IRB-File-IrbAcs-SDK
Version 2

Documentation

State: 20/12/2017

InfraTec GmbH
Infrarotsensorik und Messtechnik
Gostritzer Straße 61 – 63
01217 Dresden/GERMANY

Telefon +49 351 871-8615
Fax +49 351 871-8727
E-Mail service@InfraTec.de
www.InfraTec.eu

Table of contents

1 Introduction

The software implemented for infrared cameras saves ist data in indexed files, what allows a fast and comfortable access. To simplifying handling for external software a SDK was created, which carries out the direct access to IRB files. The name of this library is IRBACS_xx.DLL or in linux systems libirbacs_xx.so and it allows especially the access to thermographic relevant properties of the in IRB-files saved infrared images. The intention of the SDK consist in providing the thermographic data for customer applications. With that passed data a user can develop his own application environment. Feeding back the changed data in the InfraTec software environment is not a primary aim of this SDK.
The range of functionality comprises access to the temperature data, header structure und temperature relevant correction parameters. Changed frames can be saved as single frame irb-file and in a limited range are manipulation of header data possible.

The following operating systems are supported:

Windows beginning with XP -	32 bit:	irbacs_w32.dll
Windows beginning with 7 -	64 bit:	irbacs_w64.dll
Linux – 32 bit:		libirbacs_l32.so
Linux – 64 bit:		libirbacs_l64.so

The following restrictions apply :

- Windows 64 bit : Compressed IRB-files of VarioCAM HD can not decompressed by 64-bit version of irbacs.dll.
- Resolution enhanced frames are not calculated, only the original camera resolution is available.

For real time access to data of camera systems exist another SDK (IRBGRAB-SDK), what is actually only as a 32-bit-dll available.

2 Data structures

All data structures are without empty bytes for alignment (packed). Type Boolean is represented as byte.

2.1 IRB-header structures

This structure describes with its substructures image and temperature relevant informations of IRB-format.

```
TIRBgeomInfo = packed record
    pixelFormat     : Word;
    compression     : Word;
    imgWidth        : Word;
    mgHeight        : Word;
    upperLeftX      : Word;
    upperLeftY      : Word;
    firstValidX     : Word;
    lastValidX      : Word;
    firstValidY     : Word;
    lastValidY      : Word;
    position        : Single;
end;

TIRBobjPars = packed record
    emissivity      : Single;
    objDistance     : Single;
    ambTemp         : Single;
    absoConst       : Single;
    pathTemp        : Single;
    version         : LongInt;
end;

TIRBCalibPars1 = packed record
    cbData          : array[0..1567] of byte;
end;

TIRBImageInfo = packed record
    level           : Single;
    span            : Single;
    imgTime         : TDateTime;
    imgMilliTime    : Single;    // Milliseconds
    imgAccu         : Word;
    imageComment: array[0..79] of Char;
    zoom_hor        : Single;
    zoom_vert       : Single;
    imgMilliTimeEx : SmallInt;  // Milliseconds
end;
```

```
TIRBImageData1 = packed record
      geomInfo        : TIRBgeomInfo;
      objectPars      : TIRBobjPars;
      calibPars       : TIRBCalibPars1;
      imgInfo         : TIRBImageInfo;
end;

PIRBImageData1 = ^TIRBImageData1 ;
```

2.2 IRB-Calib structure

This structure contains data used by calibration:

```
TIRBCalibData = packed record
      Version         : Integer;   // Calibration algorithm, there are 8 - Single/float values in IRB-frame
                                   //                                  9 - model of VarioCAM 2 und ImageIR
      Count           : Integer;   // Count of values (version 8 and 9)
      Values          : array [0..269] of Single; // Calibration values
end;
```

If calibration version 8 or 9 is given back, then every array value refers to a temperature value.

2.3 Correction data structure

This structure contains correction data.

```
TIRBCorrPars = packed record
      epsilon  : Double;
      envTemp  : Double;
      tau      : Double;
      pathTemp  : Double;
      lambda   : Double;
      deltaLambda : Double;
end;
```

3 List of DLL-Functions

The irbacs.dll is written with programming language Pascal, so the call convention is stdcall.

3.1 version

Description
Determine Dll-version of the library what can be used for an integrity check.

Function declaration
function version(var mainver, subver : Integer): Boolean;

mainver : Main version of the library
subver : Sub-version of the library

Return value
True : Success
False : Error

3.2 loadIRB

Description
Open an IRB file for processing and activates the first available frame.

