

Testo SE & Co. KGaA

# IrApi Documentation

## Table of contents

1	Introduction.....	2
1.1	IrApi .....	2
1.2	Supported operating systems .....	2
1.3	Supported languages.....	2
1.4	Supported cameras .....	2
2	Installation.....	3
2.1	Prerequisites.....	3
2.2	Main setup.....	4
2.3	Silent.....	6
2.4	Installed components.....	6
3	Getting started .....	7
3.1	Way to get it run.....	7
4	.NET .....	9
4.1	Project settings.....	9
4.2	References.....	11
4.3	Simple example .....	11
4.4	ThermalImageApi.dll .....	15
4.4.1	Temperature unit .....	15
4.4.2	Exception handling .....	15
4.4.3	Construction .....	15
4.4.4	Methods .....	15
5	C.....	18
5.1	Getting started .....	18
5.2	Simple example .....	18
5.3	ThermalImageApi.dll .....	25
5.3.1	Temperature Unit.....	25
5.3.2	ID.....	25
5.3.3	Return value .....	25
5.3.4	Methods .....	25

## 1 Introduction

### 1.1 IrApi

*IrApi* is a program library which makes it possible to access the content of the testo picture format (.bmt) from your own application program.

Main features of *IrApi*:

- Get information like device name, serial number or temperature of a point
- View thermal images
- View visual images

*IrApi* does not allow direct communication with the camera or to work with video. For that you will need CameraAPI (0501 8985) which only supports testo 885 and testo 890.

### 1.2 Supported operating systems

*IrApi* supports the following operation systems:

- Windows XP SP 3 or higher
- Windows Vista SP 2 or higher
- Windows 7 SP 1 or higher
- Windows 8 or higher

### 1.3 Supported languages

*IrApi* supports the following programming languages:

- C
- C#
- VB.Net

### 1.4 Supported cameras

*IrApi* supports the cameras testo 880, testo 875, testo 882, testo 870, testo 885 and testo 890. For support of testo 865, testo 868, testo 871 and testo 872 you need “IrAPI testo 87x” (0501 8987)

## 2 Installation

The installation consists of two parts, which automatically start by clicking *setup.exe*: the installation of prerequisites and the installation of *IrApi* itself.

### 2.1 Prerequisites

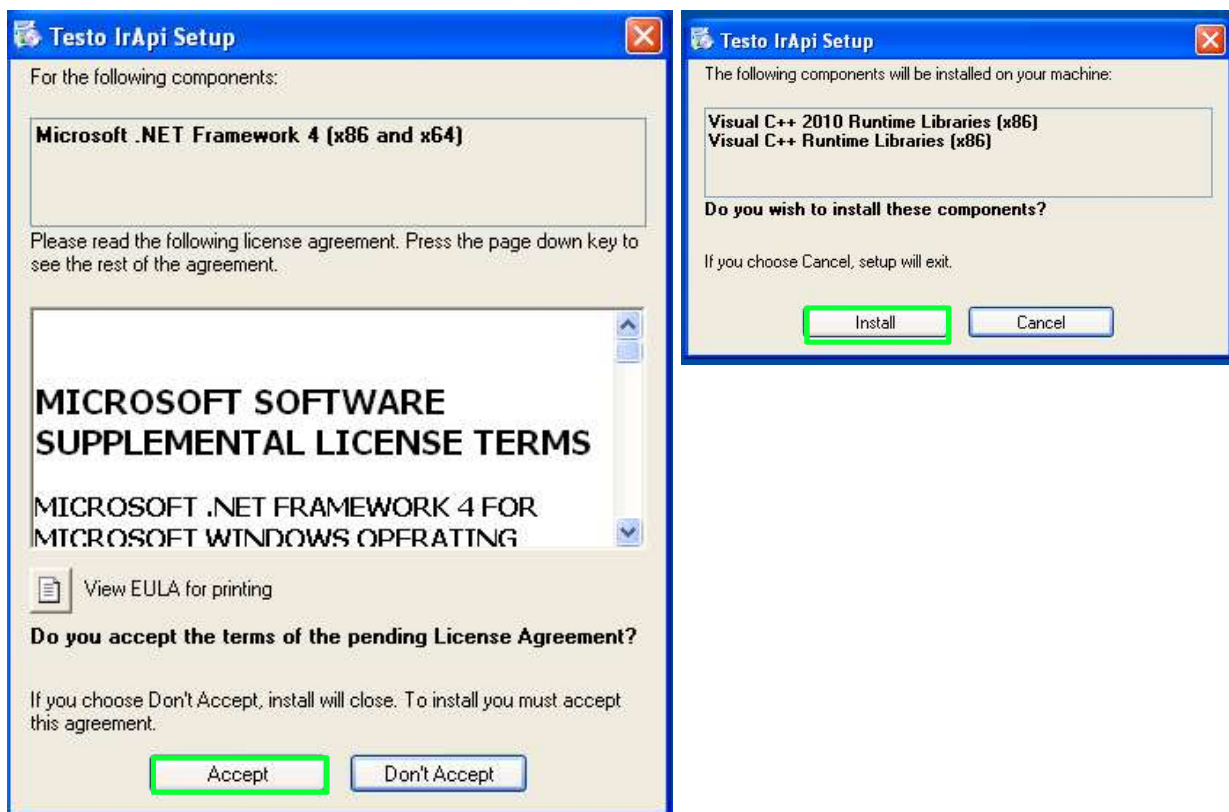
If not already installed, the *IrApi* installer will install following prerequisites:

- VCRedist.exe 2008
- VCRedist.exe 2010 x86
- .Net-Framework 4.0

For operating systems based on x64 it would additionally add:

- VCRedist.exe 2010 Sp1 x64

You must allow the installation of the prerequisites. The setup will prompt you following dialogs:



Please choose "Accept" and "Install" to continue the setup.

## 2.2 Main setup

The main setup installs *IrApi* itself. The setup will show you a welcome dialog, where you continue by click “Next”.



To continue you must accept the End-User License Agreement (EULA) and click next.



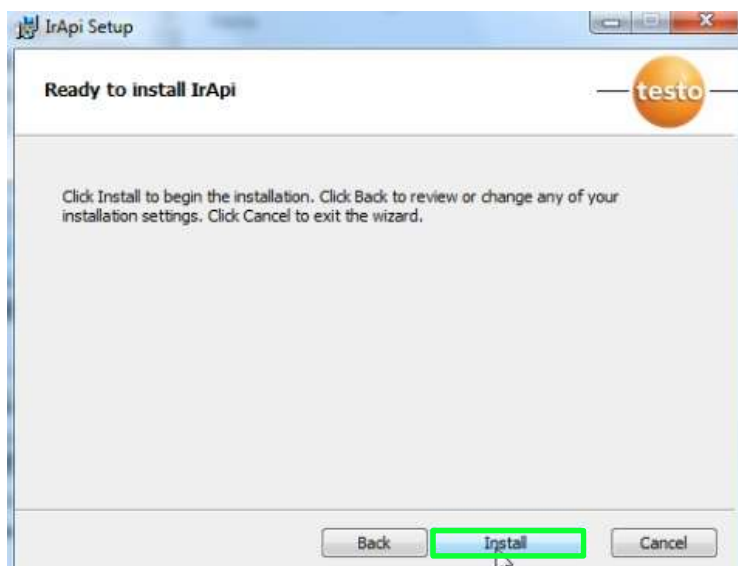
Now you can choose the path for the *IrApi* installation. By default the *IrApi* would install in “%Programfiles%\Testo\IrApi” or under x64 “%Programfiles(x86)%\Testo\IrApi”.

Be careful not to choose the path of IRSoft.

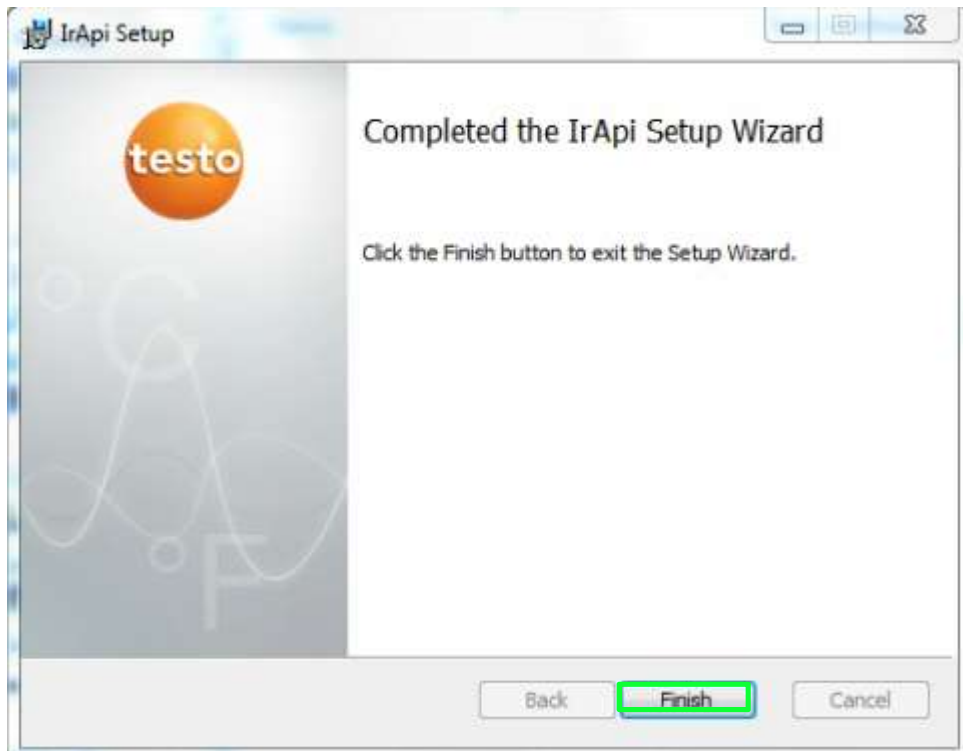
Click *Next* to continue.



