式的图片，点击 Save JPG 按钮能保存 JPG 格式的图片。

3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。



其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

5 水平和垂直镜像操作：选定 Hflip 可实现相机的水平镜像操作，选定 Vflip 可实现垂直镜像操作。

6 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

7 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可实现两组参数之间的切换，点击 Save 可对对应的参数组进行保存。

### 1.3.2 代码示例

#### 获取相机列表及初始化相机

```
int m = 0;      //相机个数
JHCap.CameraGetCount(ref m);

StringBuilder name = new StringBuilder();
StringBuilder model = new StringBuilder();

//依次获取相机分辨率名字，及初始化相机
for (int i = 0; i < m; i++)
{
    JHCap.CameraGetName(i, name, model);
    JHCap.CameraInit(i);
    JHCap.CameraSetHighspeed(i, false);
    ...//获取相机列表后，显示相机列表
}
... //可根据需要添加其他功能代码
```

相关 API 函数：

CameraGetCount  
CameraGetName  
CameraInit  
CameraSetHighspeed

#### 获取分辨率列表并为相机设置分辨率

```
int reso_width = 0, reso_height = 0, reso_count = 0,
JHCap.CameraGetResolutionCount(m_index, ref reso_count); //获取相机m_index分辨率种数
//依次获取相机m_index分辨率
for (int j = 0; j < reso_count; j++)
{
    JHCap.CameraGetResolution(m_index, j, ref reso_width, ref reso_height);
    ...//获取相机分辨率列表后，显示分辨率列表
}
... //可根据需要添加其他功能代码
JHCap.CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨率
为第index组分辨率*/
```

相关 API 函数：

**CameraGetResolutionCount****CameraGetResolution****CameraSetResolution**

## 图像显示

### 启动

```
JHCap.CameraPlay(g_index1, pictureBox1.Handle, callback);  
//在句柄为pictureBox1.Handle的窗口显示相机g_index1的画面, callback为回调函数  
JHCap.CameraPlay(g_index2, pictureBox2.Handle, callback);  
//在句柄为pictureBox2.Handle的窗口显示相机g_index2的画面, callback为回调函数
```

### 回调函数

```
//回调函数定义  
private static int Callback(IntPtr pImageBuffer, int width, int height, int format)  
{  
    //可添加代码处理获取的图像, 图像信息存放在pImageBuffer中  
    return 0;  
}
```

相关 API 函数:

**CameraPlay**

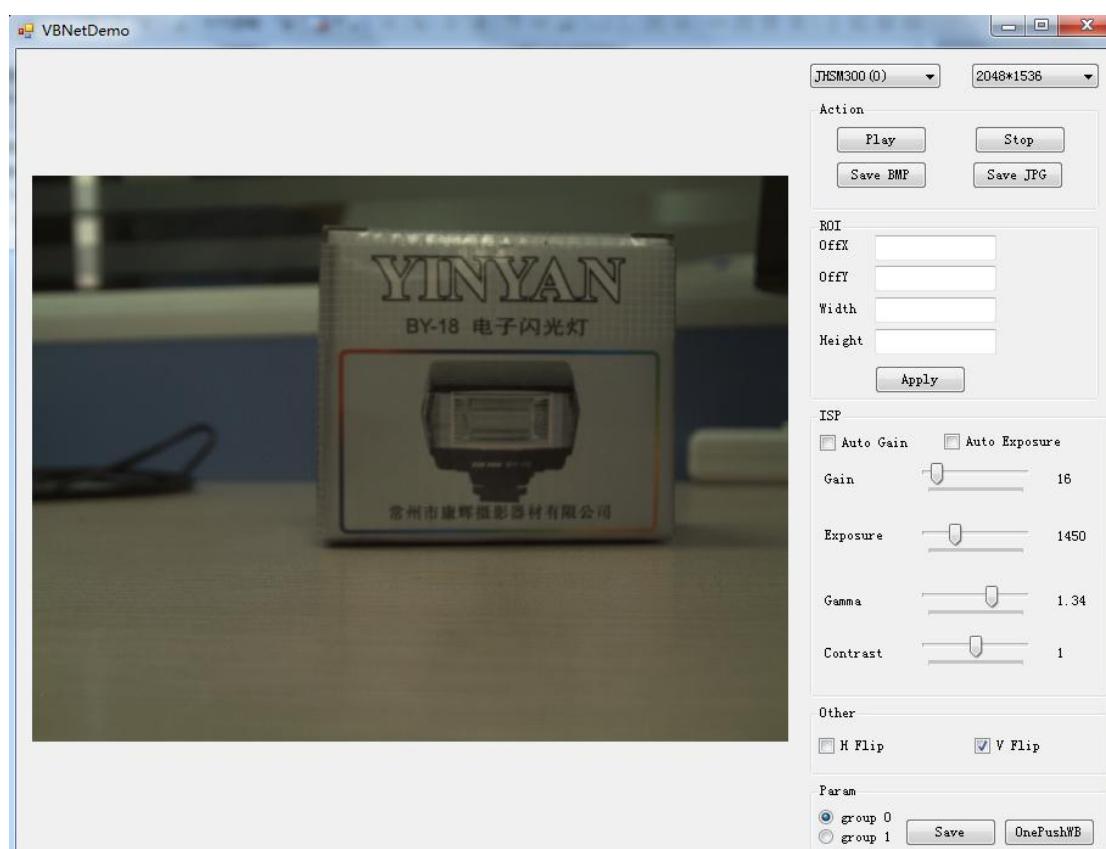


## 2 VB.net

### 2.1 OneCamera\_VBNetDemo

#### 2.1.1 功能介绍

OneCamera\_VBNetDemo 这个示例实现的功能主要包括：对单个相机采集的图像进行显示，保存图片，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度），水平和垂直镜像操作，参数保存，一键白平衡等。



#### 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 保存图片：可对界面显示的图像信息进行截取并保存，点击 Save BMP 能把当前显示的图像保存为 BMP 格式的图片，点击 Save JPG 能把当前显示的图像保存为 JPG 格式的图片。

3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 offX、offY、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，offX 为 ROI 的起始水平位置；offY 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和



Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

5 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

6 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可进行两组参数之间的切换，点击 Save 可对对应的参数组进行保存。

7 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

### 2.1.2 代码示例

#### 获取相机列表及初始化相机

```
Dim cameraCount As Integer      //已连接的相机总数
//获取相机总数
Try
    JHCap.CameraGetCount(cameraCount)
Catch ex As Exception
    MessageBox.Show("Form load exception")
End Try
//初始化所有相机
If cameraCount > 0 Then        //相机总数大于0
    JHCap.CameraInit(0)         //初始化第0个相机
    cameraCount = cameraCount - 1
End If
//依次获取相机名字
Dim name As New StringBuilder
Dim mode As New StringBuilder
For i = 0 To cameraCount - 1
    JHCap.CameraGetName(i, name, mode)      //获取相机名字
    ... //获取相机列表后，显示相机列表或加入其他功能代码
Next
```

相关 API 函数：

**CameraGetCount**  
**CameraInit**  
**CameraGetName**

#### 获取相机分辨率

```
Dim resoCount As Integer      //分辨率总数
Dim reso_width As Integer     //分辨率宽度
Dim reso_height As Integer    //分辨率高度
//依次获取相机分辨率
```



```
JHCap.CameraGetResolutionCount(index, resoCount) //获取相机index的分辨率总数  
For j = 0 To resoCount - 1  
    JHCap.CameraGetResolution(index, j, reso_width, reso_height)  
    ... //获取相机分辨率后，显示相机分辨率或加入其他功能代码  
Next
```

相关 API 函数：

**CameraGetResolutionCount**  
**CameraGetResolution**

## 图像采集和显示

启动

```
Try  
    JHCap.CameraPlay(camera_index, PictureBox1.Handle, CallFunction) /*在句柄为  
    PictureBox1.Handle的窗口显示相机camera_index的画面, SnapThreadCallback为回调函数  
*/  
Catch ex As Exception  
    MessageBox.Show("CallBack exception")  
End Try
```

