



## SDK Sample

The JHCap2 SDK program examples include code for languages such as C#, VB.net, VC, VB, QT, etc., which demonstrating how to use a single/multiple camera. Most examples include a callback version and a thread version. It also including example shows how to integrate with OpenCV. The samples can be used as a reference for using the SDK or as a basic framework for specific visual application development.



## Contents

SDK Sample .....	I
Contents .....	II
1 CSharp .....	1
1.1 OneCamera_CSharpDemo .....	1
1.1.1 Function .....	1
1.1.2 Sample .....	2
1.2 OneCamera_CSharpDemo_Thread .....	3
1.2.1 Function .....	3
1.2.2 Sample .....	5
1.3 TwoCamera_CSharpDemo .....	6
1.3.1 Function .....	6
1.3.2 Sample .....	8
2 VB.net .....	10
2.1 OneCamera_VBNetDemo .....	10
2.1.1 Function .....	10
2.1.2 Sample .....	11
2.2 OneCamera_VBNetDemo_Thread .....	13
2.2.1 Function .....	13
2.2.2 Sample .....	14
2.3 TwoCamera_VBNetDemo .....	16
2.3.1 Function .....	16
2.3.2 Sample .....	17
3 VC .....	19
3.1 OneCamera_VCDemo .....	19
3.1.1 Function .....	19
3.1.2 Sample .....	20
3.2 OneCamera_VCDemo_Thread .....	22
3.2.1 Function .....	22
3.2.2 Sample .....	23
3.3 TwoCamera_VCDemo .....	26
3.3.1 Function .....	26
3.3.2 Sample .....	28
3.4 FourCamera_VCDemo .....	30
3.4.1 Function .....	30
3.4.2 Sample .....	31
3.5 OpenCV .....	33
3.5.1 Function .....	33
3.5.2 Sample .....	33
4 VB .....	35
4.1 OneCamera_VBDemo .....	35



4.1.1 Function .....	35
4.1.2 Sample .....	36
5 QT .....	38
5.1 OneCamera_QtDemo .....	38
5.1.1 Function .....	38
5.1.2 Sample .....	39
6 Demo .....	41
6.1 Light12bit .....	41
6.1.1 Function .....	41
6.1.2 Implementation .....	41
6.1.3 Sample Code .....	42
6.2 LineScan .....	43
6.2.1 Function .....	43
6.2.2 Implementation .....	44
6.2.3 Sample Code .....	44

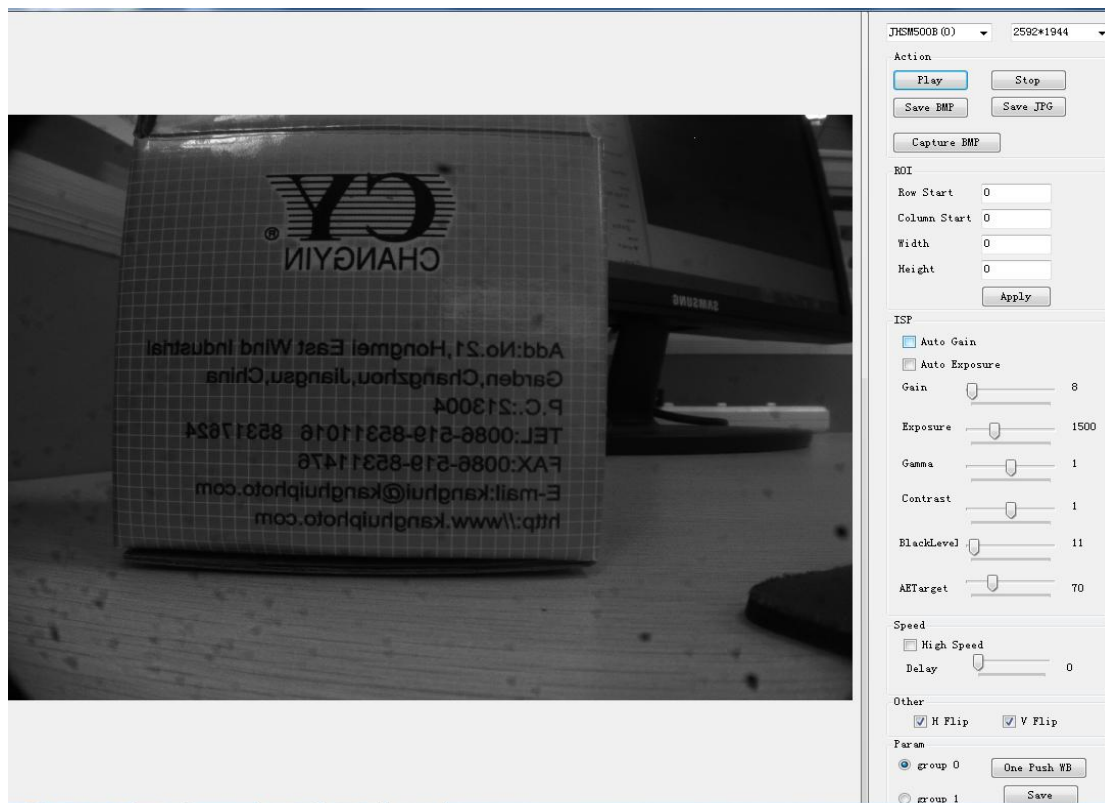


# 1 CSharp

## 1.1 OneCamera\_CSharpDemo

### 1.1.1 Function

The examples OneCamera\_CSharpDemo shows: displaying images captured by a single camera, saving images, ROI settings, camera parameter settings (automatic gain, auto exposure, gain, exposure, gamma, contrast, camera black level, camera auto exposure target value), set transfer speed, set delay time, set horizontal and vertical mirror, white balance, saving parameter, etc.



### Description

1 Image display: the camera and corresponding resolution can be selected by the two drop-down combo box at the upper right side. Click the *Play* button to stream and display the camera image, and click the *Stop* button to stop the stream.

2 Save the image: save the image displayed on the interface. Click *SaveBMP* to save the current displayed as a BMP file. Click *SaveJPG* to save the current display as a JPG file. Click *Capture BMP* to save it as BMP file.

3 ROI setting: in the ROI group box, input the ROI data you want to set into Row Start, Column



Start, Width, Height, and then click Apply.

4 Camera parameter setting: by setting Auto Gain and Auto Exposure, the camera can be enable AEC and AGC. Slide the slider below to manually set the gain and exposure.

Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~32767; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2; Blacklevel corresponds to the camera black level value, the range is 0~255; AE Target corresponds to the automatic exposure target value, the range is 0~255. At the same time, the auto exposure target value needs to be adjusted only when the auto exposure function is used. When the target value is set, the stable value of the auto exposure will keep approaching the target value.

5 Data transfer mode: The data transfer mode can be set to high speed by selecting the High Speed control, otherwise lower speed, high speed can transfer about 24M pixel/s, compare to about 40M pixels/s at high speed.

6 Set the delay time: insert delay between frames and lines, higher delay, result in lower fps.

7 Horizontal and vertical mirroring: Horizontal mirroring can be performed by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

8 One push white balance: Click the One Push WB button to achieve a one push white balance on the camera. When doing one push white balance, you'd better to shooting on a white paper. If it is not achieved good result at one time, you can click the one push white balance button multiple times until the color is good.

9 Parameter saving function: Supports saving two sets of parameters. Select Group1 or Group2 to switch between two sets of parameters. Click Save to save the corresponding parameter set.

## 1.1.2 Sample

Initialize the camera list

```
int m = 0;
JHCap.CameraGetCount(ref m);    //Get the number of connected cameras
StringBuilder name = new StringBuilder();
StringBuilder model = new StringBuilder();
//Get the camera name
for (int i = 0; i < m; i++)
{
    JHCap.CameraGetName(i, name, model);
    ...//Display camera list after getting camera list
}
... //Get the number of connected cameras
```

API:

CameraGetCount  
CameraGetName

Get a list of resolutions and set the resolution for the camera



```
int reso_width = 0, reso_height = 0, reso_count = 0,
JHCap.CameraGetResolutionCount(m_index, ref reso_count);/*Get the camera m_index
resolution number*/
//Get the camera m_index resolution in turn
for (int j = 0; j < reso_count; j++)
{
    JHCap.CameraGetResolution(m_index, j, ref reso_width, ref reso_height);
    ...//List of display resolutions after getting a list of camera resolutions
}
...//Additional feature codes can be added as needed
JHCap.CameraSetResolution(m_index, index, &width1, &height1);/*Set the resolution of
the camera m_index to the index of the index group*/
```

API:

CameraGetResolutionCount  
CameraGetResolution  
CameraSetResolution

Display Image

Play

```
JHCap.CameraPlay(g_index, pictureBox1.Handle, callback);
//The screen of the camera g_index1 is displayed in the window with the handle of
pictureBox1.Handle, and the callback is a callback function.
```

Callback

```
//Callback function definition
private static int Callback(IntPtr pImageBuffer, int width, int height, int format)
{
    //The user can add code to process the acquired image, and the image information
    is stored in pImageBuffer.
    return 0;
}
```

API:

CameraPlay

## 1.2 OneCamera\_CSharpDemo\_Thread

### 1.2.1 Function

The functions implemented by OneCamera\_CSharpDemo\_Thread mainly include: displaying the images captured by a single camera in a threaded manner, setting camera parameters



(including: automatic gain, auto exposure, gain, exposure, gamma, contrast, camera black level, ,auto exposure and auto target value), horizontal and vertical mirroring operations, one-button white balance, camera GPIO settings, software triggered shooting images, etc.



## Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera's parameters.

Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2; Blacklevel corresponds to the camera black level value, the range is 0~255; AE Target corresponds to the automatic exposure target value, the range is 0~255. At the same time, the auto exposure target value needs to be adjusted only when the auto exposure function is used. When the target is set straight, the stable value of the auto exposure will keep approaching the target value.

3 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

4 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.



5 GPIO settings: Select the Output or Input control of the IO&Trigger module, and then click the Refresh Input button to set the camera's GPIO, where Output corresponds to Out and Input corresponds to In.

6 Capture images by software trigger: Select Enter Trigger Mode to put the camera into trigger mode, click the Software Trigger button to trigger the image through software trigger.

## 1.2.2 Sample

### Initialize the camera list

```
int m = 0;
JHCap.CameraGetCount(ref m);    //Get the number of connected cameras
StringBuilder name = new StringBuilder();
StringBuilder model = new StringBuilder();
//Get the camera resolution name in turn
for (int i = 0; i < m; i++)
{
    JHCap.CameraGetName(i, name, model);
    ...//Display camera list after getting camera list
}
... //Additional feature codes can be added as needed
```

API:

CameraGetCount

CameraGetName

### Start capturing and displaying images

Open a background thread and call CameraQueryBitmap to get image information. The thread can update the interface elements by means of a delegate.

```
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();
    }
}
```

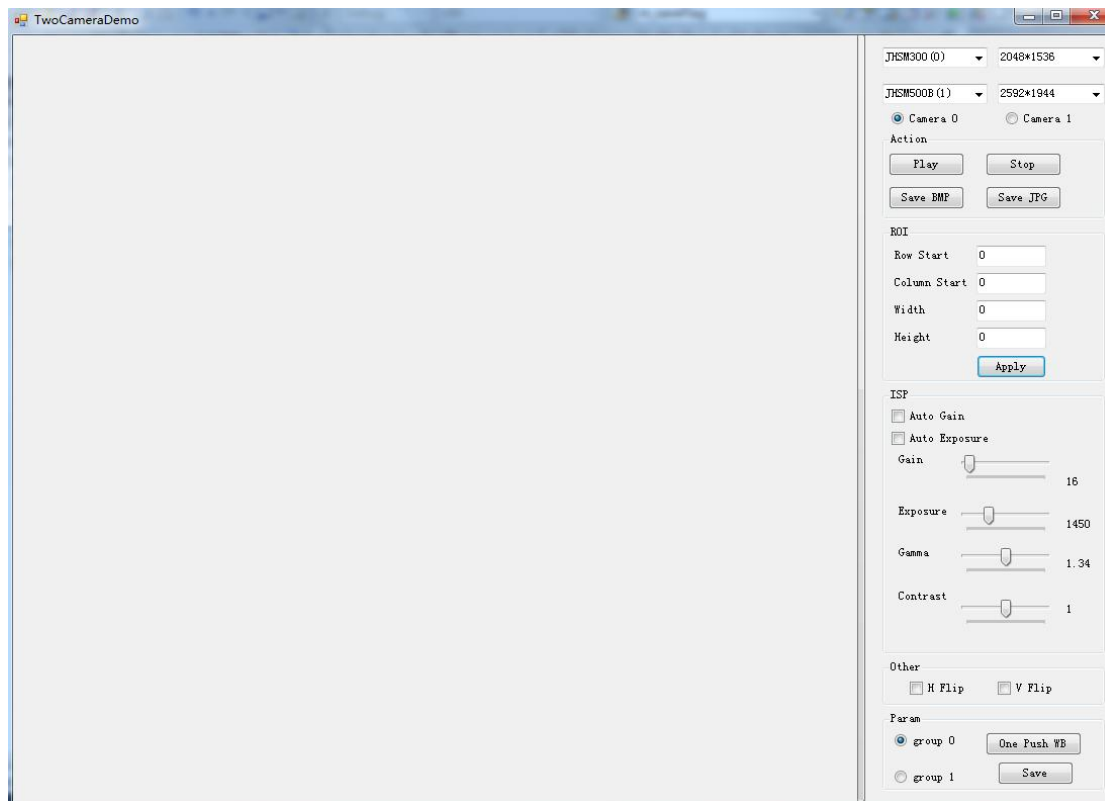
API:

CameraQueryBitmap

## 1.3 TwoCamera\_CSharpDemo

### 1.3.1 Function

The functions of the TwoCamera\_CSharpDemo example mainly include: displaying images captured by two cameras, saving images, ROI settings, camera parameter settings (including: automatic gain, auto exposure, gain, exposure, gamma, contrast), horizontal and Vertical mirroring operation, one-button white balance, parameter saving, etc.



## Description

1 Image display: The two sets of cameras and corresponding resolution can be selected by the four drop-down buttons at the upper right. Click the Play button to display the camera image. Click the Stop button to stop the display. Select Camera0 or Camera1. Select the camera you want to operate.

2 Save the image: After selecting the camera, click the Save BMP button to save the BMP format image, and click the Save JPG button to save the JPG format image.

3 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.

Where Row Start is the starting horizontal position of the ROI; Column Start is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

4 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 Horizontal and vertical mirroring operation: The HFlip is selected for horizontal mirroring of the camera, and the VFlip is selected for vertical mirroring.

6 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of



white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

7 Parameter saving function: Supports the saving function of two sets of parameters. Selecting Group1 or Group2 can switch between two sets of parameters. Click Save to save the corresponding parameter set.

## 1.3.2 Sample

### Get camera list and initialize camera

```
int m = 0;    //Number of cameras
JHCap.CameraGetCount(ref m);
StringBuilder name = new StringBuilder();
StringBuilder model = new StringBuilder();
//Get the camera resolution name in turn, and initializing the camera
for (int i = 0; i < m; i++)
{
    JHCap.CameraGetName(i, name, model);
    JHCap.CameraInit(i);
    JHCap.CameraSetHighspeed(i, false);
    ...//Display camera list after getting camera list
}
... //Additional feature codes can be added as needed
```

API:

CameraGetCount  
CameraGetName  
CameraInit  
CameraSetHighspeed

### Get a list of resolutions and set the resolution for the camera

```
int reso_width = 0, reso_height = 0, reso_count = 0,
JHCap.CameraGetResolutionCount(m_index, ref reso_count);/*Get the camera m_index
resolution number*/
//Get the camera m_index resolution in turn
for (int j = 0; j < reso_count; j++)
{
    JHCap.CameraGetResolution(m_index, j, ref reso_width, ref reso_height);
    ...//List of display resolutions after getting a list of camera resolutions
}
...//Additional feature codes can be added as needed
JHCap.CameraSetResolution(m_index, index, &width1, &height1);/*Set the resolution of
the camera m_index to the index of the index group*/
```



API:

CameraGetResolutionCount  
CameraGetResolution  
CameraSetResolution

Display Image

Play

```
JHCap.CameraPlay(g_index1, pictureBox1.Handle, callback);  
/*The screen of the camera g_index1 is displayed in the window with the handle of  
pictureBox1.Handle, and the callback is a callback function.*/  
JHCap.CameraPlay(g_index2, pictureBox2.Handle, callback);  
/*The screen of the camera g_index2 is displayed in the window with the handle of  
pictureBox2.Handle, and the callback is a callback function.*/
```

Callback

```
//Callback function definition  
private static int Callback(IntPtr pImageBuffer, int width, int height, int format)  
{  
    /*You can add code to process the acquired image, and the image information is stored  
in pImageBuffer.*/  
    return 0;  
}
```