As last step you must allow the installation to start. Therefore click "Install".



After the installation is finished, it will show you a summary.



## 2.3 Silent

It is **not** possible to run the whole setup as a silent setup so no user interaction is necessary. If you need a silent installation, you can install the prerequisites and the IrApi.msi silently. Be careful for all prerequisites to be installed before you install IrApi.msi.

## 2.4 Installed components

Besides the files needed by *IrApi* and this documentation you find a folder *examples* in the installation folder of *IrApi*. In this folder you can find a solution created with Visual Studio 2010 including one example usage with C# und one using plan C.

If you do not have the possibility to use Visual Studio 2010 please have a look at code in the subfolder of the example directory.

### 3 Getting started

There are components which are required to be in your application target directory. Otherwise the application will fail to call any *IrApi* function.

#### 3.1 Way to get it run

Please ensure that your application finds the following files originally located in the installation path of the *IRApi*:

*opencv\_calib3d242.dll*  
*opencv\_calib3d242d.dll*  
*opencv\_contrib242.dll*  
*opencv\_contrib242d.dll*  
*opencv\_core242.dll*  
*opencv\_core242d.dll*  
*opencv\_features2d242.dll*  
*opencv\_features2d242d.dll*  
*opencv\_flann242.dll*  
*opencv\_flann242d.dll*  
*opencv\_highgui242.dll*  
*opencv\_highgui242d.dll*  
*opencv\_imgproc242.dll*  
*opencv\_imgproc242d.dll*  
*opencv\_legacy242.dll*  
*opencv\_legacy242d.dll*  
*opencv\_ml242.dll*  
*opencv\_ml242d.dll*  
*opencv\_nonfree242.dll*  
*opencv\_nonfree242d.dll*  
*opencv\_objdetect242.dll*  
*opencv\_objdetect242d.dll*  
*opencv\_photo242.dll*  
*opencv\_photo242d.dll*  
*opencv\_stitching242.dll*  
*opencv\_stitching242d.dll*  
*opencv\_video242.dll*  
*opencv\_video242d.dll*  
*opencv\_videostab242.dll*  
*opencv\_videostab242d.dll*

*UsbLocID100.dll*  
*wdapi1020.dll*  
*zlib1.dll*

*t880\_x.txt*  
*t880\_y.txt*  
*t881\_t875\_x.txt*  
*t881\_t875\_y.txt*  
*t882\_x.txt*  
*t882\_y.txt*  
*TiLibAnalyzing.dll*  
*TiLibCam.dll*  
*TiLibCamComGen3.dll*  
*TiLibCamComGen3Usb.dll*



*TiLibCodecs.dll*  
*TiLibColor.dll*  
*TiLibCommon.dll*  
*TiLibCore.dll*  
*TiLibDataTypes.dll*  
*TiLibEnhancement.dll*  
*TiLibFeatureDetection.dll*  
*TiLibMarker.dll*  
*TiLibMath.dll*  
*TiLibRadiometry.dll*  
*TiLibStitching.dll*  
*TiLibStreaming.dll*

*ThermalImageApi.dll*  
*Emgu.CV.dll*  
*Emgu.Util.dll*  
*Testo.Framework.Archive.BL.Interfaces.dll*  
*Testo.Framework.Archive.DA.Interfaces.dll*  
*Testo.Library.Measurement.dll*  
*TestoIRArchive.Interfaces.dll*  
*TestoIRBase.dll*  
*TestoIRImage.dll*

You can do that by editing the global path environment variable or by simply copying all the files into the folder your application is running in.

## 4 .NET

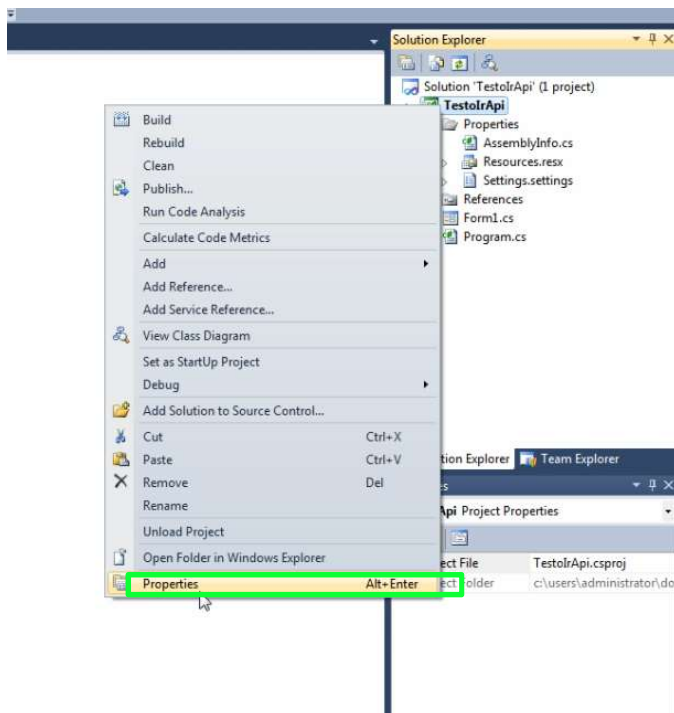
In the following section the usage of .NET interface is described.

The screenshots are made using Visual Studio .NET 2010 but of course you can use every build environment you want.

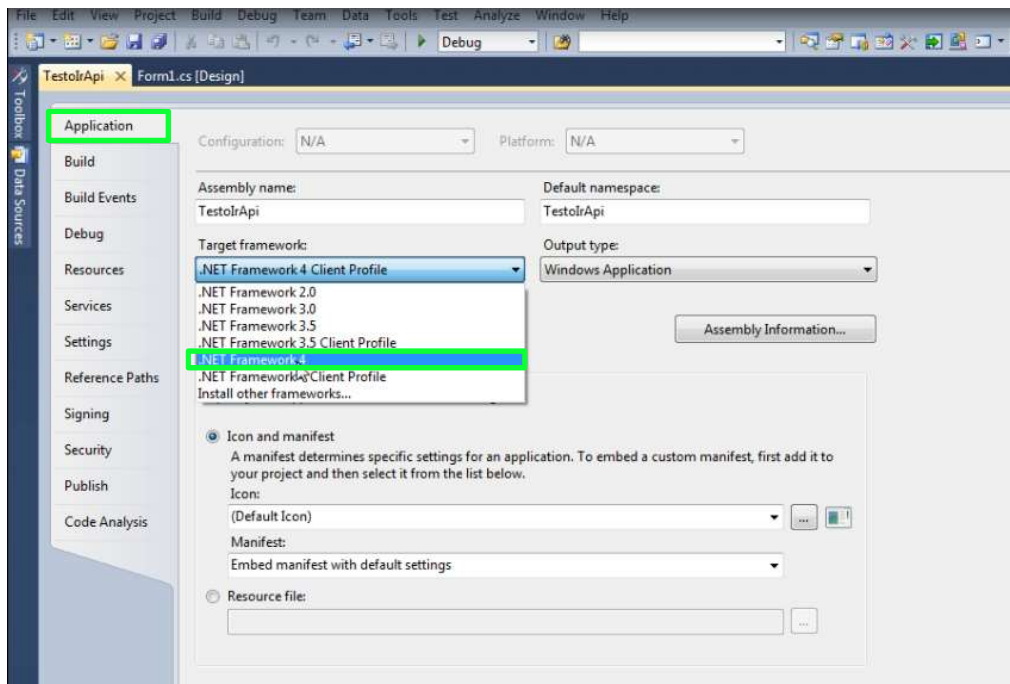
### 4.1 Project settings

Please select “.Net Framework 4” as target framework.

