ADVACAM
Imaging the Unseen

Python API

Online version: https://wiki.advacam.cz/wiki/Python_API

Contents

Overview

The python API can be used:

- directly in the system console, using a general python interpret
- in the Pixet program integrated python console

The base is pypixet object. It has methods for initialize and deinitialize, can create the pixet object.
The pixet object have device list, can create device objects and allows access to global properties.
A device objects have methods for acquisions and allows access to device parameters.
The pypxproc object is intended for use to processing of a data.
The pygui object allows you to create your own graphical interface. It can be used only if a script is run from the Pixet program.

Introductory examples

Small code example for using in the system console or other third-party environment, with the Python 3.7.9 (With all auxilliary files in the directory with the script):

```
import pypixet

print("pixet core init...")
pypixet.start()
pixet=pypixet.pixet
devices = pixet.devicesByType(pixet.PX_DEVTYPE_TPX3)
dev = devices[0]
dev.setOperationMode(pixet.PX_TPX3_OPM_EVENT_ITOT)

print("dev.doSimpleAcquisition (3 frames @ 1 sec) - start")
rc = dev.doSimpleAcquisition(3, 1, pixet.PX_FTYPE_AUTODETECT, "example.png")
print("dev.doSimpleAcquisition - end:", rc, "(0 is OK)")

#pixet.exitPixet()  # using only for old pxcore versions up-to 1.8.3 - save settings,
correct stop devices and core exit - use both lines
pypixet.exit()     # some USB devices must be power cycled if it not used / important if
third-party debug environment used
```

Small code example for using in the system console or other third-party environment, with the Python 3.7.9 (With all auxilliary files in other directory, the Pixet directory for example):

```
PIXETDIR="C:\\Program Files\\PIXet Pro"
import os
os.chdir(PIXETDIR)
```

```
# Note: Alternative is using sys.path.append(PIXETDIR)
import pypixet

print("pixet core init...")
pypixet.start()
pixet=pypixet.pixet
devices = pixet.devicesByType(pixet.PX_DEVTYPE_TPX3)
dev = devices[0]
dev.setOperationMode(pixet.PX_TPX3_OPM_EVENT_ITOT)

print("dev.doSimpleAcquisition (3 frames @ 1 sec) - start")
rc = dev.doSimpleAcquisition(3, 1, pixet.PX_FTYPE_AUTODETECT, "c:\\test-
files\\example.png")
print("dev.doSimpleAcquisition - end:", rc, "(0 is OK)")

#pixet.exitPixet()  # using only for old pxcore versions up-to 1.8.3 - save settings,
correct stop devices and core exit - use both lines
pypixet.exit()     # some USB devices must be power cycled if it not used / important if
third-party debug environment used
```

Note: os.chdir cannot work with devices requiring extra files like as firmwares or helper libraries. Add path instead it.

Small code example for using in the Pixet python console with integrated Python:

```
# do not create the pypixet and pixet, they exist by default

devices = pixet.devicesByType(pixet.PX_DEVTYPE_TPX3)
dev = devices[0]
dev.setOperationMode(pixet.PX_TPX3_OPM_EVENT_ITOT)

print("dev.doSimpleAcquisition (3 frames @ 1 sec) - start")
rc = dev.doSimpleAcquisition(3, 1, pixet.PX_FTYPE_AUTODETECT, "example.png")
print("dev.doSimpleAcquisition - end: %i (0 is OK)" % rc)

# do not execute the pixet.exitPixet(), it will cause whole the Pixet program to exit
```

Small code example for using in or outside the Pixet python console

```
pixetPresent = False

try:
    devices = pixet.devices()
    pixetPresent = True
```


```
    print("Pixet core: present")
except:
    print("Pixet core: starting...")
    import pypixet
    pypixet.start()
    pixet=pypixet.pixet
    devices = pixet.devices()
    print("Done")

dev = devices[0]
print(f"dev.doSimpleAcquisition(test.png)...")
rc = dev.doSimpleAcquisition(1, 1, pixet.PX_FTYPE_AUTODETECT, "test.png")
print(f"dev.doSimpleAcquisition - end: {rc} (0 is OK)")
if rc!=0: print(dev.lastError())

if pixetPresent:
    print("Pixet core: remain")
else:
    print("Pixet core: exit...")
    #pixet.exitPixet() # up to API 1.8.3
    pypixet.exit()
    print("Done")
```

See more examples: #Examples

Requirements

The Pixet Python API can be used from the Python interpreter integrated in the Pixet program or from command line with external Python without the Pixet.

For starting from the Pixet Python scripting plugin are not need any special files.