Function declaration
function loadIRB(const fn : PAnsiChar) : PtrUInt;

fn : Full path name of IRB file

Return value
Value > 0 : Reference to internal management structure
Value = 0 : Error

3.3 unloadIRB

Description
Free the internal management structure in the Dll.

Function declaration
procedure unloadIRB(const aHandle : PtrUInt);

aHandle : With loadIRB created reference

Return value
No return value

3.4 getFrameCount

Description
Get count of frames in IRB-file.

Function declaration
function getFrameCount(const aHandle : PtrUInt): Integer;

aHandle : With loadIRB created reference

Return value
Value > 0 : Count of IR frames
Value = 0 : Error

3.5 getIRBIndices

Description
Determine indexes of thermo images in the IRB file, the so called IRB-indexes.

Function Declaration
function getIRBIndices(const aHandle : PtrUInt; const irbIdxList : PInteger): Integer;

aHandle : With loadIRB created reference
irbIdxLst : For getting the count of IRB-indexes in the IRB-file pass nil. To allocate appropriate memory it must have the size of index count * SizeOf(int32). If a pointer to that memory is passed, then function getIRBIndices puts all IRB- indexes in it.

Return value
Value > 0 : Count of IRB-indexes in the IRB-file
Value = 0 : Error

3.6 getFrameNumber

Description
Returns the IRB-index of activated thermo image.

Function declaration
function getFrameNumber(const aHandle : PtrUInt): Integer;

aHandle : With loadIRB created reference

Return value
Value > 0 : IRB-index of active frame
Value = 0 : Error

3.7 setFrameNumber

Description
Activate the thermo image according to the passed IRB-Index.

Function declaration
procedure setFrameNumber(const aHandle : PtrUInt; const frno : Integer);

aHandle : With loadIRB created reference
frno : IRB-Index of thermo image to activate

Return value
No return value

Remark:
Success can checked by by a call of getFrameNumber.

3.8 getFrameNbByArrayIdx

Description
Determines 0-based array index of activated thermo image.

Function declaration
function getFrameNbByArrayIdx(const aHandle : PtrUInt) : Integer;

aHandle : with loadIRB created reference

Return value
Value > (-1) : Array index of active thermo image (0-based)
Value = (-1) : Error

3.9 setFrameNbByArrayIdx

Description
Activation of thermo image with array index frno.

Function declaration
procedure setFrameNbByArrayIdx(const aHandle : PtrUInt; const frno : Integer);

aHandle : With loadIRB created reference
frno : Array index of thermo image to activate (0-based)

Return value
No return value

Remark
success of the action can checked by getFrameNbByArrayIdx

3.10 getTempBBXY

Description
Determine Blackbody-temperature at position (xx,yy) in thermal image

Function declaration
function getTempBBXY(const aHandle : PtrUInt; const xx, yy : Integer) : Double;

aHandle : With loadIRB created reference
xx : Column position of measurement point (0-based)
yy : Row position of measurement point (0-based)

Return value
Value > 0 : Temperature in Kelvin,
Value = 0 : Error

3.11 getTempXY

Description
Determine corrected temperature at position (xx,yy) in thermal image

Function declaration
function getTempXY(const aHandle : PtrUInt; const xx, yy : Integer) : Double;

aHandle : With loadIRB created reference
xx : Column position of measurement point (0-base)
yy : Row position of measurement point (0-based)

Return value
Value > 0 : Temperature in Kelvin
Value = 0 : Error

3.12 getDigValXY

Description
Determine digital values at position (xx,yy) of a thermal image.

Function declaration
function getDigValXY(const aHandle : PtrUInt; const xx, yy : Integer) : Double;

aHandle : With loadIRB created reference
xx : Column position of measurement point (0-based)
yy : Row position of measurement point (0-based)

Return value
Value > 0 : Temperature in Kelvin
Value = 0 : Error

3.13 readPixelData

Description

Get the whole thermal image as a data array of digital values, black body temperatures or corrected temperatures.

Function declaration

function readPixelData(const aHandle: PtrUInt; const pData: Pointer; const what: Integer) : Integer;

aHandle :	With loadIRB created reference
pData :	Pointer to a buffer allocated by the caller. Pass nil to get the needed memory size.
what :	kind of array data 0 – black body temperatures in K (data type Double) 1 – corrected temperatures in K (data type Double) 2 – digital values (data type Double) 3 – black body temperatures in K (Data type Single)

Return value

Value > 0 :	Buffer size (byte)
Value = 0 :	Error

3.14 readIRBData

Description

Read the activated raw data thermo image, what can be compressed, incl. header.