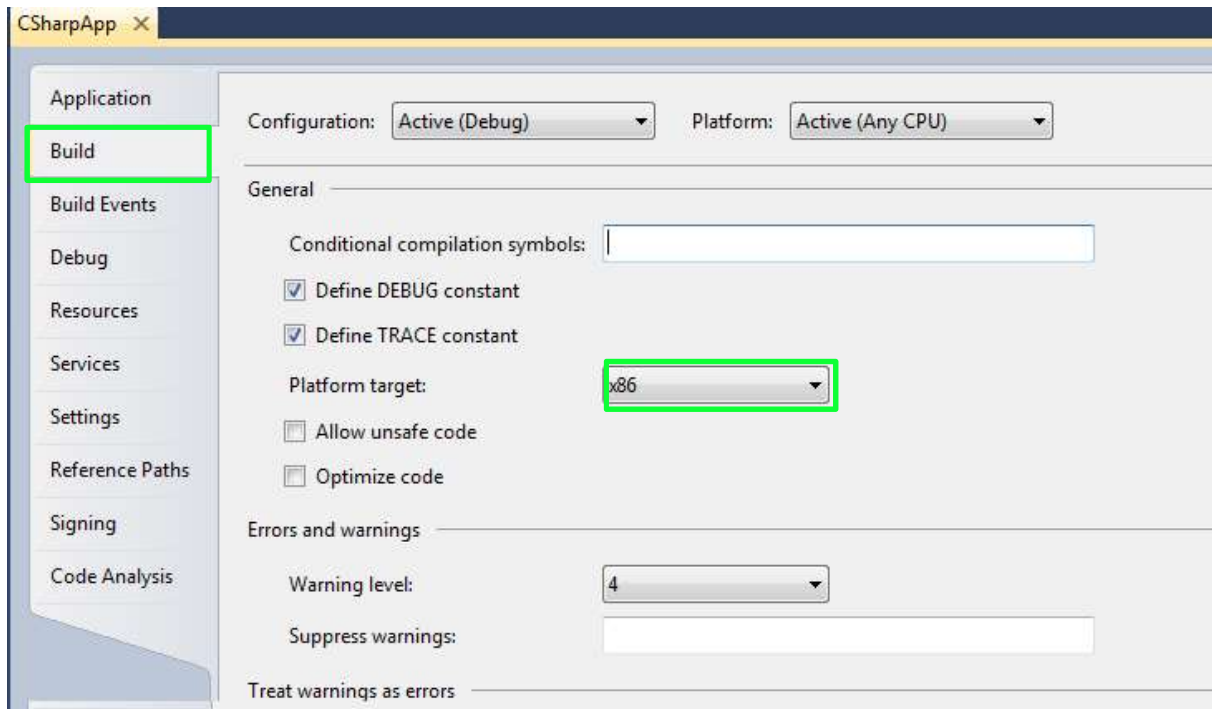
In Visual Studio 2010 you can do that by editing the properties of the solution.



Select Application and choose .NET Framework 4 as target framework.



Select Build and set Platform target to x86.



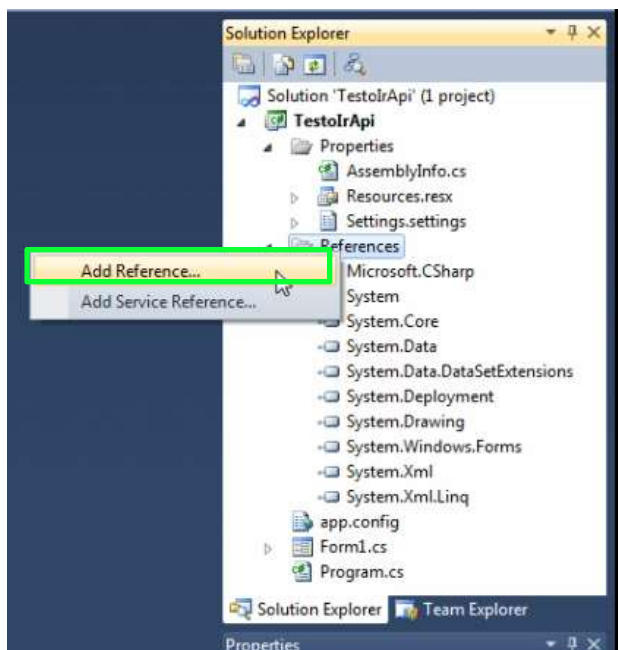
## 4.2 References

Please set references to the following 3 DLLs:

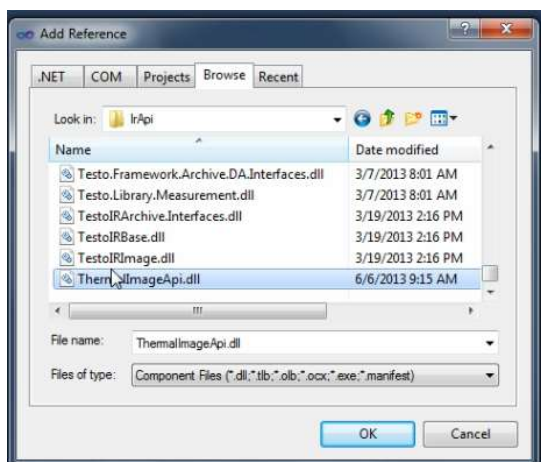
- ThermalImageApi.dll
- Testo.Library.Measurement.dll
- Testo.IRImage.dll

All of this files you can find in the installation folder of *IrApi*.

In Visual Studio 2010 you can do this by choosing “AddReference...” of the context menu of the solution explorer.

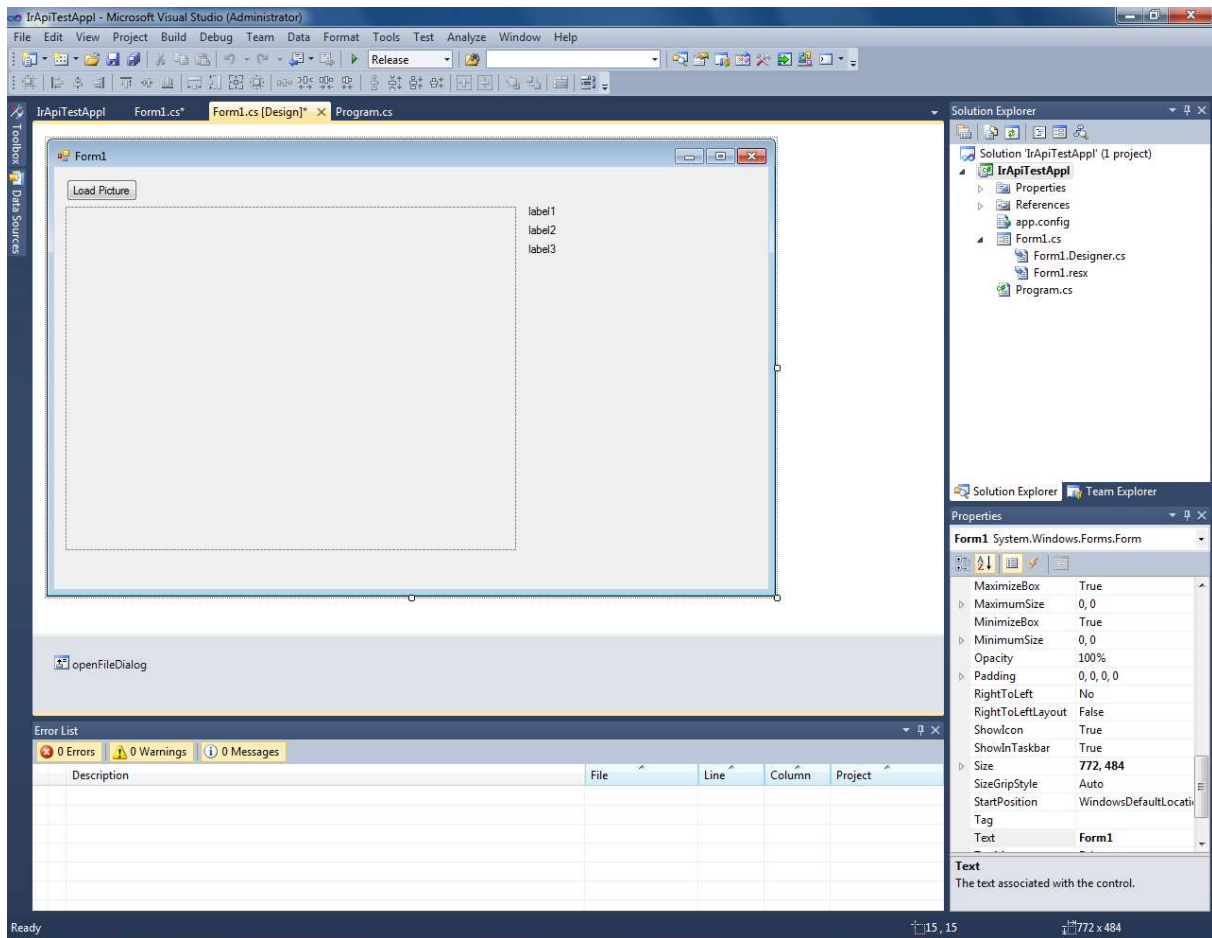


A new window pops up and you must select Browse. Browse to the installation folder of *IrApi*, mark the file and press ok.



## 4.3 Simple example

Create a WinForms application like this.