回调函数

```
Private Function CallFunction(ByVal buf As IntPtr, ByVal width As Integer, ByVal height  
As Integer, ByVal method As Integer) As Integer //回调函数定义  
    //用户可添加代码处理获取的图像，图像信息存放在pImageBuffer中  
    Return 0  
End Function
```

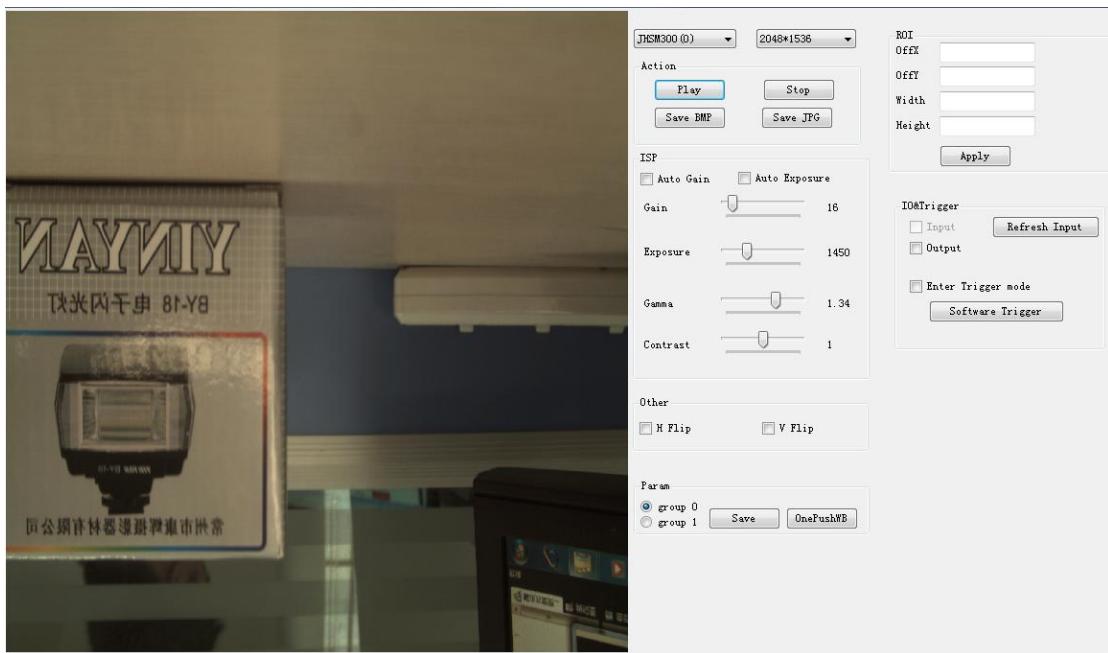
相关 API 函数：

**CameraPlay**

## 2.2 OneCamera\_VBNetDemo\_Thread

### 2.2.1 功能介绍

OneCamera\_VBNetDemo\_Thread 这个示例实现的功能主要包括：对单个相机采集的图像引用线程的方式进行显示，保存图片，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度），水平和垂直镜像操作，参数保存，一键白平衡，GPIO 设置和软件触发拍摄图像等。



### 功能实现说明

- 1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。
- 2 保存图片：可对界面显示的图像信息进行截取并保存，点击 Save BMP 把当前显示的图像保存为 BMP 格式的图片，点击 Save JPG 把当前显示的图像保存为 JPG 格式的图片。
- 3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 offX、offY、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。  
其中，offX 为 ROI 的起始水平位置；offY 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。
- 4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。  
其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。
- 5 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。
- 6 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可进行两组参数之间的切换，点击 Save 可对对应的参数组进行保存。
- 7 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。
- 8 GPIO 设置：选定 IO&Trigger 模块的 Output 控件，能设置相机的 GPIO 为 OUT。
- 9 通过软件触发拍摄图像：选定 Enter Trigger Mode 就能使相机进入触发模式，点击 Software trigger 按钮就能通过软件触发去拍摄图像。

### 2.2.2 代码示例

#### 获取相机列表及初始化相机



```
Dim cameraCount As Integer          //已连接的相机总数
Try
    JHCap.CameraGetCount(cameraCount)      //获取相机总数
Catch ex As Exception
    MessageBox.Show("Form load exception")
End Try
//初始化所有相机
If cameraCount > 0 Then           //相机总数大于0
    JHCap.CameraInit(0)            //初始化第0个相机
    cameraCount = cameraCount - 1
End If
//依次获取相机名字
Dim name As New StringBuilder
Dim mode As New StringBuilder
For i = 0 To cameraCount - 1
    JHCap.CameraGetName(i, name, mode)    //获取相机名字
    ... //获取相机列表后，显示相机列表或加入其他功能代码
Next
```

相关 API 函数：

**CameraGetCount**  
**CameraInit**  
**CameraGetName**

### 获取相机分辨率

```
Dim resoCount As Integer          //分辨率总数
Dim reso_width As Integer        //分辨率宽度
Dim reso_height As Integer       //分辨率高度
JHCap.CameraGetResolutionCount(index, resoCount)    //获取相机index的分辨率总数
//依次获取相机分辨率
For j = 0 To resoCount - 1
    JHCap.CameraGetResolution(index, j, reso_width, reso_height)
    ... //获取相机分辨率后，显示相机分辨率或加入其他功能代码
Next
```

相关API函数：

**CameraGetResolutionCount**  
**CameraGetResolution**

### 图像采集和显示

开启一个后台线程，循环调用 CameraQueryBitmap 获取图像。线程通过 delegate 的方式才能更新界面元素。

```
Public Delegate Sub UpdateData(ByVal bmp As Bitmap)

Private Sub Process()
```



```
While True      '无限循环·
    If m_work Then      '工作标志
        Dim bmp As Bitmap
        bmp = JHCap.CameraQueryBitmap(camera_index, JHCap.CAMERA_IMAGE_BMP,
    False)
        If bmp IsNot Nothing Then updateFrame(bmp)
    Else
        System.Threading.Thread.Sleep(20)
    End If
End While
End Sub

Private Sub updateFrame(ByVal bmp As Bitmap)
    If (Me.InvokeRequired) Then
        Me.Invoke(New UpdateData(AddressOf updateFrame), New Object() {bmp})
    Else
        PictureBox1.Image = bmp
        GC.Collect()
    End If
End Sub
```

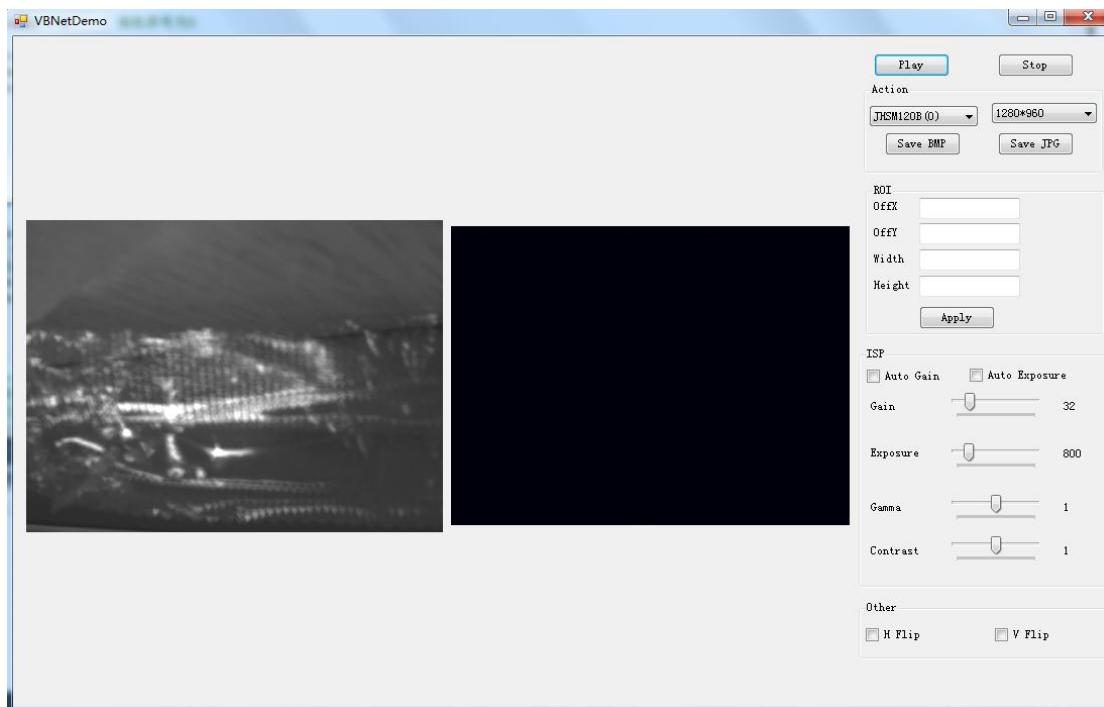
相关 API 函数:

**CameraQueryBitmap**

## 2.3 TwoCamera\_VBNetDemo

### 2.3.1 功能介绍

TwoCamera\_VBNetDemo 这个示例实现的功能主要包括：对双台相机采集的图像运用回调函数的方式进行显示，保存图片，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度），水平和垂直镜像操作等。



### 功能实现说明

1 图像的显示：启动程序后点击 Play 按钮，就能实现双台相机的图像显示功能，点击 Stop 按钮停止显示，两个下拉框分别用于选定某个相机和调节分辨率。

2 保存图片：可对界面显示的图像信息进行截取并保存，点击 Save BMP 把当前显示的图像保存为 BMP 格式的图片，点击 Save JPG 把当前显示的图像保存为 JPG 格式的图片。

3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 offX、offY、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，offX 为 ROI 的起始水平位置；offY 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

5 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

### 2.3.2 代码示例

#### 保存 BMP 格式图像

```
SaveFileDialog1.InitialDirectory = "C:\\" ' 获取保存路径
SaveFileDialog1.Filter = "bmp files (*.bmp) | *.bmp"
SaveFileDialog1.FilterIndex = 1
SaveFileDialog1.RestoreDirectory = True
SaveFileDialog1.Title = "保存"
SaveFileDialog1.DefaultExt = "bmp"
If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
```



```
Dim filename As String  
filename = SaveFileDialog1.FileName  
JHCap.CameraSaveBMPB(camera_index, filename) '保存 BMP 图片  
End If
```

相关 API 函数：

**CameraSaveBMPB**

保存 JPG 格式图像

```
SaveFileDialog1.InitialDirectory = "C:\\"      '获取保存路径  
SaveFileDialog1.Filter = "jpg files (*.jpg)|*.jpg"  
SaveFileDialog1.FilterIndex = 1  
SaveFileDialog1.RestoreDirectory = True  
SaveFileDialog1.Title = "保馈?存?"  
SaveFileDialog1.DefaultExt = "jpg"  
If SaveFileDialog1.ShowDialog() = DialogResult.OK Then  
    Dim filename As String  
    filename = SaveFileDialog1.FileName  
    JHCap.CameraSaveJpegB(camera_index, filename, True) '保存 JPG 图片  
End If
```

相关 API 函数：

**CameraSaveJpegB**

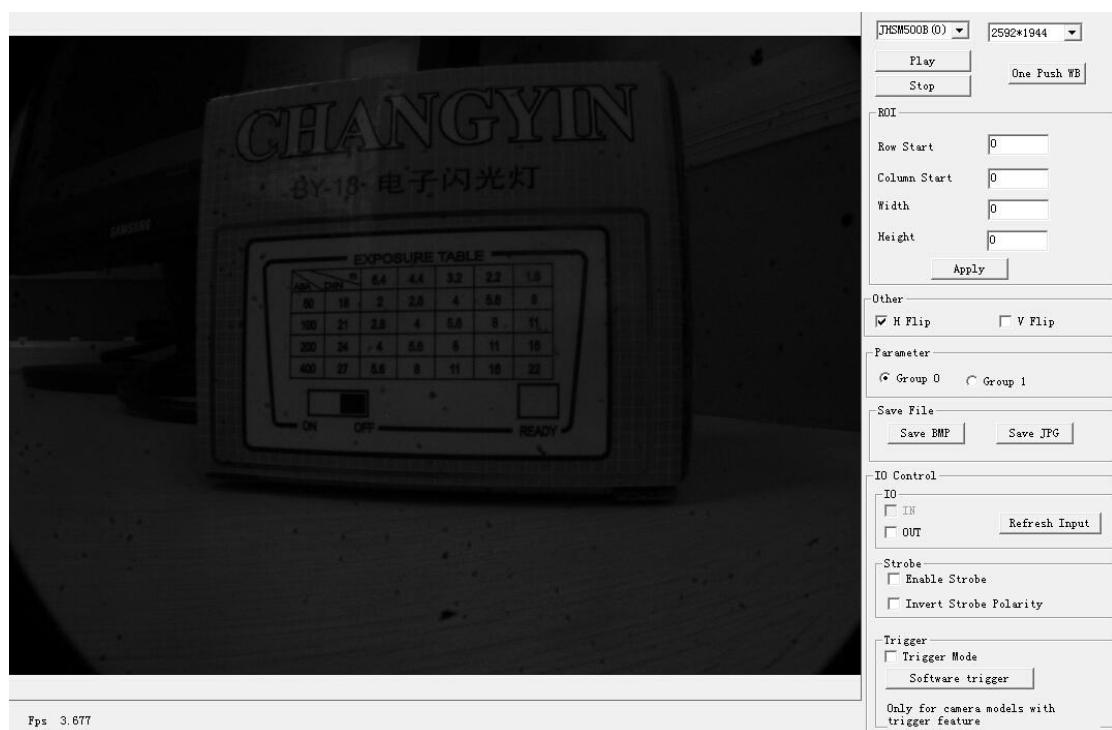


## 3 VC

### 3.1 OneCamera\_VCDemo

#### 3.1.1 功能介绍

OneCamera\_VCDemo 这个示例实现的功能主要包括：直接对相机采集的图像进行显示，实现一键白平衡，保存图片，ROI 设置，水平和垂直镜像操作，参数保存，相机 GPIO 设置，闪光灯设置，软件触发拍摄图像等。



#### 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

3 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

4 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

5 参数保存功能：支持两组参数的保存功能，点击 Group1 和 Group2 可进行两组参数



之间的切换。

6 保存图片：可对界面显示的图像信息进行截取并保存，其中点击 Save BMP 可保存 BMP 格式的图片，点击 Save JPG 可保存 JPG 格式的图片。

7 GPIO 设置：选定 IO 模块的 OUT 控件，能设置相机的 GPIO。

8 闪光灯设置：选定 Enable Strobe 使用闪光灯，选定 Invert Strobe Polarity 能设置闪光灯的输出极性。

9 通过软件触发拍摄图像：选定 Trigger Mode 使相机进入触发模式后，点击 Software trigger 按钮就能通过软件触发去拍摄图像。

### 3.1.2 代码示例

#### 初始化获取相机列表

```
int CamAllNum=0;//相机个数
CameraGetCount (&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//依次获取相机名字
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ... //获取相机列表后，显示相机列表
    }
    ...//可根据需要添加其他功能代码
}
else
    return false;
delete[] model;
model=NULL;
delete[] name;
name=NULL;
```

相关API函数：

**CameraGetCount**  
**CameraGetName**

#### 获取分辨率列表并设置相机分辨率

```
int index=0,width=0,height=0,camera_count=0;
CameraGetResolutionCount(m_index, &camera_count); //获取相机m_index的分辨率种数
//依次获取相机分辨率
for(int i=0;i<camera_count;i++)
{
    CameraGetResolution( m_index, i,&width, &height);
    ...//获取相机分辨率列表后，显示分辨率列表
}
```



```
... //可根据需要添加其他功能代码  
CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨率  
为第index组分辨率*/
```

相关 API 函数：

```
CameraGetResolutionCount  
CameraGetResolution  
CameraSetResolution
```

