Workswell WIC SDK

With C# .NET samples

- Written in C# for .NET platform
- Communication and streaming using Gigabit and USB interfaces
- Fast conversion from raw signal data to temperature data

Version: 1.0.3.40

Datasheet

WORKSWELL

WIC SDK for C# .NET platform

Introduction

Workswell s.r.o. is pleased to introduce you WIC SDK for Windows which is written in C# for .NET platform and gives you full control of your WIC thermal camera. You might write programs in every .NET programming language using this SDK.

It supports communication and streaming using Gigabit and USB interfaces. Main goal of this WIC SDK is to allow you fast conversion from raw signal data to temperature data. Your thermal camera can be used in many ways, so this SDK architecture is designed to give you full support for all your projects and to give you possibility of quick software development as well.

All technical explanations are fully described in documentation, which you can find in your WIC SDK folder after installation of WIC SDK on your PC (usually in: *\Program Files\WIC_SDK\Documentation.chm*).

Workswell company has created WIC SDK samples to present some possibilities and to introduce its advantages. You are kindly invited to continue reading and let us explain those possibilities in this document.

Key Features

- Written in C# for .NET platform
- Possibility of writing programs in every .NET language using WIC SDK
- Support of communication and streaming using Gigabit and USB interfaces
- Fast conversion from raw signal data to temperature data
- Presented for Windows, but also Linux version available

Workswell WIC SDK

Phone numbers
+420 725 877 063
+420 725 955 464

E-mail and Web
info@workswell.eu
www.workswell.eu

ID
Reg. No.: 29048575
VAT No. CZ29048575

Headquarters
Libocka 653/51b
Prague, Czech Republic

General description

Discovering cameras and connection

Before discovering a camera, it is important to select folder with the camera licences which is necessary, if you wish to add your camera to WIC SDK collection of cameras. If at least one of the licences is appropriate for connected camera, this camera will be added to WIC SDK collection of cameras.

In the code example 1, which is below, there is a short example of usage of WIC SDK which describes connection to the first detected camera. First of all, there is defined the path to the directory of licences. If at least one camera is detected, system is asked to connect to the first one.

Discovering cameras and connection - Code example 1

```
1  CameraCenter cameraCenter; // cameraCenter variable
2  // Find the specific path to folder with licences (e.g. Documents)
3  string path =
4  Environment.GetFolderPath(Environment.SpecialFolder.CommonDocuments);
5  // CameraCenter already with the specific path
6  cameraCenter = new CameraCenter(path);
7  cameraCenter.CollectionChanged += cameraCenter_CollectionChanged;
8  // If at least one camera detected, connect to the first one
9  if (cameraCenter.Cameras.Count != 0)
10 {
11     Camera myCam = cameraCenter.Cameras.First();
12     // Connection to the camera
13     myCam.Connect();
14 }
```

Refreshing of the list of cameras - Code example 2

```
1  // Count available cameras
2  int count = cameraCenter.Cameras.Count;
3  if (count > FoundCameras.Count)
4  {   // New camera was found
4      Camera addedCam = cameraCenter.Cameras.Last();
5      FoundCameras.Add(new CameraViewModel(addedCam, this));
6  }
7  else
8  {   // One of the cameras was removed
9      List<int> removedIndexes = new List<int>();
10     foreach (var Cam in FoundCameras)
11     {
12         if (cameraCenter.Cameras.Where(x => x.SerialNumber ==
13 Cam.SerialNumber).Count() == 0)
14         {
15             removedIndexes.Insert(0, FoundCameras.IndexOf(Cam));
16         }
17 }
```

In case you have physically connected new camera, which can't be found, you may refresh the list of cameras and update it. It is similar with physical disconnection of camera. Both changes, such as addition or removal of camera will be visible, because the list of found cameras in WIC SDK will be updated.

Finally, in the list of cameras you see all cameras and you can select one to work with.

Settings

Every acquisition of the scene is different, so it should be possible to change thermal parameters of the environment, which affects a measurement, like Emissivity, Reflected temperature, Atmospheric temperature, Humidity and Distance. WIC SDK allows you all that and more.

You can also choose between low and high Gain (temperature range). Then there is a possibility to select one of predefined Palette and finally, you can select one of the Calibrated lens.

In the next code example, there is described settings of two parameters: Emissivity and Reflected temperature which were mentioned higher and both are adjustable.

Settings of the environment parameters - Code example 3

```
1  public float? Emisivity // Emisivity settings
2  {
3     get{…}
4     set
5     {
6           Settings.Emissivity = (float)value;
7     }
8  }
9  public float? ReflectedTemperature // Reflected temperature settings
10 {
11    get{…}
12    set
13    {
14          Settings.ReflectedTemperatureC = (float)(value);
15    }
16 }
```

As we have already mentioned, you can also change gain (temperature range). You are allowed to choose between two modes. First of them – law gain is made for measurement, that will be highly accurate in range from -40°C to 160°C. Second of those – high gain is preferred for measurement in much higher range, up to very high temperatures.

You might use one of those modes according to the present application, so that your final image corresponds to the reality and does not contain thermal noise.