API:

CameraPlay

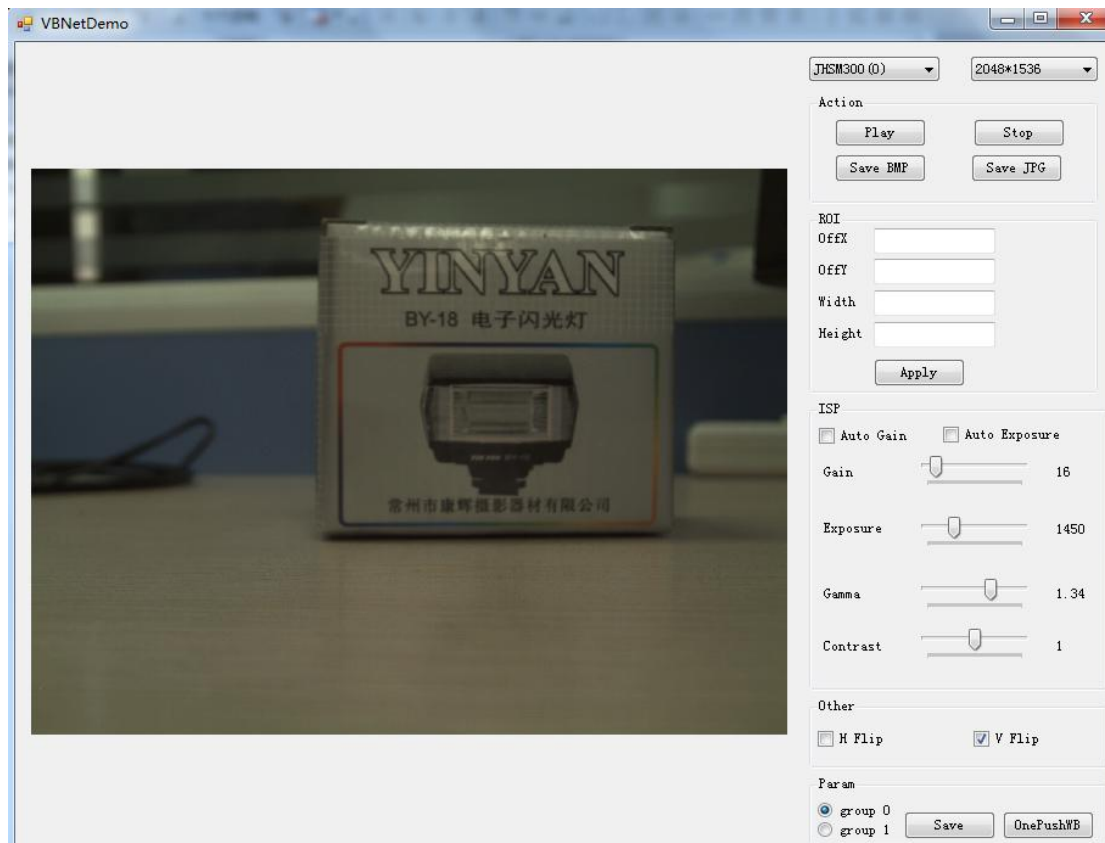


## 2 VB.net

### 2.1 OneCamera\_VBNetDemo

#### 2.1.1 Function

The functions implemented by OneCamera\_VBNetDemo mainly include: displaying images captured by a single camera, saving pictures, ROI settings, camera parameter settings (including: automatic gain, auto exposure, gain, exposure, gamma, contrast), horizontal and vertical Mirror operation, parameter saving, one-button white balance, etc.



#### Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 Save the picture: You can intercept and save the image information displayed on the interface. Click Save BMP to save the currently displayed image as a BMP format image. Click Save JPG to save the currently displayed image as a JPG format image.



3 ROI setting: In the ROI control bar, input the ROI data you want to implement into offX, offY, Width, and Height, and then click Apply to implement the ROI setting.

Where offX is the starting horizontal position of the ROI; offY is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

4 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

6 Parameter saving function: Supports the saving function of two sets of parameters. Select Group1 or Group2 to switch between two sets of parameters. Click Save to save the corresponding parameter set.

7 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

## 2.1.2 Sample

### Get camera list and initialize camera

```
Dim cameraCount As Integer      //Total number of connected cameras
//Get the total number of cameras
Try
    JHCap.CameraGetCount(cameraCount)
Catch ex As Exception
    MessageBox.Show("Form load exception")
End Try
//Initialize all cameras
If cameraCount > 0 Then        //The total number of cameras is greater than 0
    JHCap.CameraInit(0)        //Initialize the 0th camera
    cameraCount = cameraCount - 1
End If
//Get the camera name in turn
Dim name As New StringBuilder
Dim mode As New StringBuilder
For i = 0 To cameraCount - 1
    JHCap.CameraGetName(i, name, mode)    //Get the camera name
    ...//Display camera list after getting camera list or insert your codes
```

[Next](#)

API:

CameraGetCount  
CameraInit  
CameraGetName

Get camera resolution

```
Dim resoCount As Integer           //Total resolution
Dim reso_width As Integer          //Resolution width
Dim reso_height As Integer         //Resolution height
//Get camera resolution in turn
JHCap.CameraGetResolutionCount(index, resoCount)  /*Get the total resolution of the
camera index*/
For j = 0 To resoCount - 1
    JHCap.CameraGetResolution(index, j, reso_width, reso_height)
    .../*After getting the camera resolution, display the camera resolution or add other
function codes*/
Next
```

API:

CameraGetResolutionCount  
CameraGetResolution

Image acquisition and display

Play

```
Try
    JHCap.CameraPlay(camera_index, PictureBox1.Handle, CallFunction) /*Display the
camera camera_index screen in the window with the handle PictureBox1.Handle,
SnapThreadCallback is the callback function*/
Catch ex As Exception
    MessageBox.Show("CallBack exception")
End Try
```

Callback

```
Private Function CallFunction(ByVal buf As IntPtr, ByVal width As Integer, ByVal height
As Integer, ByVal method As Integer) As Integer           //Callback function definition
    /*The user can add code to process the acquired image, and the image information
is stored in pImageBuffer.*/
    Return 0
End Function
```



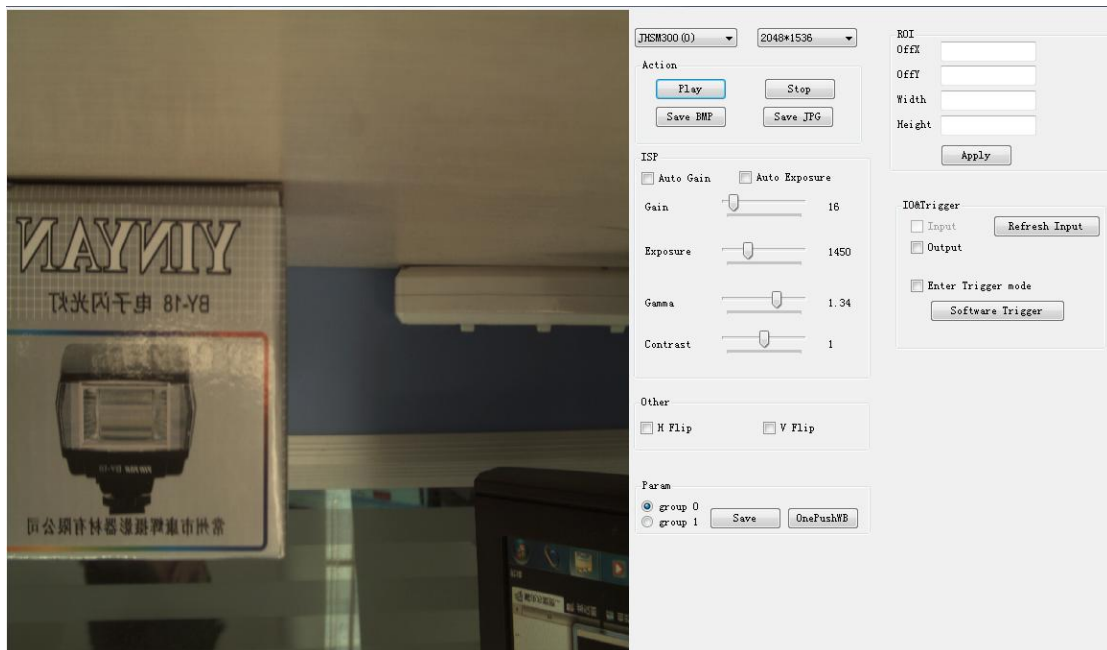
API:

CameraPlay

## 2.2 OneCamera\_VBNetDemo\_Thread

### 2.2.1 Function

The functions implemented by OneCamera\_VBNetDemo\_Thread mainly include: displaying the image reference thread captured by a single camera, saving the image, ROI settings, camera parameter settings (including: automatic gain, auto exposure, gain, exposure, gamma, contrast), horizontal and vertical mirroring operations, parameter saving, one-button white balance, GPIO settings and software triggered shooting images.



### Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 Save the picture: You can intercept and save the image information displayed on the interface. Click Save BMP to save the currently displayed image as a BMP format image. Click Save JPG to save the currently displayed image as a JPG format image.

3 ROI setting: In the ROI control bar, input the ROI data you want to implement into offX, offY, Width, and Height, and then click Apply to implement the ROI setting.

Where offX is the starting horizontal position of the ROI; offY is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width





and Height need to be less than (not including equal to) the width and height values at the current resolution.

4 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

6 Parameter saving function: Supports the saving function of two sets of parameters. Select Group1 or Group2 to switch between two sets of parameters. Click Save to save the corresponding parameter set.

7 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

8 GPIO settings: Select the Output control of the IO&Trigger module to set the camera's GPIO to OUT.

9 Capture images by software trigger: Select Enter Trigger Mode to put the camera into trigger mode, click the Software trigger button to capture images by software trigger.

## 2.2.2 Sample

### Get camera list and initialize camera

```
Dim cameraCount As Integer          //Total number of connected cameras
Try
    JHCap.CameraGetCount(cameraCount)    //Get the total number of cameras
Catch ex As Exception
    MessageBox.Show("Form load exception")
End Try
//Initialize all cameras
If cameraCount > 0 Then                //The total number of cameras is greater than 0
    JHCap.CameraInit(0)                //Initialize the 0th camera
    cameraCount = cameraCount - 1
End If
//Get the camera name in turn
Dim name As New StringBuilder
Dim mode As New StringBuilder
For i = 0 To cameraCount - 1
    JHCap.CameraGetName(i, name, mode)    //Get the camera name
    ...//Display camera list after getting camera list or insert your codes
Next
```



API:

CameraGetCount  
CameraInit  
CameraGetName

Get camera resolution

```
Dim resoCount As Integer           //Total resolution
Dim reso_width As Integer          //Resolution width
Dim reso_height As Integer         //Resolution height
JHCap.CameraGetResolutionCount(index, resoCount) //Get the total resolution of the
camera index
//Get camera resolution in turn
For j = 0 To resoCount - 1
    JHCap.CameraGetResolution(index, j, reso_width, reso_height)
    ...//After getting the camera resolution, display the camera resolution or add other
function codes
Next
```

API:

CameraGetResolutionCount  
CameraGetResolution

Image acquisition and display

Open a background thread and call CameraQueryBitmap to get the image. The thread can update the interface elements by means of a delegate.

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

Private Sub Process()

    While True           'Infinite loop •
        If m_work Then   'Work sign
            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

```

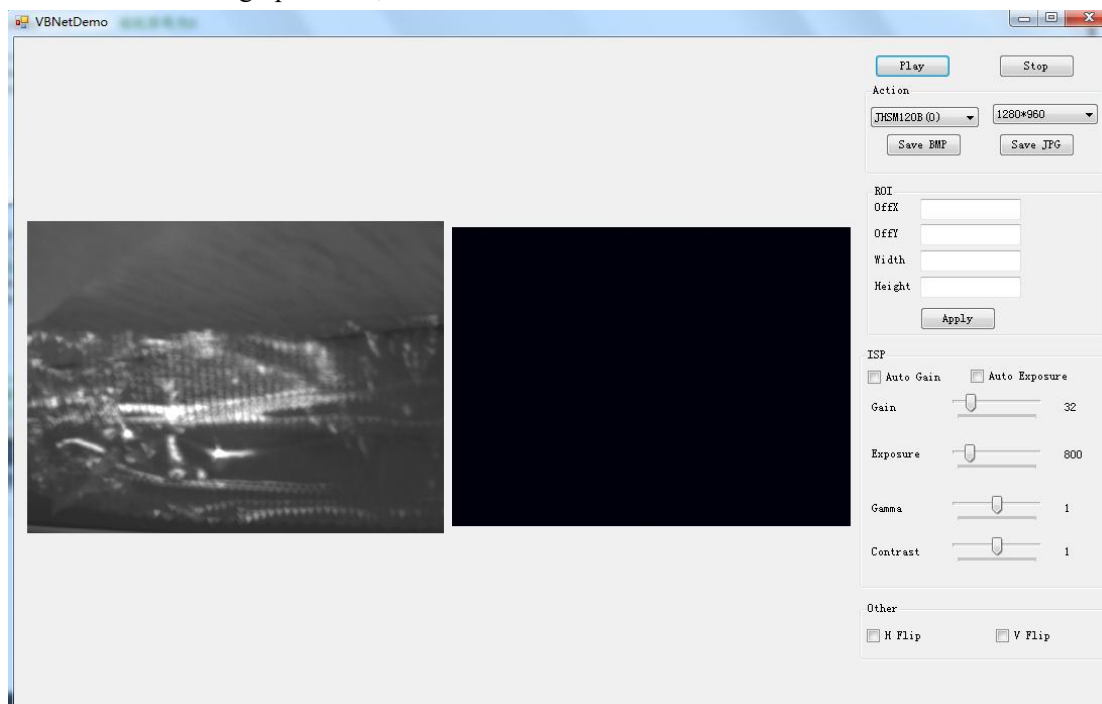
API:

CameraQueryBitmap

## 2.3 TwoCamera\_VBNetDemo

### 2.3.1 Function

The twoCamera\_VBNetDemo example implementation features: display the image captured by the dual camera using the callback function, save the image, ROI settings, camera parameter settings (including: automatic gain, auto exposure, gain, exposure, gamma, Contrast), horizontal and vertical mirroring operations, etc.



### Description

1 Image display: After launching the program and clicking the Play button, the image display function of the dual camera can be realized. Click the Stop button to stop the display. The two drop-down boxes are used to select a camera and adjust the resolution.

2 Save the picture: You can intercept and save the image information displayed on the interface. Click Save BMP to save the currently displayed image as a BMP format image. Click Save JPG to



save the currently displayed image as a JPG format image.

3 ROI setting: In the ROI control bar, input the ROI data you want to implement into offX, offY, Width, and Height, and then click Apply to implement the ROI setting.

Where offX is the starting horizontal position of the ROI; offY is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

4 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters. Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

## 2.3.2 Sample

### Save BMP format image

```
SaveFileDialog1.InitialDirectory = "C:\" 'Get save path
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) 'Save BMP image
End If
```

API:

#### CameraSaveBMPB

### Save JPG image

```
SaveFileDialog1.InitialDirectory = "C:\" 'Get save path
SaveFileDialog1.Filter = "jpg files (*.jpg)|*.jpg"
SaveFileDialog1.FilterIndex = 1
SaveFileDialog1.RestoreDirectory = True
SaveFileDialog1.Title = "保存?存?"
SaveFileDialog1.DefaultExt = ".jpg"
If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
```