开始采集和显示图像

启动

```
CameraPlay(m_device_id, hImage[m_device_id], SnapThreadCallback);  
/*在句柄为hImage[m_device_id]的窗口显示相机m_device_id的画面，SnapThreadCallback为  
回调函数，可用0替代*/
```

停止

```
CameraStop(m_device_id); //停止编号为m_device_id的相机
```

回调函数

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height,  
int format)  
{  
    //用户可添加代码处理获取的图像，图像信息存放在pImageBuffer中  
    return 0;  
}
```

相关 API 函数：

```
CameraPlay  
CameraStop
```

## 3.2 OneCamera\_VCDemo\_Thread

### 3.2.1 功能介绍

OneCamera\_VCDemo\_Thread 这个示例实现的功能主要包括：对单个相机采集的图像信息引用线程的方式进行显示，实现一键白平衡，保存图片，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度，饱和度，相机黑电平和自动曝光目标值），数据传输模式设置，延时时间设置，水平和垂直镜像操作，参数保存等。另外介绍了如何自动重连的方法。



## 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

3 保存图片：可对界面显示的图像信息进行截取并保存，选定 BMP 或者 JPG 可选择被保存图片的格式，然后点击 Save 按钮就能保存当前图像。（其中 BMP 对应图片的 BMP 格式，JPG 对应图片的 JPG 格式）

4 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

5 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2；Saturation 对应为饱和度，范围为 0~2；Blacklevel 对应为相机黑电平值，范围为 0~255；AE Target 对应为自动曝光目标值，范围为 0~255。同时，相机自动曝光目标值只有在使用自动曝光功能时才需调节，当设置好目标值后，自动曝光的稳定值会不断接近目标值。

6 数据传输模式：通过选定 High Speed 控件就能把数据传输模式设为高速，反之为低速。

7 设置延时时间：通过滑动 Delay 对应的滑动条可以设置延时时间，范围为 0~1000 毫秒。

8 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

9 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可进行两组参



数之间的切换，点击 Save 可对对应的参数组进行保存。

10 当相机进行了拔插操作，或者掉线重连操作时，此应用程序将收到 Windows 消息 WM\_DEVICE\_CHANGE，可以自动对相机进行重新初始化启动，并接着工作。

### 3.2.2 代码示例

#### 初始化获取相机列表

```
int CamAllNum=0; //相机个数
CameraGetCount (&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//依次获取相机名字
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ...//获取相机列表后，显示相机列表
    }
    ...//可根据需要添加其他功能代码
}
else
    return false;
delete [] name;
name=NULL;
delete [] model;
model=NULL;
```

相关 API 函数：

**CameraGetCount**  
**CameraGetName**

#### 获取分辨率列表并设置相机分辨率

```
int index=0, width=0, height=0, camera_count=0;
CameraGetResolutionCount(m_index, &camera_count); //获取相机m_index的分辨率种数
//依次获取相机分辨率
for(int i=0;i<camera_count;i++)
{
    CameraGetResolution( m_index, i, &width, &height);
    ...//获取相机分辨率列表后，显示分辨率列表
}
...//可根据需要添加其他功能代码
CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨率
为第index组分辨率*/
```



相关 API 函数：

**CameraGetResolutionCount**  
**CameraGetResolution**  
**CameraSetResolution**

### 开始采集和显示图像

开启一个后台线程，循环调用 CameraQueryImage 获取图像。通过发送 Windows 消息 WM\_PROCESS\_MESSAGE 通知界面线程显示。

```
UINT MyThreadProc( LPVOID pParam )
{
    int id = (int)pParam;

    while(1)
    {
        if(g_work==0) break;
        int m_width=0, m_height=0, len=0;
        CameraGetImageSize(id, &m_width, &m_height);
        CameraGetImageBufferSize(id, &len, CAMERA_IMAGE_BMP);
        unsigned char *m_inBuf = new unsigned char[len];

        TRACE("mem %x\n", m_inBuf);

        if(CameraQueryImage(id, m_inBuf, &len, CAMERA_IMAGE_BMP)==API_OK)
        {
            PicInformation info;
            info.image=m_inBuf;
            info.width=m_width;
            info.height=m_height;
            CDemoDlg *hwnd=(CDemoDlg *)AfxGetMainWnd();
            hwnd->SendMessage(WM_PROCESS_MESSAGE, 0, (LPARAM)&info);
        }
        else
        {
            delete []m_inBuf;
            Sleep(10);
        }
    }

    TRACE("Thread for camera %d end\n", id);
    return 0;
}
```

相关 API 函数：

**CameraGetImageSize**  
**CameraGetImageBufferSize**  
**CameraQueryImage**



## CameraShowBufferImage

### 动态拔插操作

添加 USB 拔插响应消息

```
ON_WM_DEVICECHANGE()
```

添加响应函数，检查设备数量，如果有变化，进行释放相机或者重新初始化操作

```
BOOL CDemoDlg::OnDeviceChange(UINT nEventType,
                               DWORD_PTR dwData)
{
    int count=0;
    CameraGetCount (&count);
    if(count != m_count)
    {
        g_work = false;
        CameraFree(m_index);
        Sleep(500);
        m_count = EnumerateDevice(count);

        if(m_count>0)
        {
            OnSelchangeMulticam();
            updateParam();
            UpdateData(false);

            g_work = true;
            AfxBeginThread(MyThreadProc, (void*)m_index);
        }
    }
    return FALSE;
}
```

相关 API 函数：

**CameraGetCount**

## 3.3 TwoCamera\_VCDemo

### 3.3.1 功能介绍

TwoCamera\_VCDemo 这个示例实现的功能主要包括：可对两个相机采集的图像进行显示，实现一键白平衡，水平和垂直镜像操作，ROI 设置，相机参数设置（包括：自动增益，自动曝光，增益，曝光，伽马值，对比度），参数保存等。



### 功能实现说明

1 图像的显示：通过右上方的四个下拉按钮可以分别对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示，通过选定 Camera0 或者 Camera1 对需要操作的相机进行选定。

2 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

3 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

4 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

5 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益 和 自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

6 参数保存功能：支持两组参数的保存功能，选定 Group1 或者 Group2 可进行两组参数之间的切换，点击 Save 可对对应的参数组进行保存。

### 3.3.2 代码示例

#### 获取相机列表

```
int CamAllNum;//相机个数  
CameraGetCount (&CamAllNum);  
char *name=new char[255];  
char *model=new char[255];  
//依次获取相机分辨率名字  
if(CamAllNum>0)
```



```
{  
    for(int i=0; i<CamAllNum; i++)  
    {  
        CameraGetName(i, name, model);  
        ... //获取相机列表后，显示相机列表  
    }  
    ... //可根据需要添加其他功能代码  
}  
  
delete[] model;  
model=NULL;  
delete[] name;  
name=NULL;
```

相关API函数：

**CameraGetCount**

**CameraGetName**

获取分辨率列表并设置相机分辨率

```
int index=0, width=0, height=0, camera_count=0;  
CameraGetResolutionCount(m_index, &camera_count); //获取相机m_index的分辨率种数  
//依次获取相机分辨率  
for(int i=0; i<camera_count; i++)  
{  
    CameraGetResolution(m_index, i, &width, &height);  
    ... //获取相机分辨率列表后，显示分辨率列表  
}  
... //可根据需要添加其他功能代码  
CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨  
率为第index组分辨率*/
```

相关 API 函数：

**CameraGetResolutionCount**

**CameraGetResolution**

**CameraSetResolution**

开始采集和显示图像

本示例包含两个示例工程分别演示了两种获取图像的方式。

**TwoCamera\_Demo** 回调函数方式获取图像。

**TwoCamera\_Demo\_Thread** 线程方式获取图像。

线程方式：

开启一个后台线程，循环调用 **CameraQueryImage** 获取图像。 显示图像可使用 **CameraShowBufferImage** 或者 Windows API 显示函数。



