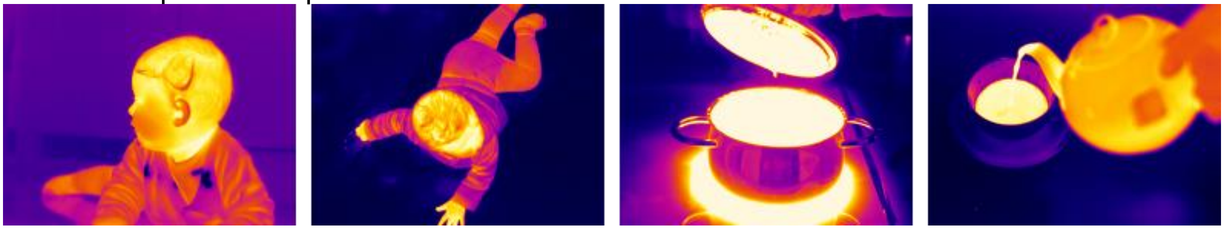
IRImagerDirect SDK - quick start

The TIM imager library is available as Debian package or Windows SDK. It contains a C-style interface and an object-oriented C++-interface for acquiring data from the thermoIMAGER TIM imager series.

thermoIMAGER TIM imager series

Thermal images can be acquired in raw format for further processing. In order to obtain a human-interpretable representation, the SDK supports several conversion functions. With different coloring palettes the user can get a false color representation of his data takes. Some examples are depicted below.



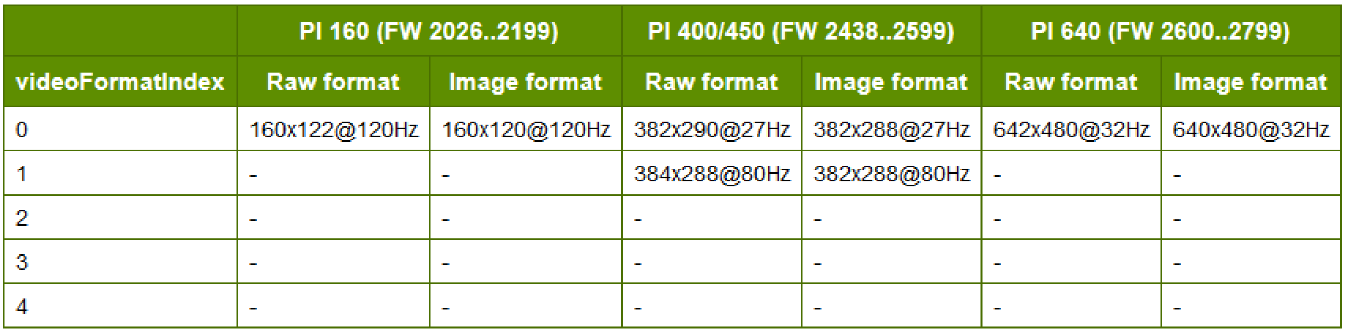
Thermal image samples from a household environment

Thermal image samples taken at a fire service drill

Video format

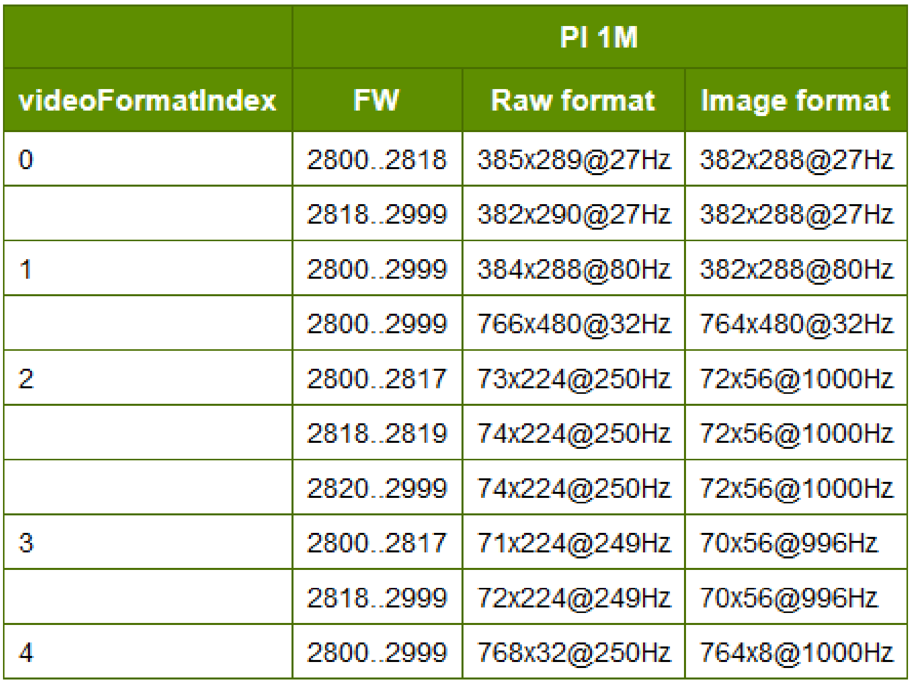
The TIM imagers provide different video formats. Depending on the camera model, the user can choose different resolutions and frame rates respectively by specifying the so called "videoFormatIndex" (see structure IRDeviceParams and section Generating configuration files).