If you want to run scripts without the Pixet, need additional files:

- API functions using of pypixet.pyd and pypxproc.pyd
- Python versions 3.12

Or if old pxcore versions up-to 1.8.3 used, need Python 2.7 to 3.8 on Linux, or to 3.7 on Windows or 3.10 for ARM.

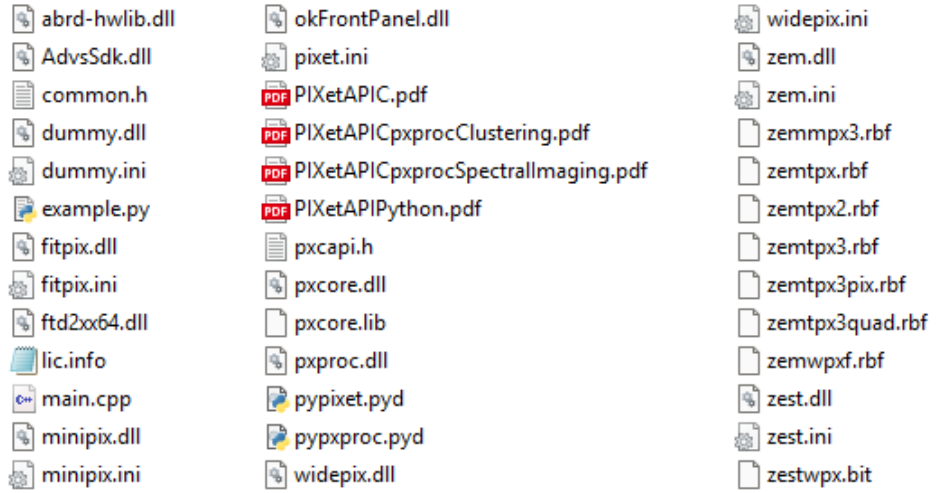
- For Windows the Pixet core dlls: pxcore.dll, pxproc.dll, or linux .so equivalents.
- For saving some advanced standard file types: Filetype plugin like as hdf5io.dll plugin, see: File types #HDF5, File types

#TIFF

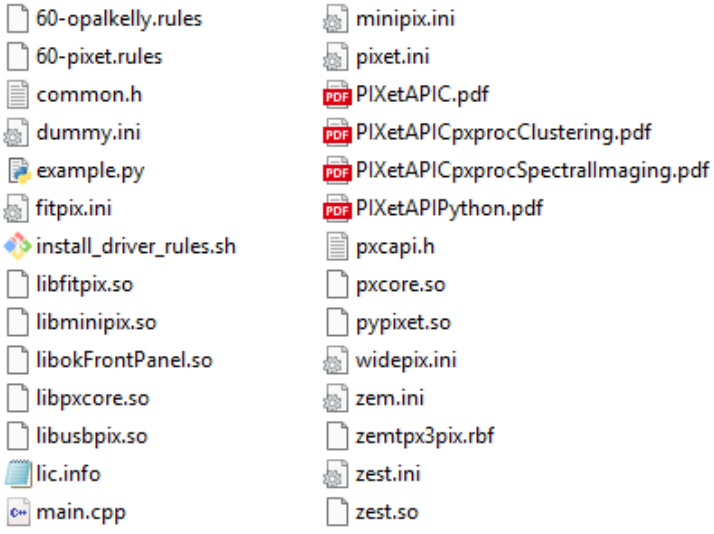
The Pixet core needs the pixet.ini file with proper hwlibs list inside, necessary hardware dll files (eq minipix.dll), subdirectory “configs” with config files for all present imaging chips (eq MiniPIX-I08-W0060.xml).