```
int m_width=0, m_height=0, len=0;
while(1)
{
    if(work)
    {
        CameraGetImageSize(m_device_id, &m_width, &m_height);
        CameraGetImageBufferSize(m_device_id, &len, CAMERA_IMAGE_RGB24);
        unsigned char *m_inBuf = new unsigned char[len];
        if((CameraQueryImage(m_device_id, m_inBuf, &len, CAMERA_IMAGE_BMP)==API_OK))
        {
            ... //正确获取图像后，显示图像
        }
        else
        {
            Sleep(100); //没有成功获取图像，等待
        }
        ...//可根据需要添加其他功能代码
        delete[] m_inBuf;
        m_inBuf=NULL;
    }
}
```

相关API函数：

**CameraGetImageSize**  
**CameraGetImageBufferSize**  
**CameraQueryImage**  
**CameraShowBufferImage**

回调函数方式：

启动

```
CameraPlay(0, hWnd0, SnapThreadCallback); //在句柄为hWnd0的窗口显示相机0的画面
CameraPlay(1, hWnd1, SnapThreadCallback1); //在句柄为hWnd1的窗口显示相机1的画面
```

停止

```
CameraStop(0);
CameraStop(1);
```

回调函数：

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height, int
format)
{
    //用户可添加代码处理获取的图像，图像信息存放在pImageBuffer中
    return 0;
}
```



相关 API 函数：

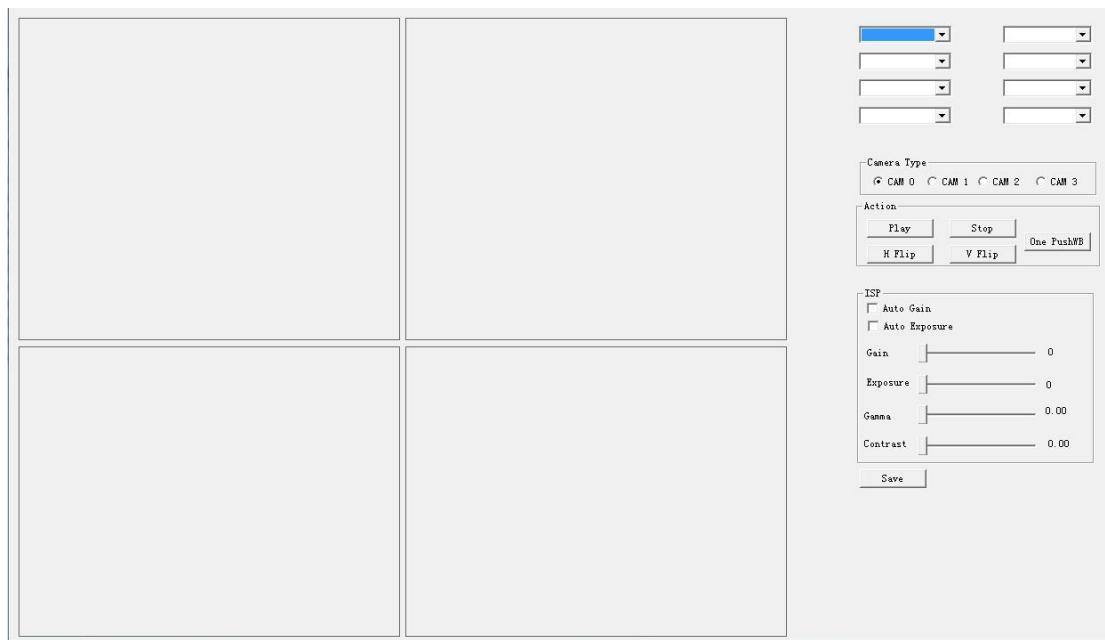
`CameraPlay`

`CameraStop`

## 3.4 FourCamera\_VCDemo

### 3.4.1 功能介绍

FourCamera\_VCDemo 这个示例实现的功能主要包括：对四组相机采集的图像进行显示和操作，可分别实现一键白平衡，水平和垂直镜像操作，相机参数设置（自动增益，自动曝光，增益，曝光，伽马值，对比度）等。



### 功能实现说明

1 图像的显示：通过右上方的八个下拉按钮可以分别对四组相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示，通过选定 CAM0、CAM1、CAM2、CAM3 四个控件可分别对对应的相机进行控制和操作。

2 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡，实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

3 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

4 相机参数设置：通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

### 3.4.2 代码示例

#### 初始化获取相机列表



```
int CamAllNum; //相机个数
CameraGetCount (&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//依次获取相机名字
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ... //获取相机列表后，显示相机列表
    }
    ...//可根据需要添加其他功能代码
}
delete[] model;
model=NULL;
delete[] name;
name=NULL;
```

相关 API 函数：

**CameraGetCount**

**CameraGetName**

获取分辨率列表并设置相机分辨率

```
int index=0, width=0, height=0, camera_count=0;
CameraGetResolutionCount(m_index, &camera_count); //获取相机m_index的分辨率种数
//依次获取相机分辨率
for(int i=0;i<camera_count;i++)
{
    CameraGetResolution( m_index, i,&width, &height);
    ...//获取相机分辨率列表后，显示分辨率列表
}
...//可根据需要添加其他功能代码
CameraSetResolution(m_index, index, &width1, &height1); /*设置相机m_index的分辨
率为第index组分辨率*/
```

相关 API 函数：

**CameraGetResolutionCount**

**CameraGetResolution**

**CameraSetResolution**

开始采集和显示图像

启动

```
CameraPlay(m_device_id, hImage[m_device_id], SnapThreadCallback);
/*在句柄为hImage[m_device_id]的窗口显示相机m_device_id的画面， SnapThreadCallback为
回调函数， 可用0替代*/
```

停止



```
CameraStop(m_device_id); //停止编号为m_device_id的相机
```

#### 回调函数

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height,  
int format)  
{  
    //用户可添加代码处理获取的图像，图像信息存放在pImageBuffer中  
    return 0;  
}
```

#### 相关 API 函数：

[CameraPlay](#)

[CameraStop](#)

## 3.5 OpenCV

### 3.5.1 功能介绍

OpenCV 把相机的图像数据转换成 IplImage 格式。并演示调用 opencv 图像算子进行图像运算。

### 3.5.2 代码示例

```
#include "../../SDK/JHCap.h"  
  
#include <opencv2/core/core.hpp>  
#include <opencv2/highgui/highgui.hpp>  
#include <opencv2/opencv.hpp>  
#include <iostream>  
  
using namespace cv;  
using namespace std;  
  
#pragma comment(lib, "../../SDK/JHCap2.lib")  
  
int main( int argc, char**argv )  
{  
    int count;  
    CameraGetCount(&count);  
    if(count<=0) return 0;  
  
    int width = 800;  
    int height = 600;  
  
    CameraInit(0);  
    //CameraSetExposure(0, 400);
```



```
CameraGetResolution(0, 0, &width, &height);

IplImage * image = cvCreateImage(cvSize(width, height), 8, 3);
cvNamedWindow( "Camera Example" );
int seq = 0;
while(1)
{
    seq++;
    //color image
    int len = 0;
    CameraGetImageBufferSize(0, &len, CAMERA_IMAGE_BMP);
    CameraQueryImage(0, (unsigned char *)image->imageData, &len, CAMERA_IMAGE_BMP);