```
Dim filename As String
filename = SaveFileDialog1.FileName
JHCap.CameraSaveJpegB(camera_index, filename, True) 'Save JPG image
End If
```

API:

CameraSaveJpegB

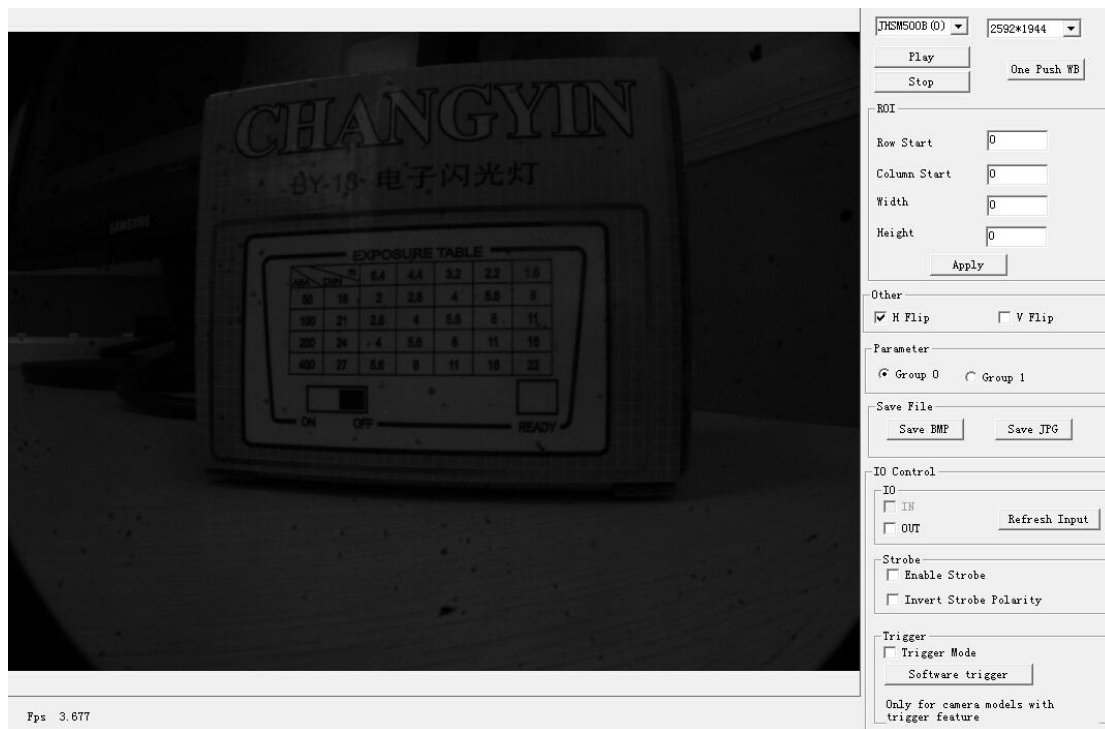


## 3 VC

### 3.1 OneCamera\_VCDemo

#### 3.1.1 Function

The functions implemented by OneCamera\_VCDemo mainly include: directly displaying the images captured by the camera, achieving one-button white balance, saving pictures, ROI settings, horizontal and vertical mirroring operations, parameter saving, camera GPIO settings, flash settings, software triggered shooting images. Wait.



#### Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

3 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.



Where Row Start is the starting horizontal position of the ROI; Column Start is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

4 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

5 Parameter saving function: Support the saving function of two sets of parameters. Click Group1 and Group2 to switch between two sets of parameters.

6 Save the picture: You can intercept and save the image information displayed on the interface. Click Save BMP to save the picture in BMP format. Click Save JPG to save the picture in JPG format.

7 GPIO settings: Select the OUT control of the IO module to set the camera's GPIO.

8 Flash settings: Select Enable Strobe to use the flash, and select Invert Strobe Polarity to set the output polarity of the flash.

9 Capture images by software trigger: After selecting Trigger Mode to put the camera into trigger mode, click the Software trigger button to trigger the image by software trigger.

### 3.1.2 Sample

#### Initialize the camera list

```
int CamAllNum=0;//Number of cameras
CameraGetCount(&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//Get the camera name in turn
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ... //Display camera list after getting camera list
    }
    ...//Additional feature codes can be added as needed
}
else
    return false;
delete[] model;
model=NULL;
delete[] name;
name=NULL;
```



API:

CameraGetCount  
CameraGetName

Get a list of resolutions and set the camera resolution

```
int index=0,width=0,height=0,camera_count=0;
CameraGetResolutionCount(m_index,&camera_count); /*Get the number of resolutions of
the camera m_index*/
//Get camera resolution in turn
for(int i=0;i<camera_count;i++)
{
    CameraGetResolution( m_index,i,&width, &height);
    ...//Get camera resolution list and display resolution list
}
...//Additional feature codes can be added as needed
CameraSetResolution(m_index,index,&width1, &height1);/*Set the resolution of the
camera m_index to the index of the index group*/
```

API:

CameraGetResolutionCount  
CameraGetResolution  
CameraSetResolution

Start collecting and displaying images

Play

```
CameraPlay(m_device_id,hImage[m_device_id],SnapThreadCallback);
/*The screen of the camera m_device_id is displayed in the window with the handle
hImage[m_device_id], and the SnapThreadCallback is a callback function, which can be
replaced by 0.*/
```

Stop

```
CameraStop(m_device_id); //Stop the camera numbered m_device_id
```

Callback

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height,
int format)
{
    //The user can add code to process the acquired image, and the image information is stored
in pImageBuffer.
    return 0;
}
```





API:

CameraPlay

CameraStop

## 3.2 OneCamera\_VCDemo\_Thread

### 3.2.1 Function

The functions implemented by OneCamera\_VCDemo\_Thread mainly include: displaying the image information of a single camera and referring to the thread, realizing one-button white balance, saving pictures, ROI settings, camera parameter settings (including: automatic gain, automatic exposure, gain, exposure), gamma value, contrast, saturation, camera black level and auto exposure target value), data transfer mode settings, delay time settings, horizontal and vertical mirroring operations, parameter saving, etc. Also introduced how to automatically reconnect.



### Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

3 Save the picture: You can capture and save the image information displayed on the interface.



Select BMP or JPG to select the format of the saved picture, and then click the Save button to save the current image. (BMP format corresponding to BMP, JPG format of JPG corresponding image)

4 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.

Where Row Start is the starting horizontal position of the ROI; Column Start is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

5 Camera parameter settings: By selecting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2; Saturation corresponds to saturation, the range is 0~2; Blacklevel corresponds to the camera black level value, the range is 0~255; AE Target corresponds to the automatic exposure target value, the range is 0~255. At the same time, the camera auto exposure target value needs to be adjusted only when the auto exposure function is used. When the target value is set, the stable value of the auto exposure will keep approaching the target value.

6 Data transfer mode: The data transfer mode can be set to high speed by selecting the High Speed control, and vice versa.

7 Set the delay time: You can set the delay time by sliding the slider corresponding to the Delay, ranging from 0 to 1000 milliseconds.

8 Horizontal and vertical mirroring operation: Horizontal mirroring can be performed by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

9 Parameter saving function: Supports the saving function of two sets of parameters. Select Group1 or Group2 to switch between two sets of parameters. Click Save to save the corresponding parameter set.

10 When the camera is plugged in or disconnected, or the connection is reconnected, the application will receive the Windows message WM\_DEVICE\_CHANGE, which will automatically re-initialize the camera and then work.

### 3.2.2 Sample

#### Initialize the camera list

```
int CamAllNum=0; //Number of cameras
CameraGetCount (&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//Get the camera name in turn
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
```



```
{
    CameraGetName(i, name, model);
    ...//Display camera list after getting camera list
}
...//Additional feature codes can be added as needed
}
else
    return false;
delete [] name;
name=NULL;
delete [] model;
model=NULL;
```

API:

CameraGetCount

CameraGetName

Get a list of resolutions and set the camera resolution

```
int index=0, width=0, height=0, camera_count=0;
CameraGetResolutionCount(m_index, &camera_count);    /*Get the number of resolutions of
the camera m_index*/
//Get camera resolution in turn
for(int i=0; i<camera_count; i++)
{
    CameraGetResolution( m_index, i, &width, &height);
    ...//Get camera resolution list and display resolution list
}
...//Additional feature codes can be added as needed
CameraSetResolution(m_index, index, &width1, &height1); /*Set the resolution of the
camera m_index to the index of the index group*/
```

API:

CameraGetResolutionCount

CameraGetResolution

CameraSetResolution

Start collecting and displaying images

Open a background thread and call CameraQueryImage to get the image. The thread is displayed by sending a Windows message WM\_PROCESS\_MESSAGE notification interface.

```
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;
}
```

#### API:

CameraGetImageSize  
CameraGetImageBufferSize  
CameraQueryImage  
CameraShowBufferImage

#### Dynamic plugging operation

#### Add USB plug response message

```
ON_WM_DEVICECHANGE()
```

Add a response function, check the number of devices, release the camera or reinitialize if there is a change

```
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;
}
```

API:

CameraGetCount

## 3.3 TwoCamera\_VCDemo

### 3.3.1 Function

The twoCamera\_VCDemo example implementation features: display of images captured by two cameras, one-button white balance, horizontal and vertical mirroring, ROI settings, camera parameter settings (including: automatic gain, auto exposure, gain, exposure, gamma value, contrast), parameter saving, etc.



## Description

1 Image display: The camera and corresponding resolution can be selected by the four drop-down buttons at the upper right. Click the Play button to display the camera image. Click the Stop button to stop the display. Select Camera0 or Camera1 to select The camera that is operating is selected.

2 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

3 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

4 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.

Where Row Start is the starting horizontal position of the ROI; Column Start is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

5 Camera parameter settings: By selecting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

6 Parameter saving function: Supports the saving function of two sets of parameters. Select Group1 or Group2 to switch between two sets of parameters. Click Save to save the corresponding parameter set.