See: Files and directories of the Pixet and SDK

The Pixet API packages



Files in the Pixet API package for Windows



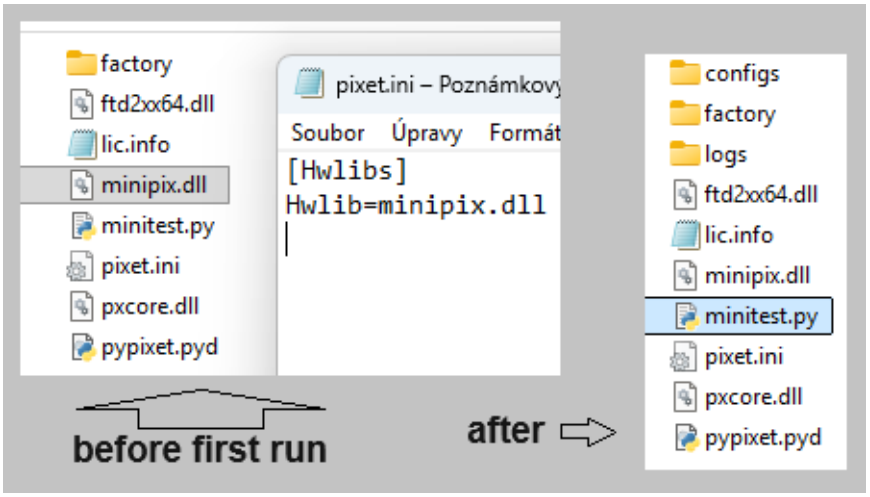
Files in Pixet API package for Linux

Download at advacam.com/downloads/
Note: Pay attention to download the correct environment version (OS, 32/64 bit, x86/ARM)

pxcore.dll (or lib...so) pxcore.lib pxproc.dll (or lib...so)	Binary libraries (pxcore.lib is only for Windows static linking)
pypixet.pyd (or .so) pypxproc.pyd (or .so)	Python core and processing libraries See Python API
pxcapi.h common.h	Header for the binary compile with pxcore.dll/so See Binary core API
pixet.ini	Pixet core configuration file See: Files and directories of the Pixet and SDK - pixet.ini file
main.cpp SampleProject.vcxproj SampleProject.sln	Sample Visual studio project (Windows package only)
*.rbf *.bit	Firmware files
install_driver_rules.sh 60-opalkelly.rules 60-pixet.rules	Linux drivers installer and it's helper files
okFrontPanel.dll libokFrontPanel.so	Helper library of the zem.dll/so hwlib
ftd2xx.dll	Helper library of the minipix.dll (Windows only)
other files .dll (or .so - no lib...so)	Hwlib files (can be in separate directory) See: Files and directories of the Pixet and SDK - hwlibs
lic.info	License file for the Pixet core - not importatnt in this version, but may be in future
pdf files	Binary and python APIs manuals

Pixet core on Windows need more Microsoft Visual Studio .NET standard dlls (vccorlib140.dll etc).

Project and auxilliary files examples

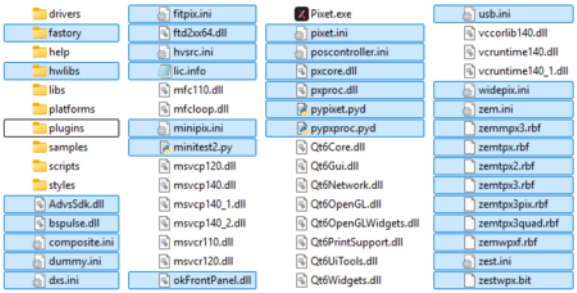


Files need for Python API program running without Pixet on a computer with MS Visual Studio installed
(This is afther 2025.09 / for old see: old version)

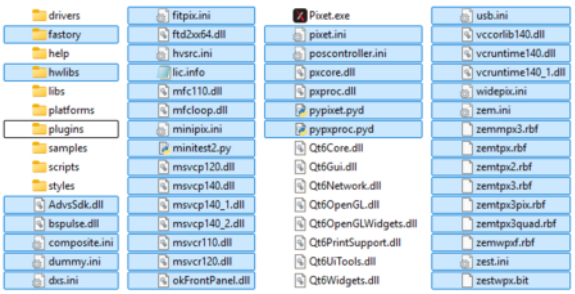
This is the directory and the pixet.ini file of the minimalistic python project. Using only with the Minipix device and can be run on a computer with properly installed MS Visual Studio or it's auxiliary files installed by another way. The "factory" directory contains the factory config file with proper name.

After the probram was started, pypixet.start() and pixet=pypixet.pixet, ... do something usesfull, ... pixet.exitPixet() and pypixet.exit() used, new directories, as visible on the right, will remain after the program.

If the MS VS auxiliary files are not accessible from the project directory, you can copy it from the Pixet program directory. In this image you can see all the auxiliary files marked, including MS VS dlls, firmware files for all our devices, hwlibs directories, ... Copying it to a python project should always make it ready to work, but not all are always needed.



Files need for PY-API, located in the Pixet directory - with VS installed on computer
(This is afther 2025.09 / for old see: old version)



Files need for PY-API, locatel in the Pixet directory - without VS installed
(This is afther 2025.09 / for old see: old version)

Note: Working with some file types, such as HDF5 or TIFF, also requires the corresponding plugin from the plugins directory, listed in the [plugins] section of the pixet.ini file.