    ////do image processing using buffer or IplImage
    cvErode(image, image, 0, 2);
    cvDilate(image, image, 0, 2);

    cvShowImage("Camera Example", image);
    int key = cvWaitKey(25);           //ESC 退出
    if( key == 27 )
    {
        break;
    }
}
CameraFree(0);
cvReleaseImage( &image );
cvDestroyWindow( "Camera Example" );
return 0;
}
```

相关 API 函数：

**CameraQueryImage**  
**CameraGetImageBufferSize**

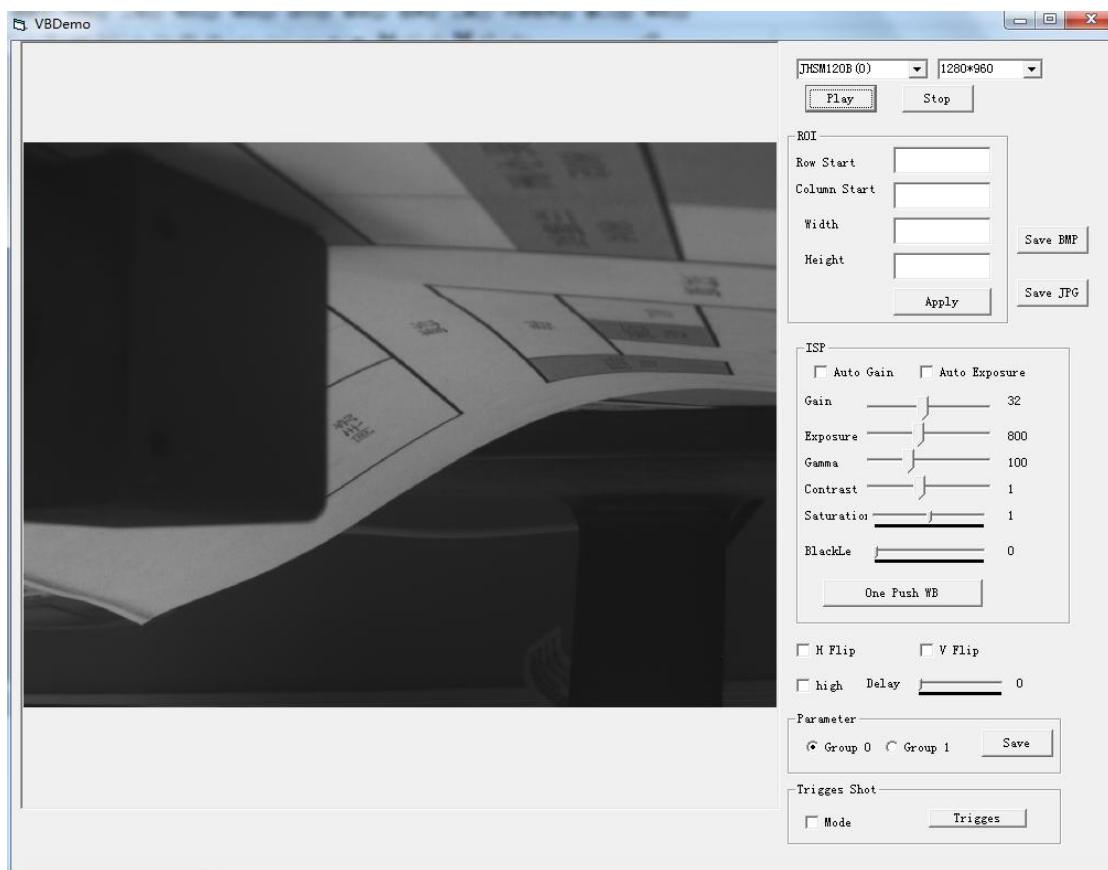


## 4 VB

### 4.1 OneCamera\_VBDemo

#### 4.1.1 功能介绍

OneCamera\_VBDemo 这个示例实现的功能主要包括：对相机采集的图像进行显示，ROI 设置，保存图片，实现一键白平衡，水平和垂直镜像操作，相机参数调节和保存，相机高速设置，软件触发拍摄图像等。



#### 功能实现说明

1 图像的显示：通过右上方的两个下拉按钮可以对相机和对应的分辨率进行选择，点击 Play 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

3 保存图片：可对界面显示的图像信息进行截取并保存，其中点击 Save BMP 可保存 BMP 格式的图片，点击 Save JPG 可保存 JPG 格式的图片。



4 参数调节和保存：支持两组参数的保存功能，点击 Group1 和 Group2 可进行两组参数之间的切换。通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动下方的滑动条可对相机的参数进行手动设置。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

5 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

6 水平和垂直镜像操作：选定 Hflip 就能实现水平镜像操作，选定 Vflip 就能实现垂直镜像操作。

7 高速设置：选定 high 复选框，就能对相机实现高速设置，滑动对应滚动条，可对相机实现降速操作。

8 通过软件触发拍摄图像：选定 Mode 能相机软件进入触发模式后，点击 Software trigger 按钮就能通过软件触发去拍摄图像。

#### 4.1.2 代码示例

##### 初始化获取相机列表

```
' 获得相机数目
Dim m_count As Long
m_count = 0
CameraGetCount m_count
' 相机名字
Dim name As String * 255
Dim model As String * 255
Dim i As Long
For i = 0 To m_count - 1
    CameraGetName i, name, model
' 显示相机名字列表.....
Next i
```

相关API函数：

**CameraGetCount**  
**CameraGetName**

##### 保存图像

```
' 保存jpg格式图像
CommonDialogSave.DialogTitle = "Save As"
CommonDialogSave.Filter = "jpg File(*.jpg) | *.jpg"
CommonDialogSave.FilterIndex = 1
CommonDialogSave.ShowSave
Dim temp1 As String
temp1 = CommonDialogSave.filename
CameraSaveJpegB 0, temp1, True
```

```
' 保存bmp格式图像
```

```
CommonDialogSave.DialogTitle = "Save As"
```



```
CommonDialogSave.Filter = "bmp File (*.bmp) | *.bmp"
CommonDialogSave.FilterIndex = 1
CommonDialogSave.ShowSave
Dim temp As String
temp = CommonDialogSave.filename
CameraSaveBMPB g_index, temp
```

相关 API 函数：

**CameraSaveJpegB**

**CameraSaveBMPB**

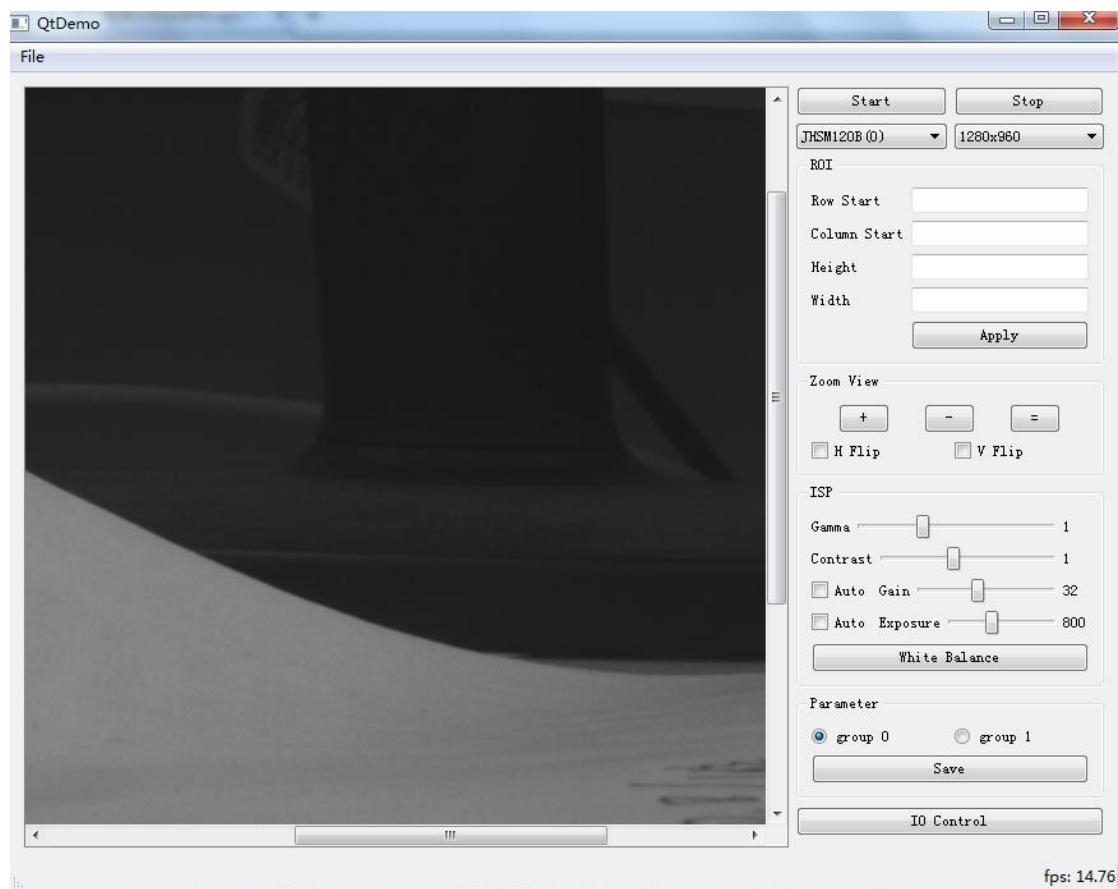


## 5 QT

### 5.1 OneCamera\_QtDemo

#### 5.1.1 功能介绍

OneCamera\_QtDemo.exe 这个示例实现的功能主要包括：对相机采集的图像进行显示，ROI 设置，水平和垂直镜像操作，图像缩小与放大，参数调节与保存，一键白平衡，相机 GPIO 设置。



#### 功能实现说明

1 图像的显示：通过右上方的两个下拉菜单可以对相机和对应的分辨率进行选择，点击 Start 按钮能够实现对相机图像的显示，点击 Stop 按钮停止显示。

2 ROI 设置：在 ROI 控制栏中，把想要实现的 ROI 数据分别输入到 Row Start、Column Start、Width、Height 中，然后点击 Apply 就能实现 ROI 设置了。

其中，Row Start 为 ROI 的起始水平位置；Column Start 为 ROI 的起始垂直位置；Width 为 ROI 的宽度；Height 为 ROI 的高度。Row Start 和 Column Start 的值需要大于或者等于 0，Width 和 Height 的值需要小于（不包括等于）当前分辨率下的宽度和高度值。

3 水平和垂直镜像操作和图像的放大与缩小：选定 Hflip 就能实现水平镜像操作，选定



VFlip 就能实现垂直镜像操作，点击“+”按钮实现放大操作，点击“-”按钮实现图像缩小操作，点击“=”还原原始图像大小。

4 参数保存功能和参数调节：支持两组参数的保存功能，点击 Group1 和 Group2 可进行两组参数之间的切换。通过选定 Auto Gain 和 Auto Exposure 能把相机设置成自动增益和自动曝光，滑动对应的滑动条可对相机的参数进行手动调节。

其中，Gain 对应为增益值，范围为 1~255；Exposure 对应为曝光值，范围为 1~5000；Gamma 对应为伽马值，范围为 0~2；Contrast 对应为对比度，范围为 0~2。

5 一键白平衡：点击 One Push WB 按钮就能对相机实现一键白平衡。实现白平衡时最好将相机对准一张白纸。如果一次没有实现可以多次点击一键白平衡按钮，直到颜色正常。

6 GPIO 设置：点击 IO Control 按钮可弹出 IO 控制面板，可分别对相关操作进行设置。

### 5.1.2 代码示例

#### 初始化获取相机列表