### 3.3.2 Sample

#### Get camera list

```
int CamAllNum;//Number of cameras
CameraGetCount(&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//Get the camera resolution name in turn
if(CamAllNum>0)
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ...//Display camera list after getting camera list
    }
    ... //Additional feature codes can be added as needed
}
delete[] model;
model=NULL;
delete[] name;
name=NULL;
```

API:

**CameraGetCount**

**CameraGetName**

#### Get a list of resolutions and set the camera resolution

```
int index=0,width=0,height=0,camera_count=0;
CameraGetResolutionCount(m_index,&camera_count); //Get the number of resolutions of
the camera m_index
//Get camera resolution in turn
for(int i=0;i<camera_count;i++)
{
    CameraGetResolution( m_index,i,&width, &height);
    ...//Get camera resolution list and display resolution list
}
...//Additional feature codes can be added as needed
CameraSetResolution(m_index,index,&width1, &height1); /*Set the resolution of the
camera m_index to the index of the index group*/
```

API:

**CameraGetResolutionCount**



CameraGetResolution

CameraSetResolution

## Start collecting and displaying images

This example contains two sample projects that demonstrate two ways to get an image.

TwoCamera\_Demo    Callback function to get the image.

TwoCamera\_Demo\_Thread    Get images in thread mode.

## Thread mode:

Open a background thread and call CameraQueryImage to get the image. Display images can be displayed using CameraShowBufferImage or Windows API.

```
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))
        {
            ... //Display image after getting the image correctly
        }
        else
        {
            Sleep(100); //Did not successfully acquire images, wait
        }
        ...//Additional feature codes can be added as needed
        delete[] m_inBuf;
        m_inBuf=NULL;
    }
}
```

## API:

CameraGetImageSize

CameraGetImageBufferSize

CameraQueryImage

CameraShowBufferImage

## Callback function mode:

## Play

```
CameraPlay(0,hWnd0, SnapThreadCallback);    /*Display the screen of camera 0 in the
```





```
window with handle hWnd0*/
    CameraPlay(1, hWnd1, SnapThreadCallback1);    //Display the screen of camera 1 in the
window with handle hWnd1*/
```

## Stop

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

## Callback:

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height, int
format)
{
    /*The user can add code to process the acquired image, and the image information is stored
in pImageBuffer.*/
    return 0;
}
```

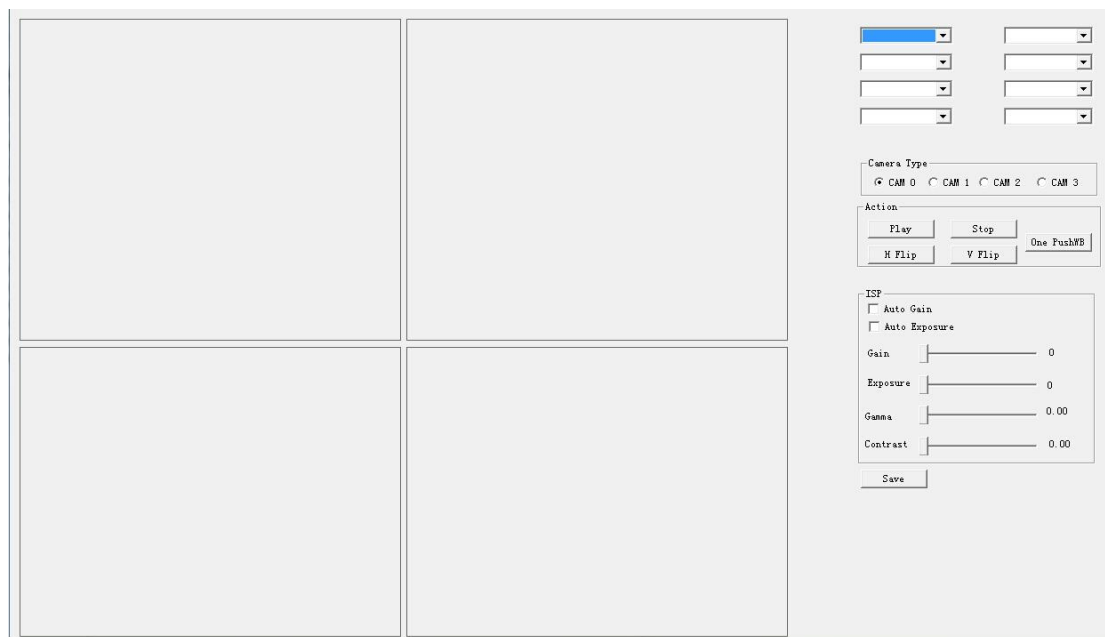
## API:

CameraPlay  
CameraStop

## 3.4 FourCamera\_VCDemo

### 3.4.1 Function

The functions of the FourCamera\_VCDemo example mainly include: displaying and operating images acquired by four groups of cameras, which can realize one-button white balance, horizontal and vertical mirroring operations, and camera parameter settings (automatic gain, auto exposure, gain, exposure, gamma, horse value, contrast and so on).



## Description

1 Image display: You can select four sets of cameras and corresponding resolutions by the eight drop-down buttons at the upper right. Click the Play button to display the camera image. Click the Stop button to stop the display. Select CAM0 and CAM1. The four controls CAM2 and CAM3 can control and operate the corresponding cameras separately.

2 One-button white balance: Click the One Push WB button to achieve one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a blank sheet of paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

3 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

4 Camera parameter setting: By setting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure. Slide the slider below to manually set the camera parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

## 3.4.2 Sample

### Initialize the camera list

```
int CamAllNum; //Number of cameras
CameraGetCount(&CamAllNum);
char *name=new char[255];
char *model=new char[255];
//Get the camera name in turn
if(CamAllNum>0)
```



```
{
    for(int i=0; i<CamAllNum; i++)
    {
        CameraGetName(i, name, model);
        ... //Display camera list after getting camera list
    }
    ...//Additional feature codes can be added as needed
}
delete[] model;
model=NULL;
delete[] name;
name=NULL;
```

API:

CameraGetCount

CameraGetName

Get a list of resolutions and set the camera resolution

```
int index=0, width=0, height=0, camera_count=0;
CameraGetResolutionCount(m_index, &camera_count);    /*Get the number of resolutions of
the camera m_index*/
//Get camera resolution in turn
for(int i=0; i<camera_count; i++)
{
    CameraGetResolution( m_index, i, &width, &height);
    ...//Get camera resolution list, display resolution list
}
...//Additional feature codes can be added as needed
CameraSetResolution(m_index, index, &width1, &height1); /*Set the resolution of the
camera m_index to the index of the index group*/
```

API:

CameraGetResolutionCount

CameraGetResolution

CameraSetResolution

Start collecting and displaying images

Play

```
CameraPlay(m_device_id, hImage[m_device_id], SnapThreadCallback);
/*The screen of the camera m_device_id is displayed in the window with the handle
hImage[m_device_id], and the SnapThreadCallback is a callback function, which can be
replaced by 0.*/
```



## Stop

```
CameraStop(m_device_id);           //Stop the camera numbered m_device_id
```

## Callback

```
int __stdcall SnapThreadCallback(unsigned char *pImageBuffer, int width, int height,
int format)
{
    /*The user can add code to process the acquired image, and the image information is stored
in pImageBuffer.*/
    return 0;
}
```

## API:

CameraPlay  
CameraStop

## 3.5 OpenCV

### 3.5.1 Function

OpenCV converts the camera's image data into an IplImage format. And demonstrate the call to the opencv image operator for image operations.

### 3.5.2 Sample