Auxilliary files details

see:

- Small: Pixet SDK: Auxilliary files
- Details: Main directory of the API-using programs, independent on the Pixet
- Configs considerations (C): Binary core API #Device configuration

Basic troubleshooting

If normal error occured, like as function returns negative return code:

Use the device.lastError() to get error message and print or log it.

```
rc = dev.doSimpleAcquisition(5, 0.1, pixet.PX_FTYPE_AUTODETECT, "")
if rc!=0: print("doSimpleAcquisition", rc, dev.lastEroor())
```

Errors like as Object has no attribute 'setOperationMode' / 'doSimpleAcquisition' and other basic methods

Very likely that your version of Python is not compatible.
See: Typical complications

If not see anything, no devices detected, etc

See the Logs directory

If You want contact technical support

Send us Your program, contents of the "logs" directory, the return code, error message and what do You want to do with the detector device.

If a problem like as "DLL load failed" occured and you want contact technical support

Use the dir command and send us Your program, the error message, version of Python that is installed on the computer and the dir output.
Tip: Use DIR with subdirs, redirected to file and python version redirected to file, than send us the files:

```
dir /s > dir.txt
python -c "import sys; print(sys.version_info)" > pyVersion.txt
```

Note: If the program changing dirrectory, use cd "path" in the commandline. Copy the path from the program, type cd ", paste the copied path, type the second " and press enter (really copy the path by copy-paste, don't manual copy it). Then save the DIR again in the next file in the above way.

Typical complications

Python version

The API supports relatively old Pythons, moreover different ones depending on the platform. See #Requirements
Error message examples:

- Object has no attribute'doSimpleAcquisition'
- ImportError: DLL load failed while importing pypixet: *The specified module was not found.*

(Italic part is usually in the system language)

pypixet.pyd is not accessible

The python library is not in the directory that python currently sees.
May be related to pypixet.pyd, pypxproc.pyd, pypixetgui.pyd.
Error message example:

ModuleNotFoundError: No module named 'pypixet'

pxcore.dll is not accessible

The DLL library is not in the directory that python currently sees.
May be related to pxcore.dll, pxproc.dll
Error message example:

(No error message. The program ends after reach the import pypixet line.)

HW library not loaded

The import to Python was successful, but the program does not see any device, although it is connected and works in Pixet, for example.
Check if the library for your device is listed in pixet.ini in the [hwlibs] section and if its file is located in the specified location.
Check if all auxilliary files of your device is accessible.

Note: Some files, like as firmwares and helper libs, must be beside the program file or in added path, but not accessible using change directory.

Error message example:

(No error message. The program simply not see the device.)

The program works the first time and then doesn't work until the IDE is restarted.

Many programming environments (like us Spyder) allow variables and objects to be preserved after the program exits. If you do not exit your program with both pixet.exitPixet() and pypixet.exit() steps, there may be a problem with its restart.

The Pixet program normally working with the device, but my program cannot find device

If the Pixet found the device, but API running outside Pixet cannot find the device

1. pxcore in API was not load HW library, because it is not properly listed in it's pixet.ini
2. The HW library cannot access some auxilliary file, like as firmware image actual directory
3. The pxcore or HWlib has no some system/framework DLL files
4. You can copy Your program in the Pixet diractory and try to run it. If OK, find files You need or simple copy pixet.ini, all libs and the hwlibs directory.

   See: Files and directories of the Pixet and SDK

API version changes

Pxcore version 1.8.4 -> 1.8.5

- We changed supported python version from 3.7.x to 3.12.x
- Events changes:
  - Register: regID = dev.registerEvent(eventName, eventCallback)
  - Unregister: rc = dev.unRegisterEvent(regID)
  - Callback parameter:
    - Old: Single value, frame cout for example
    - New: Object with .data (some valua as was), .obj (calling object, device for example), .name (event name)
- Exit pixet core changes: Use only pypixet.exit()

First steps

See the Overview section for difference between run Py scripts from the Pixet embeded Python and run from a system console or third patrty IDE. This chapter contains a system console examples.

Mini test

Here we see the minimum necessary steps for the program to perform a measurement.

```
import pypixet
pypixet.start()
pixet=pypixet.pixet
devices = pixet.devices()
dev = devices[0]
dev.doSimpleAcquisition(1, 0.25, pixet.PX_FTYPE_AUTODETECT, "testFile.png")
```

- This program works, takes one measurement for 0.25 seconds and saves the result as an image testfile.png. However, it does not contain any detection and treatment of errors, setting the operating mode, properly terminating idle time with the device.
- It is possible that the program will measure something, but it is also possible that the resulting file will not appear. Or it is possible that the program will measure something once and it will be necessary to unplug and plug in the device before the next measurement.