Add using directives and instantiate the ThermalImageApi.dll.

```
ThermalImageApi tImage = new ThermalImageApi();
```

Now you can open picture using the following line.

```
tImage.Open(openFileDialog.FileName);
```

With the information in tImage you could read out e.g.: the device name, serial number or the Field of View (FoV).

```
deviceLabel.Text = "Device: " + tImage.DeviceName;
serialNumberLabel.Text = "Serial Number: " + tImage.SerialNumber;
foVLabel.Text = "FoV: " + tImage.FoV;
```

In the end you can show the image as thermal image in a picturebox. That could be done with following code.

```
pictureBox.Image = tImage.GetThermalImage(Unit.GradC);
```

The whole code could look like this:

```
using System;
using System.Windows.Forms;
using Testo.IRSoft.API.Image;
using Testo.Library.Measurement;
```

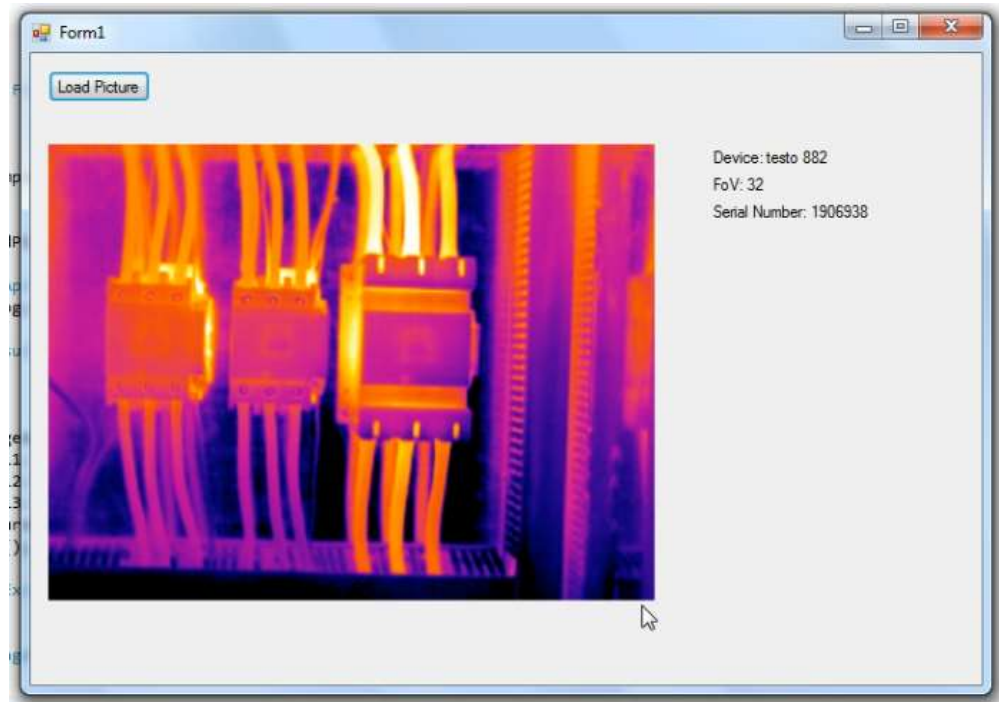
```

namespace IrApiTestApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void LoadPicButton_Click(object sender, EventArgs e)
        {
            openFileDialog.Filter = "bmt files (*.bmt)|*.bmt";
            if (DialogResult.OK == openFileDialog.ShowDialog())
            {
                try
                {
                    ThermalImageApi tImage = new ThermalImageApi();
                    tImage.Open(openFileDialog.FileName);
                    deviceLabel.Text = "Device: " + tImage.DeviceName;
                    serialNumberLabel.Text = "Serial Number: " + tImage.SerialNumber;
                    foVLabel.Text = "FoV: " + tImage.FoV;
                    pictureBox.Image = tImage.GetThermalImage(Unit.GradC);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }
            }
        }
    }
}

```

If you run this code, you should see something similar like this:



## 4.4 ThermallImageApi.dll

The ThermallImageApi.dll provides following methods and properties to work with *IrApi*.

### 4.4.1 Temperature unit

All temperatures are in °C.

### 4.4.2 Exception handling

All internal exceptions of the *IrApi* are of the type *IRApiImageException*, which is derieved from *System.Exception*.

### 4.4.3 Construction

The ThermallImageApi has a standard constructor.

```
ThermalImageApi();
```

### 4.4.4 Methods

<b>Name</b>	Open
<b>Parameter</b>	<code>string</code> path
<b>Return type:</b>	<code>void</code>
<b>Description:</b>	Opens the file and loads the .bmt-image into the memory.

<b>Name</b>	GetThermalImage
<b>Parameter</b>	Testo.Library.Measurement. <code>Unit</code> value
<b>Return type:</b>	<code>Bitmap</code>
<b>Description:</b>	Gets a thermal image from .bmt-file. Attention: use only the temperature units GradC or GradF.

<b>Name</b>	GetThermalImageWithPalette
<b>Parameter</b>	Testo.Library.Measurement. <code>Unit</code> value
<b>Return type:</b>	<code>Bitmap</code>
<b>Description:</b>	Gets a thermal bitmap with scale from .bmt-file. Attention: use only the temperature units GradC or GradF.

<b>Name</b>	GetVisualImage
<b>Parameter</b>	Non
<b>Return type:</b>	<code>Bitmap</code> /NULL
<b>Description:</b>	Gets the real image from .bmt-file if existing; null otherwise.

<b>Name</b>	GetMeasurementRange
<b>Parameter</b>	<code>ref float</code> min, <code>ref float</code> max
<b>Return type:</b>	<code>void</code>
<b>Description:</b>	Returns the measurement range information of the .bmp-fil.

<b>Name</b>	GetTemperature
<b>Parameter</b>	<code>int</code> x, <code>int</code> y
<b>Return type:</b>	<code>float</code>
<b>Description:</b>	Returns the temperature value of a point of the .bmt-file. The point of origin is on the upper left corner.

#### 4.4.4.1 Device information

<b>Name</b>	DeviceName
<b>Return type:</b>	<code>string</code>
<b>Getter/Setter:</b>	Getter
<b>Description:</b>	Returns the type of IR-Camera.



**Possible Value:** Any camera  
e.g.: testo 882  
t890-2

**Name** FoV (Field of View)  
**Return type:** `int`  
**Getter/Setter:** Getter  
**Description:** Returns the view angle in angle degree.  
E.g.: an angle of 42° indicates wide angle lens.

**Name** SerialNumber  
**Return type:** `uint`  
**Getter/Setter:** Getter  
**Description:** Returns the serial number of IR-Camera.

#### 4.4.4.2 Picture information

**Name** CreationDateTime  
**Return type:** `DateTime`  
**Getter/Setter:** Getter  
**Description:** Returns the date and time of shooting the IR-Image. The date/time would written in the currently thread culture. See Microsoft for more information.

**Name** Height  
**Return type:** `int`  
**Getter/Setter:** Getter  
**Description:** Returns the height of the picture in Pixel.

**Name** Width  
**Return type:** `int`  
**Getter/Setter:** Getter  
**Description:** Returns the width of the picture in Pixel.

**Name** ReflectedTemperature  
**Return type:** `double`  
**Getter/Setter:** Getter and Setter  
**Description:** Returns or sets the reflected temperature value for an IR-Image. This value has influence on the calculation of the temperatures.

**Name** Emissivity  
**Return type:** `double`  
**Getter/Setter:** Getter and Setter  
**Description:** Returns or sets the emissivity value for an IR-Image. This value has influence on the calculation of the temperatures.

**Name** MinScaleTemperature  
**Return type:** `float`  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the minimal scale temperature value. This value stands for the lowest scale temperature.

**Name** MaxScaleTemperature  
**Return type:** `float`  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the maximal upper scale temperature value. This value stands for the highest scale temperature.

**Name** Palette  
**Return type:** `Testo.IRSoft.Image.Palette`  
**Getter/Setter:** Getter and Setter  
**Description:** Returns or sets the color palette of the picture.

**Possible Value:** IronBow = 0,  
RainBow = 1,  
GreyScale = 2,  
GreyScaleInv = 3,  
Sepia = 4,  
BlueRed = 5,  
HotCold = 6,  
Testo = 7,  
DewPoint = 8,  
Hochtemp = 9,  
RainbowHC = 10

#### 4.4.4.3 Humidity

**Name** Humidity  
**Return type:** double  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the humidity in %rH.

#### 4.4.4.4 Highlighting

**Name** UseLimits  
**Return type:** bool  
**Getter/Setter:** Getter and setter  
**Description:** Turns on or off if limits are used. Returns current state.

**Name** UseIsotherm  
**Return type:** bool  
**Getter/Setter:** Getter and setter  
**Description:** Turn on or off if isotherm highlighting is used. Returns current state

**Name** LowerIsoTemperature  
**Return type:** float  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the lower isotherm temperature value. This value stands for the lowest highlighted isotherm temperature.

**Name** UpperIsoTemperature  
**Return type:** float  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the upper isotherm temperature value. This value stands for the highest highlighted isotherm temperature.

**Name** LowerLimitTemperature  
**Return type:** float  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the lower limit temperature value. This value stands for the lowest limit temperature.

**Name** UpperLimitTemperature  
**Return type:** float  
**Getter/Setter:** Getter and setter  
**Description:** Returns or sets the upper limit temperature value. This value stands for the highest limit temperature.

