Helpdesk XIMEA

xiAPI.NET

Support SK - 2023-10-19 - in xiAPI.NET

https://www.ximea.com/support/wiki/apis/xiapinet

xiAPI.NET¶

xiAPI.NET stands for XIMEA Application Programming Interface for Dot Net - **Microsoft Visual C#**
It is a new common interface for all XIMEA cameras and represents a simplified version of generic xiAPI.

Architecture¶

API a is the software interface between the camera, operating system driver and application.
xiAPI.NET run-time is based on xiCam class implemented in *xiAPI.NET.dll* calling the *xiapi32.dll* on 32bit Windows operating systems and *xiAPI.NETX64.dll* calling the *xiapi64.dll* on 64bit Windows operating systems.

Installation¶

To add xiAPI.NET support:

- install XIMEA Windows Software Package
- When choosing components make sure: API and Examples are checked.
- Use one of the available versions of the .NET SDK for your application, supported versions are:
  - Microsoft .NET Framework 4.7.2
  - .NET Core 2.2
  - .NET Core 3.1
- We recommend to use Microsoft Visual Studio 2013 or a later version for application

development.

Sample¶

.NET Sample code - See this section with various .NET examples.
After installation of the XIMEA Windows Software Package - the sample project is installed in the directory XIMEA/Examples/xiAPI.NET.

Here follows a sample code that captures ten images from the camera and saves them as .bmp files.
Exposure time is set 2 ms and gain is set to 5 dB.

```
// Sample for XIMEA API.NET

using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Drawing;
using xiApi.NET;
namespace xiApi.NET_example
{
class Program
{
  static void Main(string[] args)
  {
    xiCam myCam = new xiCam();

    try
    {
      // Initialize first camera
      myCam.OpenDevice(0);

      // Set device exposure to 2 milliseconds
      int exposure_us = 2000;
      myCam.SetParam(PRM.EXPOSURE, exposure_us);

      // Set device gain to 5 decibels
      float gain_db = 5;
      myCam.SetParam(PRM.GAIN, gain_db);

      // Set image output format to monochrome 8 bit
      myCam.SetParam(PRM.IMAGE_DATA_FORMAT, IMG_FORMAT.MONO8);
```

```
      //Start acquisition
      myCam.StartAcquisition();

      // Capture images
      Bitmap myImage;
      int timeout = 1000;
      for (int i = 0; i < 10; i++)
      {
        myCam.GetImage(out myImage, timeout );
        string fName = string.Format("image{0}.bmp", i);
        myImage.Save(fName);
      }
      // Stop acquisition
      myCam.StopAcquisition();
    }

    catch (System.ApplicationException appExc)
    {
      // Show handled error
      Console.WriteLine(appExc.Message);
      System.Console.ReadLine();
      myCam.CloseDevice();
    }

    finally
    {
      myCam.CloseDevice();
    }
  }
} // end of Program
} // end of namespace xiApi.NET_example
```

Interface¶

The core of xiAPI.NET are the following functions, which allow the control of most of the camera functionality.

```
// open interface
xiCam.OpenDevice(int DevId);

// set parameter
myCam.SetParam(PRM paramName, int paramVal);
```

```
myCam.SetParam(PRM paramName, float paramVal);

// get parameter
myCam.GetParam(PRM paramName, out int paramVal);
myCam.GetParam(PRM paramName, out float paramVal);
myCam.GetParam(PRM paramName, out string paramVal);

// get next image from buffer
myCam.GetImage(out myImage);

// close interface
xiCam.CloseDevice();
```

Offline Processing¶

xiAPI.NET allows to process of already captured and stored images using Offline Processing. Please read more at xiAPI.NET Offline Processing.

Documentation¶

xiAPI.NET Manual

Accessing xiAPI.NET pixel data in .NET Bitmap