```
#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 exit
    if( key == 27 )
    {
        break;
    }
}
CameraFree(0);
cvReleaseImage( &image );
cvDestroyWindow( "Camera Example" );
return 0;
}
```

API:

CameraQueryImage

CameraGetImageBufferSize

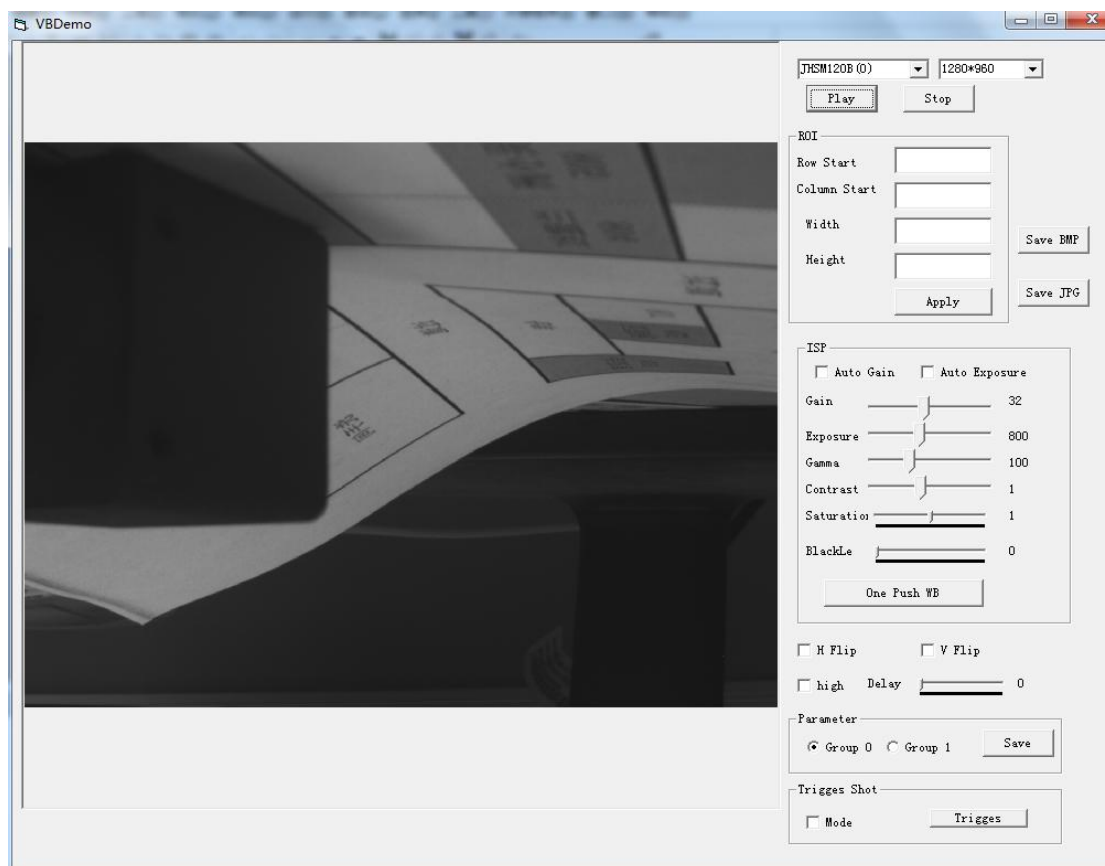


## 4 VB

### 4.1 OneCamera\_VBDemo

#### 4.1.1 Function

The functions implemented by OneCamera\_VBDemo mainly include: displaying images captured by the camera, ROI settings, saving pictures, achieving one-button white balance, horizontal and vertical mirroring operations, camera parameter adjustment and saving, camera high-speed setting, software triggering captured images, etc.



#### Description

1 Image display: The camera and corresponding resolution can be selected by the two drop-down buttons at the upper right. Click the Play button to display the camera image, and click the Stop button to stop the display.

2 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.

Where Row Start is the starting horizontal position of the ROI; Column Start is the starting



vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

3 Save the picture: You can intercept and save the image information displayed on the interface. Click Save BMP to save the picture in BMP format. Click Save JPG to save the picture in JPG format.

4 Parameter adjustment and saving: Support the saving function of two sets of parameters. Click Group1 and Group2 to switch between two sets of parameters. By selecting Auto Gain and Auto Exposure to set the camera to auto gain and auto exposure, slide the slider below to manually set the camera's parameters.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

6 Horizontal and vertical mirroring operation: Horizontal mirroring operation can be realized by selecting HFlip, and vertical mirroring operation can be realized by selecting VFlip.

7 High-speed setting: Select the high check box to enable high-speed setting of the camera, slide the corresponding scroll bar, and reduce the speed of the camera.

8 Trigger the image by software trigger: After the Mode can be entered into the trigger mode, click the Software trigger button to capture the image through software trigger.

## 4.1.2 Sample

### Initialize the camera list

```
'Get the number of cameras
Dim m_count As Long
m_count = 0
CameraGetCount m_count
'Camera name
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
'Display camera name list.....
Next i
```

API:

### CameraGetCount

Shenzhen Jinghang Technology Co., Ltd. Tel : 0755-21505130 <http://www.jinghangtech.com>



## CameraGetName

### Save Image

```
' Save JPG image
CommonDialogSave.DialogTitle = "Save As"
CommonDialogSave.Filter = "jpg File( *.jpg) | *.jpg"
CommonDialogSave.FilterIndex = 1
CommonDialogSave.ShowSave
Dim templ As String
templ = CommonDialogSave.filename
CameraSaveJpegB 0, templ, True

' Save BMP format image
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
```

### API:

**CameraSaveJpegB**

**CameraSaveBMPB**

Refer to **OneCamera--VB**



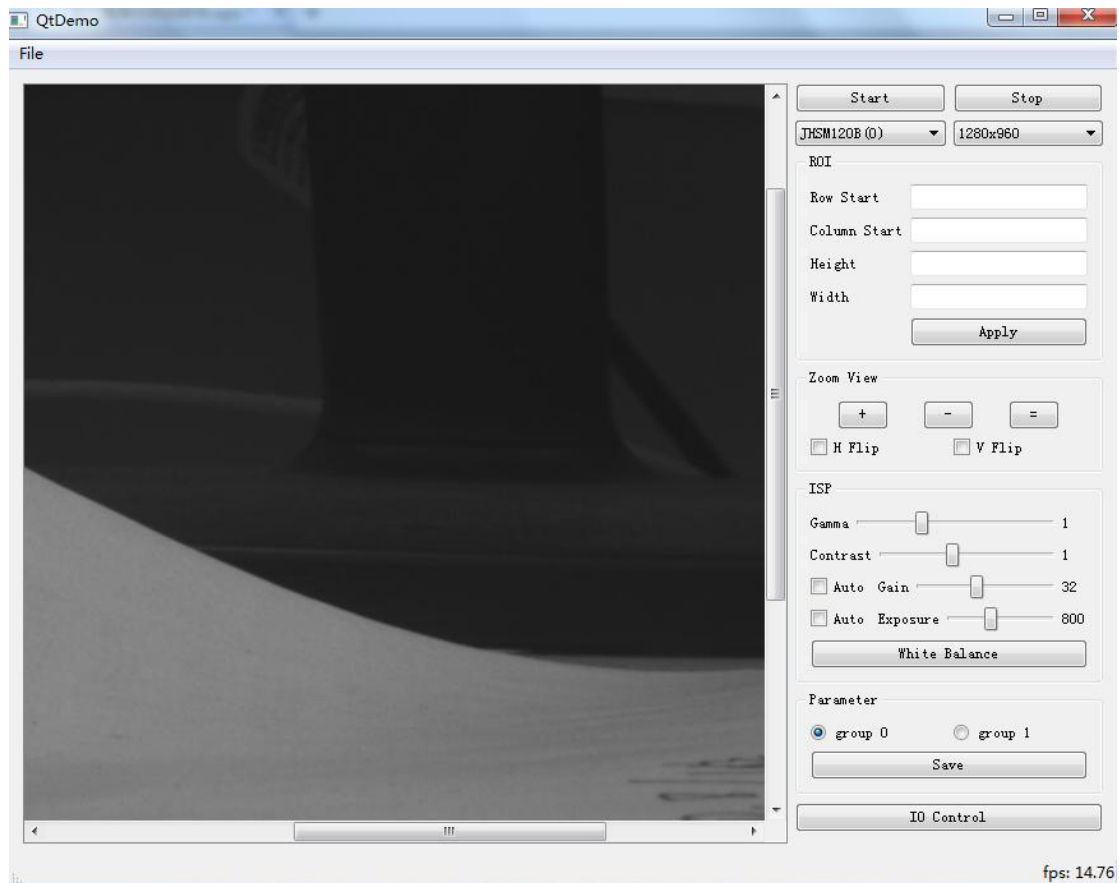


## 5 QT

### 5.1 OneCamera\_QtDemo

#### 5.1.1 Function

The functions implemented by OneCamera\_QtDemo.exe mainly include: display of images captured by the camera, ROI settings, horizontal and vertical mirroring operations, image reduction and enlargement, parameter adjustment and saving, one-button white balance, and camera GPIO settings.



#### Description

1 Image display: You can select the camera and the corresponding resolution through the two drop-down menus at the top right. Click the Start button to display the camera image, and click the Stop button to stop the display.

2 ROI setting: In the ROI control bar, input the ROI data you want to implement into Row Start, Column Start, Width, Height, and then click Apply to implement the ROI setting.



Where Row Start is the starting horizontal position of the ROI; Column Start is the starting vertical position of the ROI; Width is the width of the ROI; and Height is the height of the ROI. The values of Row Start and Column Start need to be greater than or equal to 0, and the values of Width and Height need to be less than (not including equal to) the width and height values at the current resolution.

3 Horizontal and vertical mirroring operation and image enlargement and reduction: Horizontal mirroring operation can be realized by selecting HFlip. Vertical mirroring operation can be realized by selecting VFlip. Click “+” button to zoom in and click “-” button to zoom in. Action, click “=” to restore the original image size.

4 Parameter saving function and parameter adjustment: Support the saving function of two sets of parameters. Click Group1 and Group2 to switch between two sets of parameters. By selecting Auto Gain and Auto Exposure, the camera can be set to auto gain and auto exposure, and the corresponding slider can be manually adjusted by sliding the corresponding slider.

Among them, Gain corresponds to the gain value, the range is 1~255; Exposure corresponds to the exposure value, the range is 1~5000; Gamma corresponds to the gamma value, the range is 0~2; Contrast corresponds to the contrast, the range is 0~2.

5 One-button white balance: Click the One Push WB button to achieve a one-button white balance on the camera. When achieving white balance, it is best to aim the camera at a piece of white paper. If it is not implemented at one time, you can click the one-button white balance button multiple times until the color is normal.

6 GPIO settings: Click the IO Control button to pop up the IO control panel, which can be set separately.

### 5.1.2 Sample

Initialize the camera list