```
int count;
CameraGetCount(&count);
for(int i=0; i<count; i++)
{
    char name[255], model[255];
    CameraGetName(i, name, model);
    //显示相机列表.....
}
```

相关API函数：

    CameraGetCount  
    CameraGetName

#### 开始采集和显示图像

##### 图像采集

```
forever
{
    int width, height, len;

    if(work)
    {
        CameraGetImageSize(index,&width, &height);
        CameraGetImageBufferSize(index,&len, CAMERA_IMAGE_RGB24);
        unsigned char *buffer = new unsigned char[len];
        if(CameraQueryImage(index,buffer, &len,
CAMERA_IMAGE_RGB24)==API_OK)
        {
            if(term) break;
            QImage img(buffer, width, height, QImage::Format_RGB888);
            emit captured(img, buffer);
        } else usleep(1000);
    } else usleep(1000);
}
```



```
    if(term) break;  
}
```

## 绑定消息和槽

```
connect(m_thread, SIGNAL(captured(QImage, unsigned char *)),  
        this, SLOT(process(QImage, unsigned char *)));
```

## 槽函数

```
void MainWindow::process(QImage img, unsigned char *buffer)  
{  
    //显示 img 图像或者其他操作  
}
```

## 相关 API 函数:

```
CameraGetImageSize  
CameraGetImageBufferSize  
CameraQueryImage
```



## 6 Demo

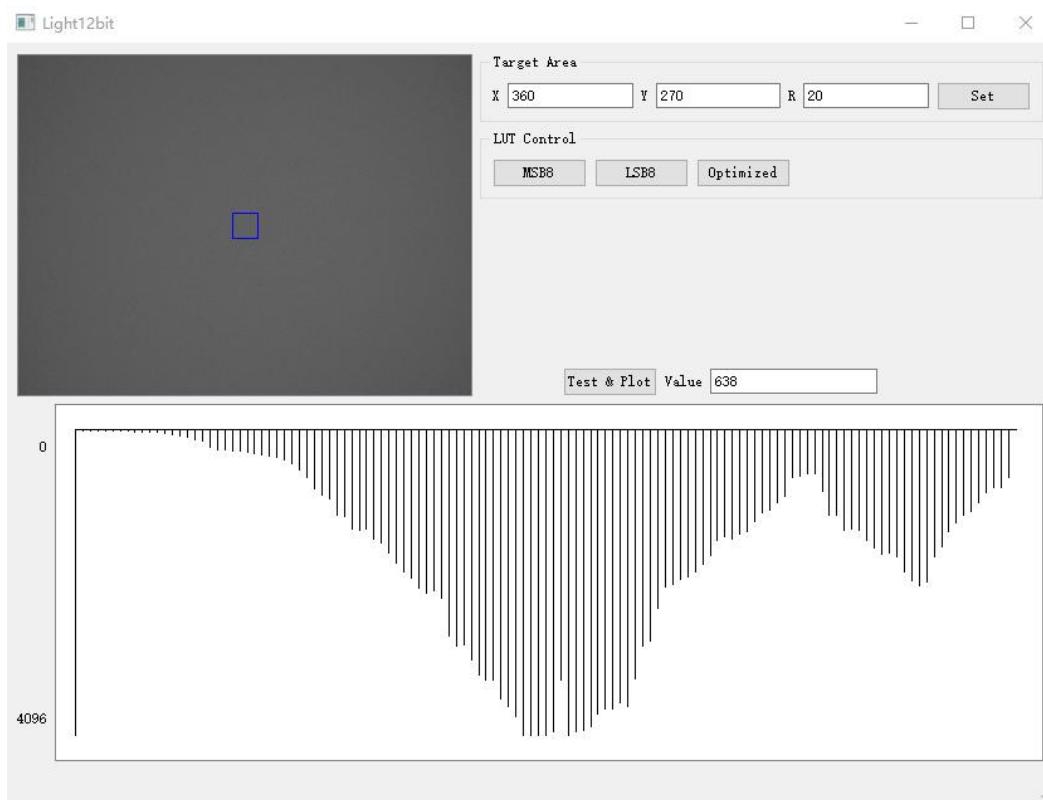
Demo 为演示 API 的同时实现具有一定应用价值的演示程序.

### 6.1 Light12bit

#### 6.1.1 功能介绍

本实例演示 12bit 相机的应用，测定光线亮度变化，并按时间（测量次数）显示为图形。

相对于 8bit 位深的相机，12bit 相机的测量结果范围更大（0-4095），对于需要精确检测图像亮度的应用有显著的优势。由于正常计算机显示的图像均为 8 位，12 位图像可应用于计算中，最终显示仍然需要转换为 8 位，需要通过查找表实现 12bit 转 8bit 的过程。



#### 6.1.2 功能实现说明

软件界面左上方显示图像，图像上的蓝框为设置的探测区域（Target Area），探测的值为本区域的平均值。显示图像可以通过 LUT Control 控制显示的 12 位数据的部分信息。

统计区域设置: Target Area, 定义一个以(X,Y)为中心, 2R 为边长的正方形区域用于计算灰度值。灰度值是基于从相机采集的 RAW 数据(12bit)。

显示设置: LUT Control, 使用预设的查找表, MSB8, 显示高 8 位, LSB8, 显示低 8 位,



Optimized，优化设置，提升暗区的区分度。

操作：Test & Plot，采集一张图片，计算灰度值显示为 Value，并在下方的区域绘制一条直线，直线的高度对应灰度的大小。最多可绘制 126 条。

### 6.1.3 代码示例