## 5 C

In the following section the usage of the C interface is described.

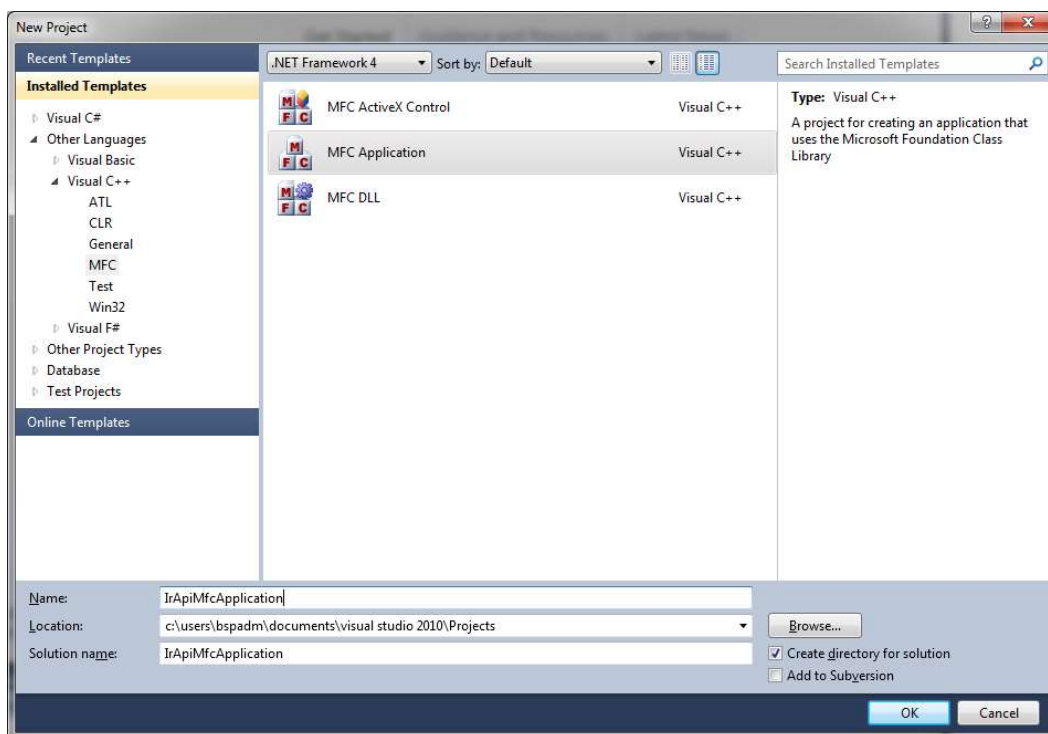
The screenshots are made using Visual Studio 2010 but of course you can use every build environment you want.

### 5.1 Getting started

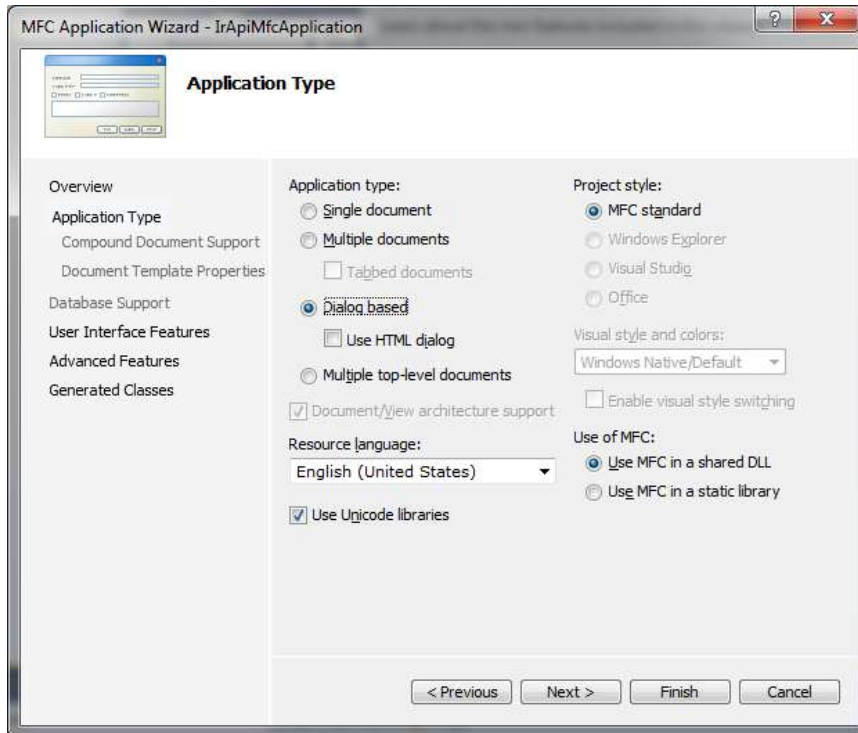
There are components which need to be loaded in your application target directory. Otherwise the application will fail to call any IrApi function. Make sure that all Files from chapter “3.1 Way to get it run” are in the target directory of your program.

### 5.2 Simple example

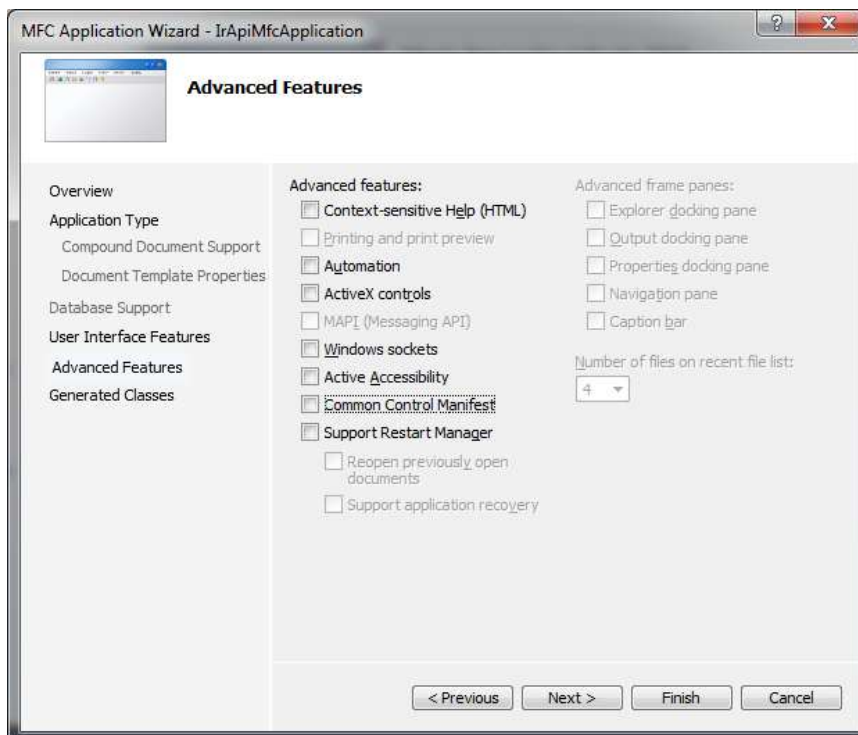
In this simple example you we use a MFC Dialog Application.



Choose “Dialog based” Application Type.