```
int count;
CameraGetCount(&count);
for(int i=0; i<count; i++)
{
    char name[255], model[255];
    CameraGetName(i, name, model);
    //Display camera list.....
}
```

API:

CameraGetCount  
CameraGetName

Start collecting and displaying images

Capture Image

```
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;
}
```

### Connect messages and slots

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

### Slot function

```
void MainWindow::process(QImage img, unsigned char *buffer)
{
    //Display img images or other operations
}
```

### API:

CameraGetImageSize  
CameraGetImageBufferSize  
CameraQueryImage

## 6 Demo

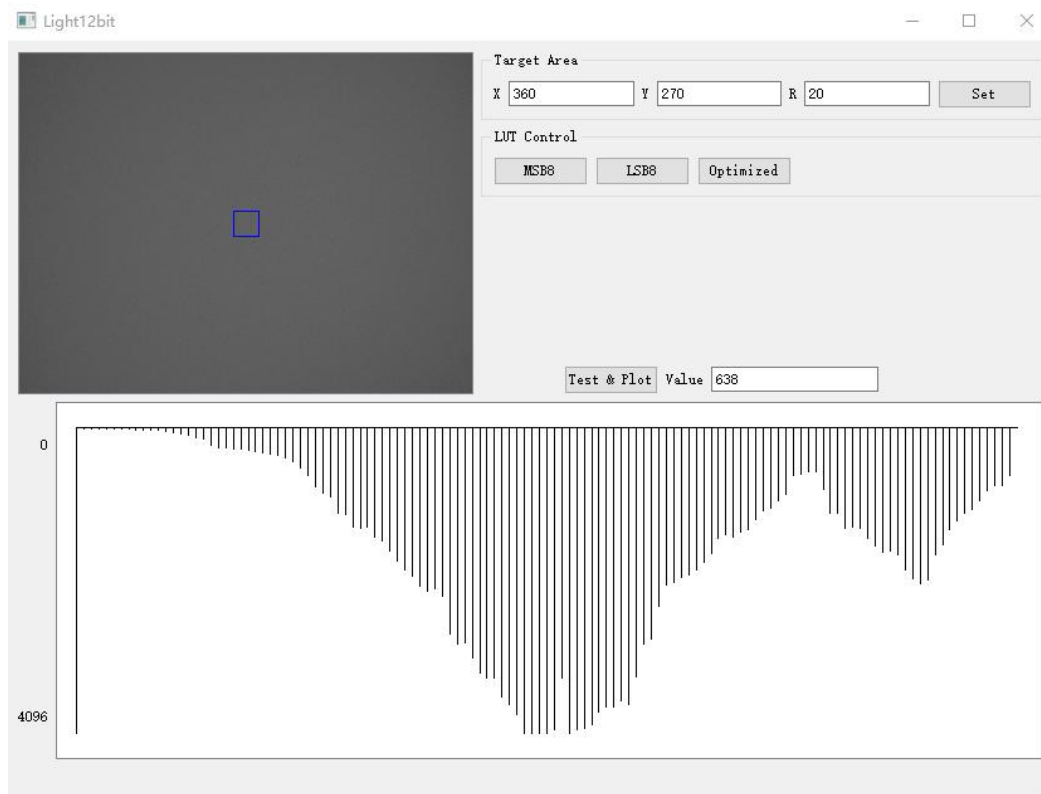
Demo programs provide prototypes for some practical application, with some special API.

### 6.1 Light12bit

#### 6.1.1 Function

This sample demonstrate a 12bit application probing the light intensity, and plot as a visual plot.

The detect range of the intensity is extended to 0-4095 with 12bit camera, comparing to 0-255 with 8bit camera. This is the major merit of 12bit camera in precise measurement applications. The ultimate display image is always 8bit, so a 12 bit image should be converted to 8 bit by a Look Up Table(LUT).



#### 6.1.2 Implementation

Up left part is the live image of the camera. The blue rectangle on the image is the Target Area for probing. The average intensity is averaged on the target area. The display image is applied LUT to show part of the whole 12bit data.



Target Area, defines a area whose center is (X,Y), and width and height are both 2\*R. Average intensity is calculated with the RAW12 data captured from camera.

LUT Control, select predefined LUT, MSB8, Most significant 8bit, LSB8, Least significant 8 bit, Optimized, improve the dark part of the image.

Test & Plot, capture a image, and calculate the average as Value, and plot a line indicate the intensity. The maximum number of lines is 126.

### 6.1.3 Sample Code

```
//initialize camera, get image depth, value ranges: 8bit (0-255) , 12bit (0-4095)
CameraInit(0);
CameraGetImageDepth(0, &m_depth);

// capture RAW12 data, 2 bytes per pixel
// CAMERA_IMAGE_RAW capture RAW8 or RAW12, depends on camera output.
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)
{
    //apply LUT to convert to GRAY8
    if(CameraISP(index, buffer, ISPbuf, width, height, CAMERA_IMAGE_GRAY8) == API_OK)
    {
        //buffer is RAW12, and ISPbuf is GRAY8 data
    }
}

//convert RAW12 buffer pointer from unsigned char to unsigned short
unsigned short *ptr = (unsigned short *)buffer;
//statistic
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];
    }
}
```



```
//get the avg finally  
double avg = sum/(4*m_r*m_r);
```

Related API:

CameraGetImageDepth  
CameraQueryImage  
CameraISP

## 6.2 LineScan

### 6.2.1 Function

This example demonstrate a line scan application. The camera shot on a rotating cylinder. The capability of line scan is that it can capture line by line on the surface and reconstruct a full flat image of the surface. Multiple image can be captured and stitching to a single image.

Camera should install properly to make line scan function correct working. The long side of the sensor should keep parallel with the rotating axis. Rotating direction is from back to front, where the label is on the front of the camera.





### 6.2.2 Implementation

Exposure, Gain, controlling the brightness of the image. There need very powerful light source to reduce the exposure time.

Line scan parameter : Line Height, how many lines exposures very time. As a area scan sensor, user can choose more or less lines in line scan application. The less, the more it like a line scan camera. The minimum number is 2.

Line Trigger, Auto , triggered by camera automatically. External, need hardware trigger to trigger every line, this demo only work with Auto.

Frames per Trigger, how many image will captured by a single software trigger.

FitView, adjust the size of the display image based on the window size. Save Image, save the stitching image as a image file.

Trigger Mode, enter trigger mode by check this option, the live image will keep still in this mode until a trigger signal is received. Soft Trigger, send a trigger signal. Enable FPN, fix the fix pattern noise on the image, which is always a significant defect in line scan mode.

### 6.2.3 Sample Code

```
//initialize the camera
CameraInit(0);
//set line scan related parameters
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);
//set gain & exposure
CameraSetGain(0, 32);
CameraSetExposure(0, 20);

//trigger camera
CameraTriggerShot(0);
```

Related API:

- CameraSetResolutionMode
- CameraSetLineHeight
- CameraSetLineTrigger
- CameraSetSnapNum
- CameraSetFPN