For instance, the parameter chooses specific formats for the following devices:



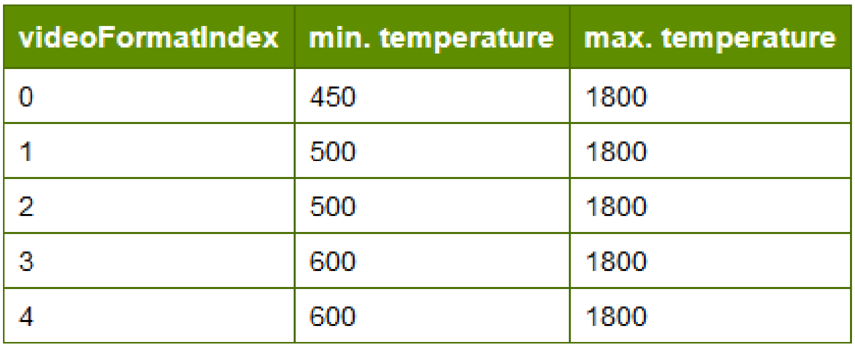
	PI 160 (FW 2026..2199)		PI 400/450 (FW 2438..2599)		PI 640 (FW 2600..2799)	
videoFormatIndex	Raw format	Image format	Raw format	Image format	Raw format	Image format
0	160x122@120Hz	160x120@120Hz	382x290@27Hz	382x288@27Hz	642x480@32Hz	640x480@32Hz
1	-	-	384x288@80Hz	382x288@80Hz	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-

For a TIM 1M device, the format maps as follows:



	PI 1M		
videoFormatIndex	FW	Raw format	Image format
0	2800..2818	385x289@27Hz	382x288@27Hz
	2818..2999	382x290@27Hz	382x288@27Hz
1	2800..2999	384x288@80Hz	382x288@80Hz
	2800..2999	766x480@32Hz	764x480@32Hz
2	2800..2817	73x224@250Hz	72x56@1000Hz
	2818..2819	74x224@250Hz	72x56@1000Hz
	2820..2999	74x224@250Hz	72x56@1000Hz
3	2800..2817	71x224@249Hz	70x56@996Hz
	2818..2999	72x224@249Hz	70x56@996Hz
4	2800..2999	768x32@250Hz	764x8@1000Hz

Important notice: The chosen temperature range is related to these image formats.



videoFormatIndex	min. temperature	max. temperature
0	450	1800
1	500	1800
2	500	1800
3	600	1800
4	600	1800

The user should check available video formats with a standard camera application, e.g., guvcview and valid temperature parameters by examining the provide calibration files.

Temperature data format

The callback methods are using an unsigned short data representation for temperatures. In order to get temperature values in floating point format, the following conversion needs to be applied:

```
void onThermalFrame(unsigned short* thermal, unsigned int w, unsigned int h, IRFrameMetadata meta, void* arg)
{
   ...
   float t = ((((float)data[i])-1000.f))/10.f;
   ...
}
```

Industrial Process Interface (PIF)

The Direct SDK supports the digital PIF input pin of ThermoIMAGER imagers. On raising edge two different functions can be used: *Image capture triggering* and *flag control*. The selection of one of both functions can be done via the xml configuration file:
<pifmode>0</pifmode> means to use image capturing.
<pifmode>1</pifmode> means to use flag control.

PIF snapshot triggering

In order to receive images at raising edge events of the PIF digital input, the user needs to register a callback function via the IRImager class instance:
void IRImager::setThermalFramePIFCallback(fptrIRThermalFrame callback)
for thermal images and
void IRImager::setVisibleFramePIFCallback(fptrIRVisibleFrame callback)
for RGB images of cameras supporting bispectral technology.
Alternatively, one can use the object oriented interface by deriving the class IRImagerClient. For more information see the provided examples.

PIF flag control

The automatic flag control can be deactivated (<autoflag> tag in the xml configuration file). Flag cycles can be triggered manually by the PIF digital input by setting the <pifmode> parameter in the xml configuration file (see description above).