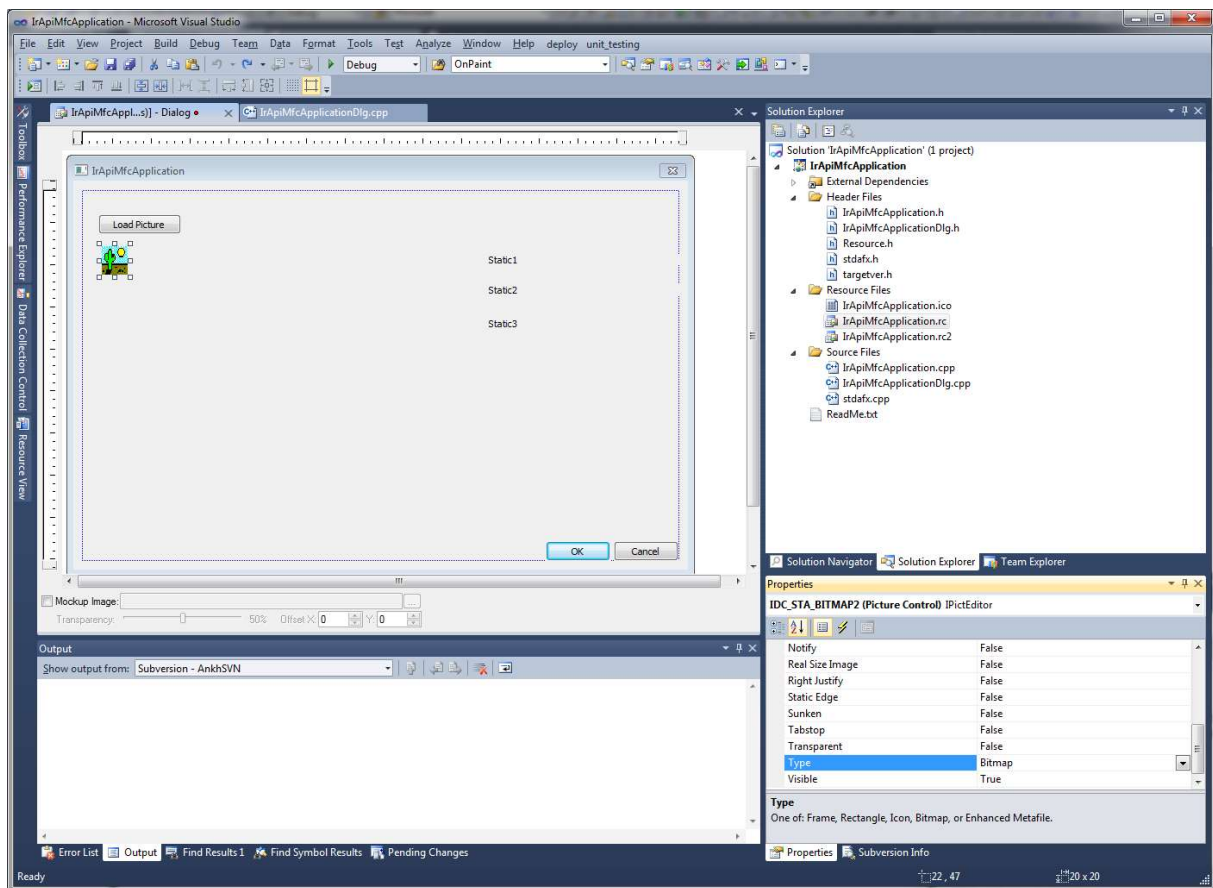


You can deselect advanced features.



You can “Finish” the Application Wizard.

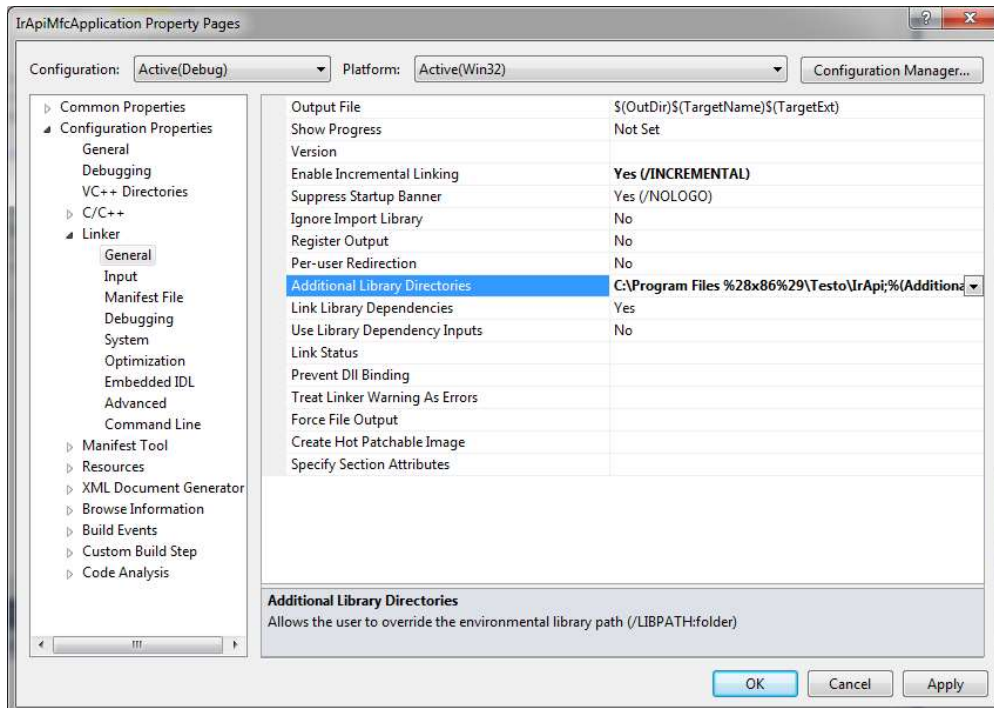
Place some Controls on your Dialog.



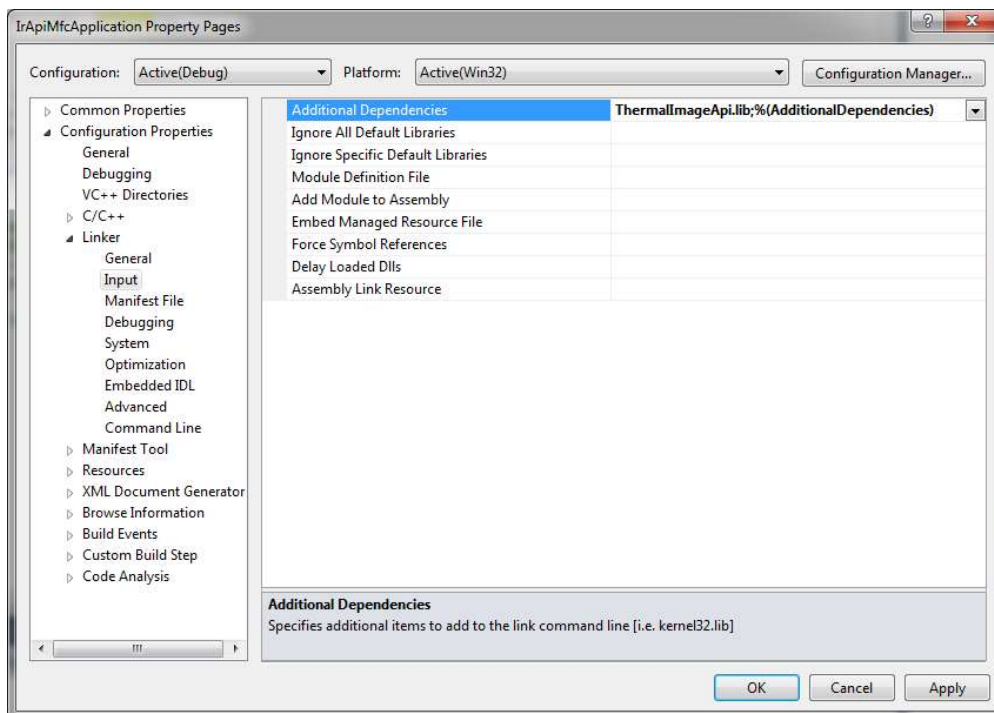
For the Picture Control you should use the “Bitmap” type.

To use the C interface you should include the ThermallImageApi.lib to your project. You can do that by editing the project properties.

Add your installation folder of the IrApi as an “Additional Library Directory”.

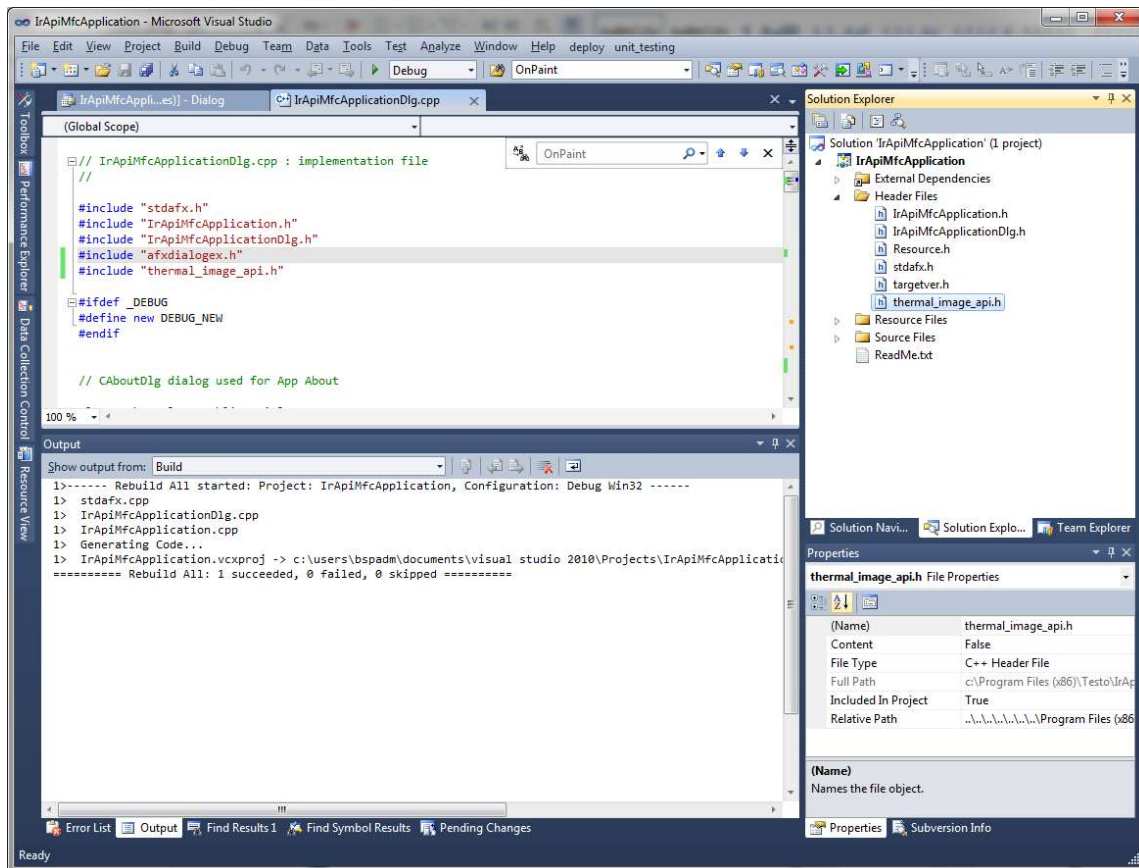


Add the ThermallImageApi.lib as an “Additional Dependency”.