Function declaration

function readIRBData(const aHandle: PtrUInt; const pIRBFrame: PIRBImageData1) : Integer;

aHandle :	With loadIRB created reference
pIRBFrame :	Pointer to a buffer allocated by the caller. Pass nil to get the memory size to allocate.

Return value

Value > 0 :	Buffer size (byte)
Value = 0 :	Error

3.15 readIRBDataUncompressed

Description

Read the activated uncompressed raw data thermo image incl. header

Function declaration

function readIRBDataUncompressed(const aHandle : PtrUInt; const pIRBFrame: PIRBImageData1) : Integer;

aHandle :	With loadIRB created reference
pIRBFrame :	Pointer to a buffer allocated by the caller. Pass nil to get the memory size to allocate.

Return value

value > 0 : buffer size (byte)
value = 0 : error

3.16 convertPixelToKelvin

Description

Converts raw values (digital values) to temperatures by using the internal calibration and the passed correction parameters.

Function declaration

function(const aHandle: PtrUInt; const pData: Pointer; const cnt: Integer; const corrpars: Pointer): Integer;

aHandle : With loadIRB created reference
pData : Source/destination pointer to a buffer allocated by the caller
cnt : Buffer size in Byte (implicating count of data of type Double to convert)
Corrpars : Structure with correction parameters (see above TIRBCorrPars). If nil is passed the internal correction parameters are used.

Return value

value > 0 : Count of converted values
value = 0 : Error

Example:

```
// Load IRB-file
FIrbacsHnd := _irbacs64_loadIRB( filename );

// get the needed memory size
iMemorySize := _irbacs64_readPixelData( FIrbacsHnd, nil, 2 );
Getmem( kelvdat, iMemorySize );

// read raw values of pixels
iMemorySize := _irbacs64_readPixelData( FIrbacsHnd, kelvdat, 2 );

// example manipulation of raw values (values are of type Double)
pDoub := PDouble( kelvdat );
iPixelsize := iMemorySize div SizeOf( Double );
For ii := 0 to iPixelsize -1 do
begin
  pDoub^ := pDoub^ - 10;
  Inc( pDoub);
end;

// set correction parameters
Corrpars.epsilon      := 0.95;
Corrpars.envtemp      := 293.15;     // in K
Corrpars.tau          := 0.8;
Corrpars.pathtemp     := 283.15;     // in K
Corrpars.lambda       := 0;          // 0 -> value of calibration should be used
Corrpars.deltaLambda  := 0;          // 0 -> value of calibration should be used

Res := _irbacs64_convertPixelToKelvin( FIrbacsHnd, kelvdat, iPixelSize,
                          @corrpars );
```

3.17 getParam

Description

Read a parameter value of type Single (float) of a thermal image.

Function declaration

function getParam(const aHandle: PtrUInt; const what: Integer; var aValue: Double): Boolean;

aHandle : With loadIRB created reference

what : Kind of data to read

0 – Image width
1 – Image height
2 – Temperature level in K
3 – Temperature span in K
4 – Emissivity
5 – Distance
6 – Path temperature
7 – Environment temperature
8 – Absorption
9 – IRB-version (100 or 101)
10 – Zoom (1 = 100%)
14 – PixelFormat :
    1 = Byte (8 Bit unsigned)
    2 = Word / (16 Bit, unsigned)
    61572 (0xF084) = Single (float) (32 Bit floating point signed)
15 – Transmission (tau)
16 – Centroid wavelength Lambda
17 – Delta Lambda
18 – Timestamp with milliseconds relative to start of acquisition
19 – timestamp with absolut time (days since 30.12.1899 12.00 , 1 = 1 day)
20 – Lower limit of calibration range in K
21 – Upper limit of calibration range in K
22 – Trigger state
23 – Integration time in µsec
24 – HFOV in °
25 – VFOV in °
26 – Smooth
27 – Camera temperature in K
28 – Sensor temperature in K
29 – Process interface digital values (optional)
30 – Process interface analog value 1 (optional)
31 – Process interface analog value 2 (reserviert)
32 – Process interface analog value 3 (reserviert)
33 – Process interface analog value 4 (reserviert)
34 – Process interface analog value 5 (reserviert)

aValue : Returned parameter value

Return value

True : Success, aValue is valid
False : Error

3.18 setParam

Description

Set a parameter of type Single (float) of a thermal image. If the IRB-frame is saved with saveSingleFrame() these changes will be saved too.

