Integration of the Flir A35sc Camera using MATLAB on 64Bit Windows

1. Softwaresetup

Optional:

To avoid problems with older versions of Pleora drivers, run the Pure Clean Tool from Flir Software Download section:

http://80.77.70.144/SwDownload/Assets/Thermovision/Pleora%20PureClean%201.0.0.1142.zip

Install the Pleora eBus SDK 2.2.3.2436

http://80.77.70.144/SwDownload/Assets/ThermoVision/pleora_ebus_sdk_2.2.3.2436.exe

To avoid Problems in data communication and stream, go to your network device in the HW-manager, right click, properties, register card advanced, setting “Jumbo Frames” to maximum value.

MATLAB Setup:

In this case, MATLAB 2014b with Image Acquisition Toolbox is used.

Go to 

at the register HOME>Get Hardware Support Packages > Download from Internet > GigE Vision as well as GenICam

imaqhwinfo shows the available adaptors. There should be an Adaptor called ‚gige‘.

2. Image Acquisition in MATLAB

Further information are found in the Image Acquisition Documentation:

http://de.mathworks.com/help/imaq/acquire-images-from-gige-vision-cameras.html?refresh=true

As shown in the documentation, **DONT** use a *gigecam* object.

- You can't access the 16-Bit Data as the *preview* function maps to 8-Bit.
- You can use the *snapshot* function in a loop witch won't provide Frames in a constant framerate – the framerate depends on MATLAB performance on windows

To access the full Bit depth data, use a *videoinput* object:

```
v = videoinput('gige', 1);
```

The image acquisition toolbox uses some GenICam standard functions/registers/values internally. They are integrated into the functions provided for the *videoinput* object.

For the function overview, see the functions of the following link:

http://de.mathworks.com/help/imaq/acquisition-using-any-hardware.html

To acquire a certain amount of frames from a GigE/GenICam Device, usually you would set up the "*AcquisitionMode*" to "*MultiFrame*", define an "*AcquisitionFrameNumber*" and use the "*AcquisitionStart*" to trigger the stream/block of frames.

In Acquisition Toolbox, you use the registers you find by having a look at the camera registers

```
s = v.Source
```

witch **differ** from registers available in the device XML.

For example, I mapped the Digital Input GPI to behave as a hardware trigger with

```
set(s,'PLC_Q14_Variable0','PLC_I0');
set(s,'GrbCh0TrigCfgPLCTriggerable','true');
```

See as well the Documentation for Understanding the Setup of the logic controllers:

http://www.gevicam.com/images/iPORT.Reference.Programmable_Logic_Controller.pdf

Furthermore, you can define your trigger using three different Types*, 'immediate'* for starting the acquisition directly when starting (*start*(obj))the videostream, '*manual'* for starting the acquisition with the *trigger*(obj) function any time after the stream started, and '*hardware'* for using signals at the device inputs.

In my example:

```
triggerconfig(v, 'Hardware');
```

To use the "*AcquisitionFrameNumber*" function, IAT provides the following attributes:

```
samplenr=100;
v.FramesPerTrigger = samplenr;
v.TriggerRepeat = 0;
```

Acquisition now needs a *start*(obj), *trigger*(obj) (for *triggerconfig* '*manual'*) and *stop*(obj).

Data acquisition (fetch data from memory to workspace) works with *getdata* function, if you want to save the frames on disk, follow the description (http://de.mathworks.com/help/imaq/logging-image-data-to-disk.html)

The *peekdata* function fetches some data from memory without deleting it from memory.

The following code shows one frame every 0.1 seconds and saves the frame-block to '*frames'* and the SW timestamps to 'ts'. The HW-Timestamps are accessible in metadata.

```
%% Acquire images
% start videoinput and wait for acquisition to complete
start(v);

%% If triggerconfig(v,'manual'):
% pause(20);
% trigger(v);

%% Visualization
figure(1)
i=0;
while (i<samplenr)
% Visualisierung: peekdata holt die Daten aus dem memory, ohne sie dort zu
% löschen, hier 1 Bild
imagesc(peekdata(v,1));
caxis([49000 52000]);
pause(0.1);
% Steigt bis zur angegebenen Bildanzahl
i=v.FramesAcquired;
end
% waitDuration = 10;
% wait(v, waitDuration);getdata function

% get frames and relative timestamps
[frames, ts, metadata] = getdata(v, v.FramesPerTrigger);
```

In this case of hardware triggering, you don't need the *stop*(obj) function as the *FramesPerTrigger* attribute stops the object after all frames are acquired.