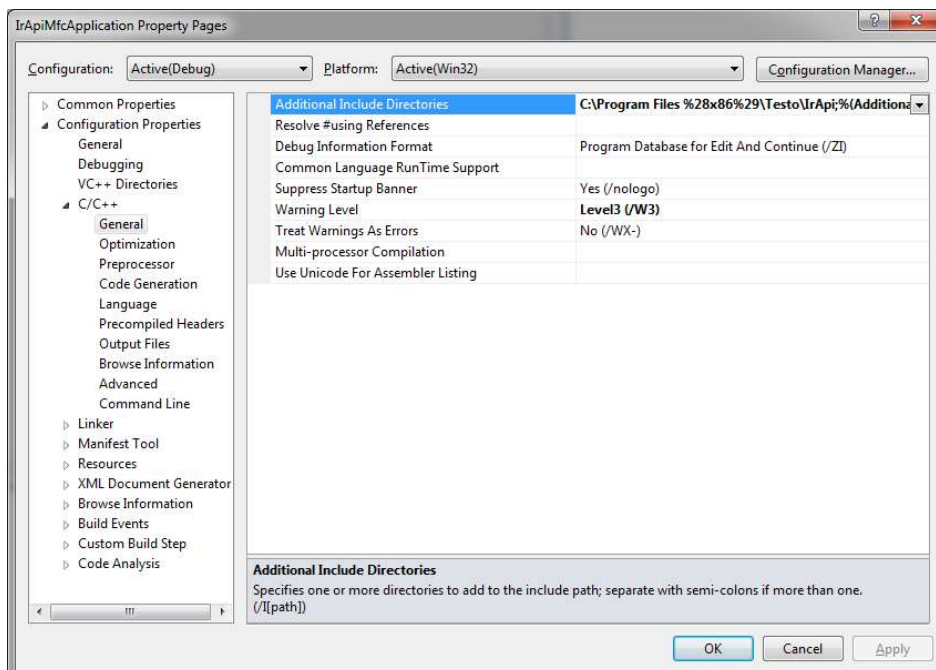


For calling the IrApi functions you need to include the thermal\_image\_api.h (header) file.





You should add an “Additional Include Directory”



With this settings you have done so far, you can implement the “Load Picture” event handler.

```
// <summary>
/// Called when clicked on loadimage button.
/// </summary>
void CIrApiMfcApplicationDlg::OnBnClickedButLoadpicture()
```

```

{
    wchar_t resultBuffer[100];
    try
    {
        int id;
        USES_CONVERSION;
        LPCTSTR fileName = _T("*.bmt");
        CFileDialog fileDlg(TRUE/*Open=TRUE Save=False*/,
            _T("bmt")/*Filename Extension*/,
            fileName/*Initial Filename*/,
            OFN_ENABLESIZING|OFN_EXPLORER|OFN_FILEMUSTEXIST/*Flags*/,
            _T("*.bmt")/*Filetype Filter*/,
            this/*parent Window*/);
        if (fileDlg.DoModal() == IDOK)
        {
            CString fileName = fileDlg.GetPathName();

            // get the ir image
            TESTO_IRAPI_RESULT tresult = testo_irimage_open(&id, (const wchar_t*)fileName);
            swprintf(resultBuffer, 100, L"Result: %s\n", testo_irimage_error_string(tresult));

            wchar_t path[_MAX_PATH];
            tresult = testo_irimage_thermal_image(id, TESTO_IRAPI_UNIT_CELSIUS, sizeof(path), path);
            swprintf(resultBuffer, 100, L"Result: %s\n", testo_irimage_error_string(tresult));
            HBITMAP startBitmap = (HBITMAP)LoadImage(NULL, path, IMAGE_BITMAP, 400, 300, LR_LOADFROMFILE);
            m_picture.SetBitmap(startBitmap);

            const int textLength = 100;
            wchar_t text_buffer[textLength+1];
            wchar_t display_text[textLength+1];

            // get the device name
            tresult = testo_irimage_get_devicename(id, textLength, text_buffer);
            swprintf(resultBuffer, 100, L"Result: %s\n", testo_irimage_error_string(tresult));
            swprintf(display_text, textLength, L"Device: %s\n", text_buffer);
            m_label1.SetWindowTextW(display_text);

            // get the serial number
            unsigned int serial;
            tresult = testo_irimage_get_serialnumber(id, &serial);
            swprintf(resultBuffer, 100, L"Result: %s\n", testo_irimage_error_string(tresult));
            swprintf(display_text, textLength, L"Serial Number: %d\n", serial);
            m_label2.SetWindowTextW(display_text);

            // get the FOV
            int fov;
            tresult = testo_irimage_get_fov(id, &fov);
            swprintf(resultBuffer, 100, L"Result: %s\n", testo_irimage_error_string(tresult));
            swprintf(display_text, textLength, L"FoV: %d\n", fov);
            m_label3.SetWindowTextW(display_text);

            testo_irimage_close(id);
        }
    }
    catch(...)
    {
        MessageBox(resultBuffer);
    }
}

```

In this handler, with the CFileDialog, you can choose a picture file (.bmt). Then this file is opened with `testo_irimage_open` and an image is created with `testo_irimage_thermal_image`. This image is then loaded and displayed in the picture control of the dialog. Additionally the device name, serial number and FOV are read and displayed in the static text controls on the right side of the picture.

If you run this code, you should see something like this:





## 5.3 ThermalImageApi.dll

### 5.3.1 Temperature Unit

All temperatures are in °C.

### 5.3.2 ID

While testo\_irimage\_open IrApi creates an imageID. This imageID would be needed from all methods.

### 5.3.3 Return value

TESTO\_IRAPI\_OK would return if everything is ok, otherwise an errorcode would be transmitted.

### 5.3.4 Methods

Name	testo_irimage_open
Parameter	non
Reference parameter:	<code>int*</code> id, <code>const wchar_t*</code> fname
Description:	Opens the file and loads the .bmt-image into the memory.
Name	testo_irimage_close
Parameter	<code>int</code> id
Description:	Closes the .bmt-image and unloads it from the memory
Name	testo_irimage_thermal_image
Parameter	<code>int</code> id, <code>TESTO_IRAPI_UNIT</code> unit, <code>int</code> length
Reference parameter:	<code>wchar_t*</code> path
Description:	Gets a thermal image from .bmt-file. Attention: use only the temperature units GradC or GradF.
Name	testo_irimage_thermal_image_with_palette
Parameter	<code>int</code> id, <code>TESTO_IRAPI_UNIT</code> unit, <code>int</code> length
Reference parameter:	<code>wchar_t*</code> path
Description:	Gets a thermal bitmap with scale from .bmt-file. Attention: use only the temperature units GradC or GradF.
Name	testo_irimage_visual_image
Parameter	<code>int</code> id, <code>int</code> length
Reference parameter:	<code>wchar_t*</code> path
Description:	Gets the real image from .bmt-file if existing.
Name	testo_irimage_get_measurement_range
Parameter	<code>int</code> id
Reference parameter:	<code>float*</code> min, <code>float*</code> max
Description:	Writes the measurement range information of the .bmt-file into given parameters.
Name	testo_irimage_get_temperature
Parameter	<code>int</code> x, <code>int</code> y
Reference parameter:	<code>float*</code> temperature
Description:	Writes the temperature value of a point of the .bmt-file into given parameter. The point of origin is on the upper left corner.

#### 5.3.4.1 Device information

Name	testo_irimage_get_devicename
Parameter:	<code>int</code> id, <code>int</code> length
Reference parameter:	<code>wchar_t*</code> text_buffer
Getter/Setter:	Getter
Description:	Writes the type of IR-Camera into given parameter.

**Possible Value:** Any camera  
e.g.: testo 882  
t890-2

**Name** testo\_irimage\_get\_fov (Field of View)  
**Parameter:** `int` id  
**Reference parameter:** `int*` fov  
**Getter/Setter:** Getter  
**Description:** Writes the view angle in angle degree into given parameter.  
E.g.: an angle of 42° indicates wide angle lens.

**Name** testo\_irimage\_get\_serialnumber  
**Parameter:** `int` id  
**Reference parameter:** `unsigned int*` serial  
**Getter/Setter:** Getter  
**Description:** Writes the serial number of IR-Camera into given parameter..