Calibration/Configuration

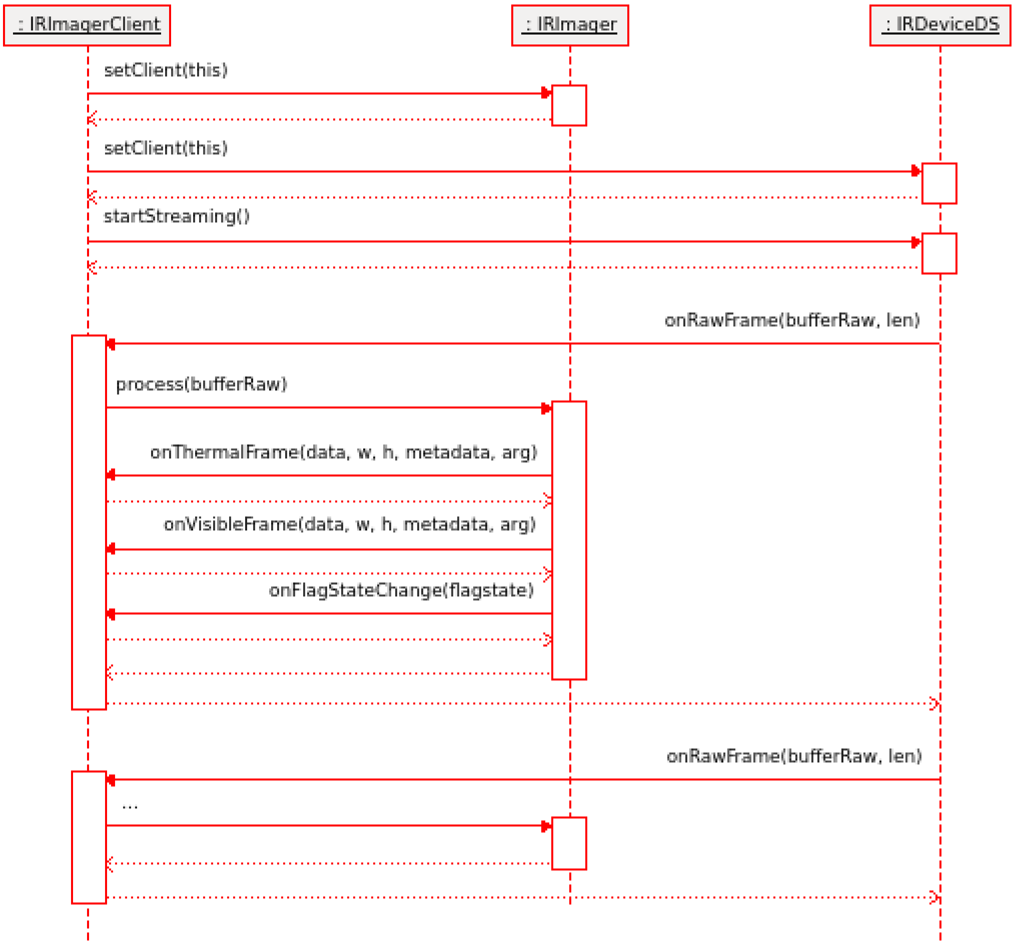
It is mandatory that the formats file (Formats.def) and the calibration files for the specific device can be found by the library. The thermoIMAGER TIM imagers are shipped with a data storage medium containing those files. Please copy them to the desired directory or import them by using the TIM connect software. The default folder for the calibration files is C:\Users\<myname>\AppData\Roaming\Imager. formats.def is found in the TIM Connect installation folder. A custom directory can be configured with the IRDeviceParams structure. The structure can also be populated with data through the IRDeviceParamsReader interface via an XML file. Add the following tags in the configuration XML file:
<calipath>C:\YOUR\SPECIFIC\PATH\TO\CALIBRATION\FILES</calipath>
<formatspath>C:\YOUR\SPECIFIC\PATH\TO\FORMATS\DEF</formatspath>

Creating configuration files

Please use the included generic configuration file generic.xml as a template. Parameter meanings are documented in the file itself.

Basic interface usage

Typically, a DirectShow graph is used to acquire raw data. The user can employ an own implementation or use the library-contained IRDeviceDS class. After calling the blocking run method, the class instance will call a callback function/method frequently as soon as data is available. This data is to be passed to the process method of an IRImager instance. It uses itself the callback functions/methods onThermalData, onVisibleFrame and onFlagStateChange to pass converted data to the user. The following sequence diagram documents this typical usage.


: IRImagerClient
: IRImager
: IRDeviceDS
setClient(this)
setClient(this)
startStreaming()
onRawFrame(bufferRaw, len)
process(bufferRaw)
onThermalFrame(data, w, h, metadata, arg)
onVisibleFrame(data, w, h, metadata, arg)
onFlagStateChange(flagstate)
onRawFrame(bufferRaw, len)
...

Typical usage of SDK on Windows platforms

A C-style example is also available. The main difference is the usage of callback functions, instead of callback class methods.

Troubleshooting

- Executing the examples via the console, requires to add the location of the libraries to the PATH-Variable, e.g. set PATH=%PATH%;..\..\sdk\x64, before executing ir_imager_show.exe ..\..\generic.xml.
- In order to see where the problem in starting an application is, increase the debug level, e.g.:
  IRLogger::setVerbosity(IRLOG_DEBUG, IRLOG_OFF);
  at the beginning of your application.

Frequently Asked Questions (FAQ)

Q: Can I convert RAVI files recorded with TIM-Connect with the Direct-SDK to different Formats?
A: No, RAVI files are only used by TIM -Connect. TIM -Connect can import *.raw data files recorded with the Direct-SDK, not vice versa.

Q: Can I import data from my Direct-SDK based application to TIM-Connect?
A: Yes, the class IRFileWriter supports a raw data format, that can be read by TIM-Connect. The example serializeRaw.cpp shows the usage of this class.

V. 1.0_05/18