Function declaration

function setParam(const aHandle: PtrUInt; const what: Integer; const aValue: Double) : Boolean;

aHandle : With loadIRB created reference

what : see getParam
aValue : Parameter value to set

Return value

True : Success, aValue is valid
False : Error

3.19 getParamS

Description

Read a parameter of type PAnsiChar of a thermal image.

Function declaration

function getParamS(const aHandle: PtrUInt; const what: Integer; const aValue: PAnsiChar) : Boolean;

aHandle : With loadIRB created reference
what : Kind of data to read
11 – Camera name
12 - Camera serial number
13 – Objektive name

aValue : Pointer to a buffer with minimal 256 byte (use constant cMaxParamString). Maximal 256 bytes are filled with a zero terminated string.

Return value

True : Success, aValue is valid
False : Error

3.20 getMilliTime

Description

Read millisecond-timestamp of a thermal image.

Function declaration
function getMilliTime(const aHandle: PtrUInt) : Double;

aHandle : With loadIRB created reference

Return value
Value > 0 : High resolution timestamp
value = 0 : Error

3.21 getFrameTimeStamp

Description
Read date and time of a thermal image.

Function declaration
function getFrameTimeStamp(const aHandle : PtrUInt; var timestamp: TSystemTime):Boolean;

aHandle : With loadIRB created reference
timestamp : TSystemTime-record with the following structure:

```
type TSystemTime = record
  Year: Word;
  Month: Word;
  DayOfWeek: Word;
  Day: Word;
  Hour: Word;
  Minute: Word;
  Second: Word;
  MilliSecond: Word;
end;
```

Return value
Value > 0 : Success, returned timestamp value is valid
Value = 0 : Error

3.22 getIRBCalibData*

Description
Read calibration data of a thermal image.

Function declaration
function getIRBCalibData(const aHandle: PtrUInt; var aIRBCalibData: TIRBCalibData) : Boolean;

aHandle : With loadIRB created reference
aIRBCalibData: Calibration data (For description of structure TIRBCalibData please see 2.2 IRB-Calib structure)

Return value
True : Success, IRBCalibData is valid
False : Error

**This function is only for compatibility with older version 1 SDK contained in this interface. There is no longer need for its functionality by function convertPixelToKelvin.*

3.23 getIRBHeader*

Description
Read IRB-header of a thermal image.

Function declaration
function getIRBHeader(const aHandle: PtrUInt; var aIRBHeader: TIRBImageData1) : Boolean;

aHandle : With loadIRB created reference
aIRBHeader : IRB-header data (For description of structure of TIRBImageData1 please see 2.1 IRB-header structures)

Return value
True : Success, aIRBHeader is valid
False : Error

**This function is only for compatibility with older version 1 SDK contained in this interface. There is no longer need for its functionality by function convertPixelToKelvin.*

3.24 setIRBHeader*

Description
Write IRB-header back to thermal image.

Function declaration
function setIRBHeader(const aHandle: PtrUInt; const aIRBHeader: TIRBImageData1) : Boolean;

aHandle : With loadIRB created reference
aIRBHeader : IRB-header data (For description of structure of TIRBImageData1 please see 2.1 IRB-header structures)

Return value
True : Success
False : Error

**This function is only for compatibility with older version 1 SDK contained in this interface. There is no longer need for its functionality by function convertPixelToKelvin.*

3.25 saveSingleFrame

Description
Save an thermal image in a IRB-file with specified name.

Function declaration
function saveSingleFrame(const aHandle: PtrUInt; fn : PChar; pIRBFrame: PIRBImageData1) : Boolean;

aHandle : With loadIRB created reference
fn : Full name with path of IRB-file
pIRBFrame : Pointer to a buffer with IRB-frame, what consists of header plus pixel data (e.g. previously read by readIRBDataUncompressed)or alternatively nil. In this case the activated frame will be saved.

Return value
True : Success
False : Error

3.26 exportVisBitmap

Description
Save visual image of actual thermal image in a file of corresponding format.

Function declaration
function exportVisBitmap(const aHandle: PtrUInt; const fn: PAnsiChar) : Boolean;

aHandle : With loadIRB created reference
fn : Full name of bitmap-file to save

Return value

True : Success
False : Error

3.27 audioComment

Description

Save audio comment of actual thermal image as wav-file.

Function declaration

function audioComment(const aHandle: PtrUInt; const fn: PAnsiChar) : Boolean;

aHandle : With loadIRB created reference
fn : Full name of wav-file to save

Return value

True : Success
False : Error