Settings of Gain mode - Code example 4

```
public string SelectedGainMode // Selection of Gain Mode
{
    get{…}
    set
    {
        switch (value)
        {
            case "Low": // Switch gain to law
               myCam.GainMode = CameraSerialSettings.GainModes.Low;
               break;
            case "High": // Switch gain to high
               myCam.GainMode = CameraSerialSettings.GainModes.High;
               break;
            default:
               break;
         }
     }
}
```


Streaming from camera

After connect to the camera, it is convenient to see live stream from it. WIC SDK is also prepared for a requirement like this.

Using a method `StartAcquisition` you can start the acquisition from the selected camera, using a property `ThermoImageSource` you might get the thermal image from the camera which is currently connected. Eventually, for stopping the acquisition there is a similar method `StopAcquisiton`.

In the short code example below, you may see a usage of those commands.

System connects to the first detected camera, if there is some. Then the acquisition is started and finally the thermal image is acquired from camera.

Streaming from camera - Code example 5

```
// If at least one camera found
if (cameraCenter.Cameras.Count != 0)
{
    Camera myCam = cameraCenter.Cameras.First();
    // Connect to the first camera which was found
    myCam.Connect();
    // Start camera acquisition
    myCam.StartAcquisition();
    // Get source for thermal image
    BitmapSource ThermalImage = myCam.ThermoImageSource;
}
```

WIC SDK also gives you possibility of getting temperature array directly. This is made by a method `ConvertToTemperatureArray`, which belongs to `CameraSerialSettings` class and calculates temperatures.

Thanks to this possibility, it is also possible to get current temperature under your cursor. Short code example is below.

Current temperature in the thermal image area - Code example 6

```
// Get pixel temperature from the current position
public float GetPixelTemperature(int x, int y)
{
      int width = myCam.Settings.ResolutionX; // Width dimension of the image
      int height = myCam.Settings.ResolutionY; // Height dimension of the image

    // Get an array of all temperatures in the image
    float[] temperatures = Settings.ConvertToTemperatureArray(myCam.ImageInfo);

      // Return specific temperature for a positition
      return temperatures[y * width + x];
}
```

Getting device information

As it is very convenient and sometimes necessary to read device information, for these cases WIC SDK is also capable to find information about camera and to let you know them.

If you want to read specific information, you can simply ask WIC SDK for them. There is need to connect to the right camera instance, then you ask for a specific property and finally you might get it using getter method.

In the following code example, we are showing an example of how to get property like: Manufacturer info, Model Name and Serial Number.

Getting device information - Code example 7

```
// Manufacturer info
public string ManufacturerInfo
{
    get
    {
        return myCam.ManufacturerInfo;
    }
}
// Model name
public string ModelName
{
    get
    {
        return myCam.ModelName;
    }
}
// Serial number
public string SerialNumber
{
    get
    {
        return myCam.SerialNumber;
    }
}
```

As it is only the example, you can certainly read much more properties of your camera from WIC SDK. For example: Version of the device, User-defined device name, MAC address etc.

WIC SDK will automatically detect the maximum and minimum temperature so you might also read these values on every frame, without any need to go through temperature array. It is made by MaxTemperature and MinTemperature properties which belongs to Camera class.

Lens calibration

There is always some difference between two lenses. Every property of two lenses might be more or less different, which may cause the inaccuracy in conversion of raw data to temperature. That is why it is important to set correctly your camera's lens.

As you might use more than one lens, it is necessary to tell WIC SDK which one is currently used, so that it may use the appropriate calibration data for selected lens.

Searching for available calibrated lenses is demonstrated in the code example below.

Lens calibration - Code example 8

```
1  foreach (var lens in myCam.AvailableCalibratedLenses)
2  {
3      AvailableCalibratedLenses.Add(lens);
4  }
```

Conclusion

To conclude, WIC SDK was created by Workswell company and is very useful tool in cases when you deal with software development for WIC thermal camera. It will help you simplify your work a lot.

WIC SDK might be solution for your WIC thermal camera discovering and connection to PC, its settings and live streaming of thermal images and more. WIC SDK is definitely prepared to help you improve your own projects, where you can use your thermal camera in the unlimited number of ways.

This document describes WIC SDK in version for windows users, but we would like to ensure you, that also Linux and LabVIEW versions are available from Workswell company.

Useful download links

WIC SDK for Windows: http://www.workswell.cz/WIC_SDK/Windows/

WIC SDK for Linux x86: http://www.workswell.cz/WIC_SDK/Linux/

WIC SDK for ARM Linux: http://www.workswell.cz/WIC_SDK/Arm/

WIC SDK for LabVIEW: http://www.workswell.cz/WIC_SDK/Labview/

WORKSWELL

Contacts

Workswell WIC SDK

Sales Department

Adam Svestka, Msc., MBA

Mobile: +420 725 955 464

E-mail: adam.svestka@workswell.cz

Headquarters

Libocka 653/51b

161 00, Prague 6

Czech Republic

Branches

Mezericska 100

756 61, Roznov p. R.

Czech Republic

Univerzitni 1

010 08, Zilina

Slovakia

Company contact details

Mobile: +420 725 877 063

E-mail: info@workswell.eu

Web: www.workswell.eu