#### 5.3.4.2 Picture information

**Name** testo\_irimage\_get\_datetime  
**Parameter:** `int` id, `int` length  
**Reference parameter:** `int*` year ,`int*` month ,`int*` day ,`int*` hour ,`int*` minute ,`int*` second  
**Getter/Setter:** Getter  
**Description:** Writes the date and time of shooting the IR-Image into given parameter.

**Name** testo\_irimage\_get\_height  
**Parameter:** `int` id  
**Reference parameter:** `int*` height  
**Getter/Setter:** Getter  
**Description:** Writes the height of the picture in Pixel into given parameter.

**Name** testo\_irimage\_get\_width  
**Parameter:** `int` id  
**Reference parameter:** `int*` width  
**Getter/Setter:** Getter  
**Description:** Writes the width of the picture in Pixel into given parameter.

**Name** testo\_irimage\_get\_reflected\_temperature  
**Parameter:** `int` id  
**Reference parameter:** `double*` refl\_temperature  
**Getter/Setter:** Getter  
**Description:** Writes the reflected temperature value for an IR-Image into given parameter.  
This value has influence on the calculation of the temperatures.

**Name** testo\_irimage\_set\_reflected\_temperature  
**Parameter:** `int` id, `double` refl\_temperature  
**Getter/Setter:** Setter  
**Description:** Sets the reflected temperature value for an IR-Image.  
This value has influence on the calculation of the temperatures.

**Name** testo\_irimage\_get\_emissivity  
**Parameter:** `int` id  
**Reference parameter:** `double*` emissivity  
**Getter/Setter:** Getter  
**Description:** Writes the emissivity value for an IR-Image into given parameter.  
This value has influence on the calculation of the temperatures.

**Name** testo\_irimage\_set\_emissivity  
**Parameter:** `double` emissivityValue  
**Getter/Setter:** Setter

**Description:** Sets the emissivity value for an IR-Image.  
This value has influence on the calculation of the temperatures.

**Name** testo\_irimage\_get\_min\_scale  
**Parameter:** `int` id  
**Reference parameter:** `float*` minScale  
**Getter/Setter:** Getter  
**Description:** Writes the minimal scale temperature value into given parameter.  
This value stands for the lowest scale temperature.

**Name** testo\_irimage\_set\_min\_scale  
**Parameter:** `int` id, `float` minScale  
**Getter/Setter:** Setter  
**Description:** Sets the minimal scale temperature value. This value stands for the lowest scale temperature.

**Name** testo\_irimage\_get\_max\_scale  
**Parameter:** `int` id  
**Reference parameter:** `float*` maxScale  
**Getter/Setter:** Getter  
**Description:** Writes the maximal upper scale temperature value into given parameter. This value stands for the highest scale temperature.

**Name** testo\_irimage\_set\_max\_scale  
**Parameter:** `int` id, `float` maxScale  
**Getter/Setter:** Setter  
**Description:** Sets the maximal upper scale temperature value. This value stands for the highest scale temperature.

**Name** testo\_irimage\_get\_palette  
**Parameter:** `int` id  
**Reference parameter:** TESTO\_IRAPI\_PALETTE\* Palette  
**Getter/Setter:** Getter  
**Description:** Writes the color palette of the picture into given parameter.  
**Possible Value:** TESTO\_IRAPI\_PALETTE\_IRONBOW =0,  
TESTO\_IRAPI\_PALETTE\_RAINBOW =1,  
TESTO\_IRAPI\_PALETTE\_GREYSCALE =2,  
TESTO\_IRAPI\_PALETTE\_GREYSCALEINV =3,  
TESTO\_IRAPI\_PALETTE\_SEPIA =4,  
TESTO\_IRAPI\_PALETTE\_BLUERED =5,  
TESTO\_IRAPI\_PALETTE\_HOTCOLD =6,  
TESTO\_IRAPI\_PALETTE\_TESTO =7,  
TESTO\_IRAPI\_PALETTE\_DEWPOINT =8,  
TESTO\_IRAPI\_PALETTE\_HOCHTEMP =9,  
TESTO\_IRAPI\_PALETTE\_RAINBOWHC=10

**Name** testo\_irimage\_set\_palette  
**Parameter:** `int` id  
**Reference parameter:** TESTO\_IRAPI\_PALETTE\* Palette  
**Getter/Setter:** Setter  
**Description:** Sets the color palette of the picture.  
**Possible Value:** TESTO\_IRAPI\_PALETTE\_IRONBOW =0,  
TESTO\_IRAPI\_PALETTE\_RAINBOW =1,  
TESTO\_IRAPI\_PALETTE\_GREYSCALE =2,  
TESTO\_IRAPI\_PALETTE\_GREYSCALEINV =3,  
TESTO\_IRAPI\_PALETTE\_SEPIA =4,  
TESTO\_IRAPI\_PALETTE\_BLUERED =5,  
TESTO\_IRAPI\_PALETTE\_HOTCOLD =6,  
TESTO\_IRAPI\_PALETTE\_TESTO =7,  
TESTO\_IRAPI\_PALETTE\_DEWPOINT =8,  
TESTO\_IRAPI\_PALETTE\_HOCHTEMP =9,  
TESTO\_IRAPI\_PALETTE\_RAINBOWHC=10

### 5.3.4.3 Humidity

**Name** testo\_irimage\_get\_humidity  
**Parameter:** `int` id  
**Reference parameter:** `double*` humidity  
**Getter/Setter:** Getter  
**Description:** Writes the humidity in %rH into given parameter.

**Name** testo\_irimage\_set\_humidity  
**Parameter:** `int` id, `double` humidity  
**Getter/Setter:** Setter  
**Description:** Sets the humidity in %rH.

### 5.3.4.4 Highlighting

**Name** testo\_irimage\_limits\_applied  
**Parameter:** `int` id  
**Reference parameter:** `int*` uselimits  
**Getter/Setter:** Getter  
**Description:** Writes current state into given parameter.

**Name** testo\_irimage\_apply\_limits  
**Parameter:** `int` id, `int` uselimits  
**Getter/Setter:** Setter  
**Description:** Turns on or off if limits are used.

**Name** testo\_irimage\_iso\_applied  
**Parameter:** `int` id  
**Reference parameter:** `int*` useIsotherme  
**Getter/Setter:** Getter  
**Description:** Write current state into given parameter.

**Name** testo\_irimage\_apply\_iso  
**Parameter:** `int` id, `int` useIsotherme  
**Getter/Setter:** Setter  
**Description:** Turn on or off if isotherm highlighting is used.

<b>Name</b>	testo_irimage_get_lower_iso_temperature
<b>Parameter:</b>	<code>int</code> id
<b>Reference parameter:</b>	<code>float*</code> lowerIsoTemp
<b>Getter/Setter:</b>	Getter
<b>Description:</b>	Writes the lower isotherm temperature value into given parameter. This value stands for the lowest highlighted isotherm temperature.
<b>Name</b>	testo_irimage_get_lower_iso_temperature
<b>Parameter:</b>	<code>int</code> id, <code>float</code> lowerIsoTemp
<b>Getter/Setter:</b>	Setter
<b>Description:</b>	Sets the lower isotherm temperature value. This value stands for the lowest highlighted isotherm temperature.
<b>Name</b>	testo_irimage_get_upper_iso_temperature
<b>Parameter:</b>	<code>int</code> id
<b>Reference parameter:</b>	<code>float*</code> upperIsoTemp
<b>Getter/Setter:</b>	Getter
<b>Description:</b>	Writes the upper isotherm temperature value into given parameter. This value stands for the highest highlighted isotherm temperature.
<b>Name</b>	testo_irimage_set_upper_iso_temperature
<b>Parameter:</b>	<code>int</code> id, <code>float</code> upperIsoTemp
<b>Getter/Setter:</b>	Setter
<b>Description:</b>	Sets the upper isotherm temperature value. This value stands for the highest highlighted isotherm temperature.
<b>Name</b>	testo_irimage_get_lower_limit_temperature
<b>Parameter:</b>	<code>int</code> id
<b>Reference parameter:</b>	<code>float*</code> lowerLimitTemp
<b>Getter/Setter:</b>	Getter
<b>Description:</b>	Writes the lower limit temperature value into given parameter. This value stands for the lowest limit temperature.
<b>Name</b>	testo_irimage_set_lower_limit_temperature
<b>Parameter:</b>	<code>int</code> id, <code>float</code> lowerLimitTemp
<b>Getter/Setter:</b>	Setter
<b>Description:</b>	Sets the lower limit temperature value. This value stands for the lowest limit temperature.
<b>Name</b>	testo_irimage_get_upper_limit_temperature
<b>Parameter:</b>	<code>int</code> id
<b>Reference parameter:</b>	<code>float*</code> upperLimitTemp
<b>Getter/Setter:</b>	Getter
<b>Description:</b>	Writes the upper limit temperature value into given parameter. This value stands for the highest limit temperature.
<b>Name</b>	testo_irimage_set_upper_limit_temperature
<b>Parameter:</b>	<code>int</code> id, <code>float</code> upperLimitTemp
<b>Getter/Setter:</b>	Setter
<b>Description:</b>	Sets the upper limit temperature value. This value stands for the highest limit temperature.