```
//初始化, 获取位深度, 位深为 8 时, 测量范围 (0-255), 位 12 时 (0-4095)
CameraInit(0);
CameraGetImageDepth(0, &m_depth);

// 采集 RAW12 数据, 一个像素占据 2 个字节
// CAMERA_IMAGE_RAW 选项采集相机输出的 RAW8 或者 RAW12
CameraGetImageSize(index, &width, &height);
CameraGetImageBufferSize(index, &len, CAMERA_IMAGE_RAW);

CameraGetISPImageBufferSize(index, &size, width, height, CAMERA_IMAGE_GRAY8);
unsigned char *buffer = new unsigned char[len];
unsigned char *ISPbuf = new unsigned char[size];
if(CameraQueryImage(index, buffer, &len, CAMERA_IMAGE_RAW)==API_OK)
{
    //应用 LUT 转换为 GRAY8
    if(CameraISP(index, buffer, ISPbuf, width, height,
        CAMERA_IMAGE_GRAY8)==API_OK)
    {
        //buffer 为 RAW12 数据, ISPbuf 为 GRAY8 数据
    }
}

//将 RAW12 图像缓存从 unsigned char 转换为 unsigned short
unsigned short *ptr = (unsigned short *)buffer;
//统计
long sum = 0;
for(int j=m_y-m_r; j<m_y+m_r; j++)
{
    int jwidth = j*m_image_item->pixmap().width();
    for(int i=m_x-m_r; i<m_x+m_r; i++)
    {
        if(m_depth==12) sum += ptr[jwidth+i];
        else sum += buffer[jwidth+i];
    }
}
//得到平均值 avg
double avg = sum/(4*m_r*m_r);
```



相关API函数：

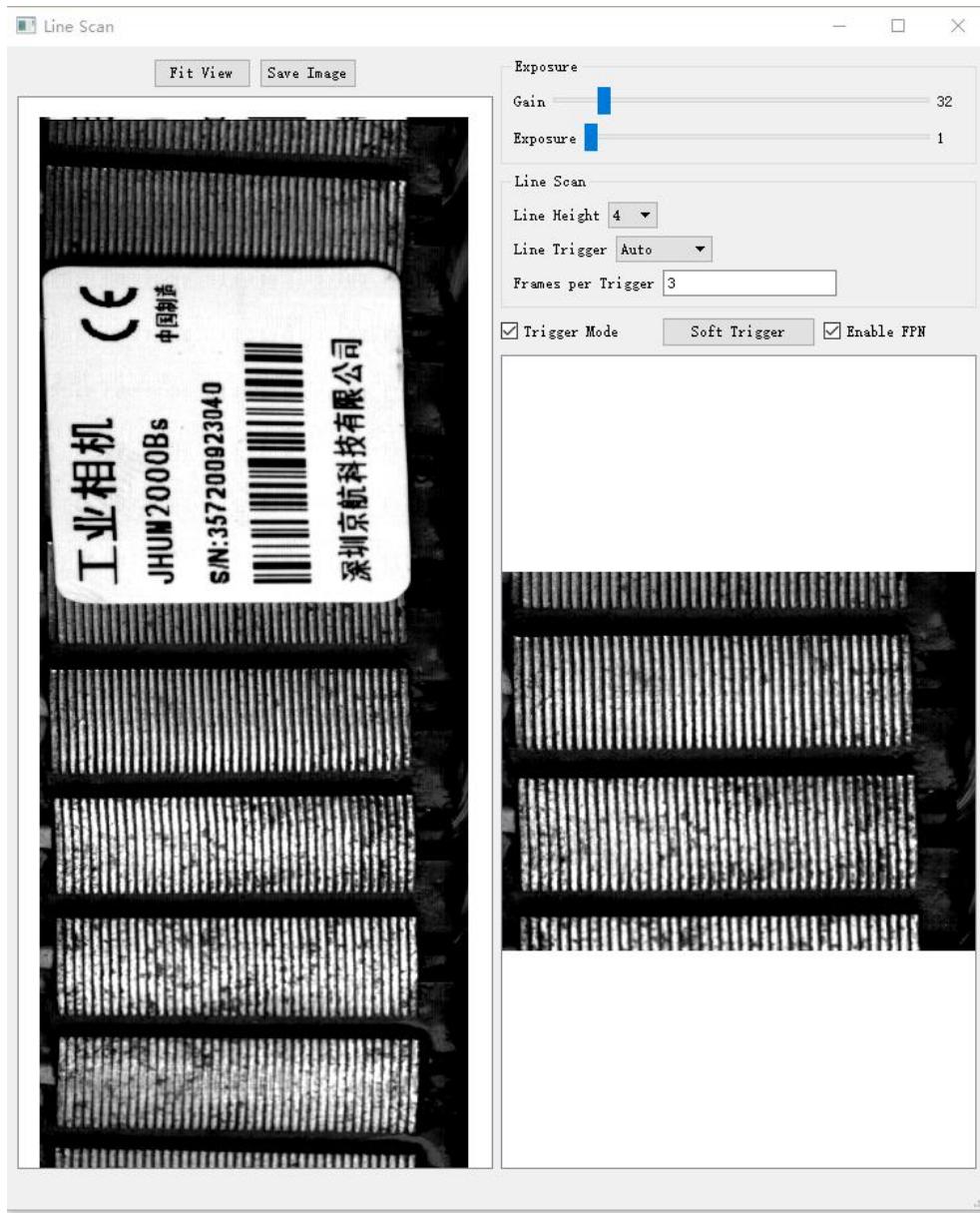
CameraGetImageDepth  
CameraQueryImage  
CameraISP

## 6.2 LineScan

### 6.2.1 功能介绍

本实例演示线扫扫描相机的应用，相机拍摄一个旋转的柱面，开启相机的线扫功能，将多张线扫图形拼接为完整的一张图形，将圆柱表面展开成一个平面图形。

柱面线扫时相机的安装，图像的水平方向，即感光芯片的长边，需要和旋转轴平行，旋转方向由相机背面转至正面（有标签的一面）。





## 6.2.2 功能实现说明

曝光和增益控制：Exposure, Gain 可以调节画面亮度，线扫应用中，需要高亮度的光源以减少 Exposure 时间来避免拖影。

线扫参数：Line Height，对于面阵相机的线扫功能，可以借助面阵相机的优势，调节线扫描感光的行数。行数越高线扫描特性越低，图形帧率越高。

Line Trigger，无同步触发时选 Auto，自动提供行触发信号，外部触发或者软触发作为帧触发使用。如果 External 方式下，需要接外部触发源，本示例只能预览图像。

Frames per Trigger，一次软件触发触发图像的张数，如果一个柱面一张照片无法扫描完全，可以增加触发的张数，所有触发的图像，将在左方的列表一次排列显示。

视图和文件操作：FitView 可以根据窗口大小排列图像，Save Image 可以保存拼合的图像为一个图像文件。

触发模式：Trigger Mode，勾选以后进入触发模式，此时画面静止，点击 Soft Trigger 触发指定数量的图片。Enable FPN 可以开启固定噪声修复。

## 6.2.3 代码示例

```
//初始化
CameraInit(0);

//设置线扫参数
CameraSetResolutionMode(0, CAMERA_RESOLUTION_LINESCAN);
CameraSetSnapMode(0, CAMERA_SNAP_CONTINUATION);
CameraSetLineHeight(0, 2);
CameraSetLineTrigger(0, CAMERA_LINE_TRIGGER_AUTO);
CameraSetSnapNum(0, 3);
CameraSetFPN(0, true);

//设置曝光和增益
CameraSetGain(0, 32);
CameraSetExposure(0, 20);

//触发拍照
CameraTriggerShot(0);
```

相关API函数：

CameraSetResolutionMode  
CameraSetLineHeight  
CameraSetLineTrigger  
CameraSetSnapNum  
CameraSetFPN