Enhanced test

The improved program will make it possible to see whether initialization, measurement or termination is running, whether and what error occurred during measurement.

```
import pypixet
print("Pixet core starting...")
pypixet.start()
pixet=pypixet.pixet
devices = pixet.devices()
dev = devices[0]
print("doSimpleAcquisition...")
rc = dev.doSimpleAcquisition(1, 0.25, pixet.PX_FTYPE_AUTODETECT, "testFile.png")
print("doSimpleAcquisition rc:", rc, "(0 is OK)")
if rc!=0: print("  ", dev.lastError())
print("Exit pixet core...")
#pixet.exitPixet() # up to API 1.8.3
pypixet.exit()
```

- Most API functions have a return code. 0 means the operation was successful, negative values are error codes.
- If an error is detected, the last error message can be found using the .lastError() method.
- The functions pixet.exitPixet() and pypixet.exit() ensure that the work with the device is properly terminated.

We know what we are measuring

In this example, the operating mode is set before the measurement.

```
import pypixet
print("Pixet core starting...")
pypixet.start()
pixet=pypixet.pixet
devices = pixet.devices()
dev = devices[0]

rc = dev.setOperationMode(pixet.PX_TPX3_OPM_TOATOT)
print("setOperationMode rc:", rc, "(0 is OK)")
if rc!=0: print("  ", dev.lastError())
```

```
print("doSimpleAcquisition...")
rc = dev.doSimpleAcquisition(1, 0.25, pixet.PX_FTYPE_AUTODETECT, "testFile.png")
print("doSimpleAcquisition rc:", rc, "(0 is OK)")
if rc!=0: print("  ", dev.lastError())

print("Exit pixet core...")
#pixet.exitPixet() # up to API 1.8.3
pypixet.exit()
```

- Using setOperationMode, we set the operation mode to the desired value.
- If the command is not used, the device will measure according to what is stored in the configuration file since the last time.
- Note: Some devices hasn't the setOperationMode method. Use the pixel matrix configuration instead it (the Timepix chip, used in the Minipix-EDU device)

Is the device connected?

It is usually necessary to know whether the desired device is connected.

```
import pypixet
print("Pixet core starting...")
pypixet.start()
pixet=pypixet.pixet
#devices = pixet.devices()
devices = pixet.devicesTpx3() # detecting Timepix3 devices only.

devCnt = pixet.deviceCount()
if devCnt<1:
    print("No tpx3 devices detected")
    print("Exit pixet core...")
    pixet.exitPixet()
    pypixet.exit()
    exit()

dev = devices[0]

rc = dev.setOperationMode(pixet.PX_TPX3_OPM_TOATOT)
print("setOperationMode rc:", rc, "(0 is OK)")
if rc!=0: print("  ", dev.lastError())

print("doSimpleAcquisition...")
rc = dev.doSimpleAcquisition(1, 0.25, pixet.PX_FTYPE_AUTODETECT, "testFile.png")
print("doSimpleAcquisition rc:", rc, "(0 is OK)")
if rc!=0: print("  ", dev.lastError())
```

```
print("Exit pixet core...")
#pixet.exitPixet() # up to API 1.8.3
pypixet.exit()
```

There are several ways to get access to the device:

1. pixet.devices() This will create a list of devices of all types. If no physical device is detected, the virtual device named "FileDevice 0" is in this list.
2. devicesByType(type) List of devices of the type selected by the parameter.
3. devicesMpx2(), devicesMpx3(), devicesTpx3() List of devices of one type according to the function used.

- First way can detect all device types, but with default pixet.ini settings count of devs is never 0. To detect the "no device" state, test the condition devices[0].fullName()=="FileDevice 0".

```
devices = pixet.devices()
if devices[0].fullName()=="FileDevice 0":
    print("No devices connected")
    #pixet.exitPixet() # up to API 1.8.3
    pypixet.exit()
    exit()
dev0 = devices[0] # first of connected devices
```

- If an specialized dev list used, detect the "no device" state, by test the condition len(devices)==0.

```
devices = pixet.devicesTpx3()
if len(devices)==0:
    print("No Tpx3 devices connected")
    #pixet.exitPixet() # up to API 1.8.3
    pypixet.exit()
    exit()
dev0 = devices[0] # first of connected Tpx3 devices
```

List of the devices

Here, the display of the list of devices and their basic properties is added.

```
import pypixet
print("Pixet core starting...")
pypixet.start()
pixet=pypixet.pixet

devices = pixet.devicesTpx3() # detecting Timepix3 devices only.
```

```
devCnt = pixet.deviceCount()

if devCnt<1:
    print("No tpx3 devices detected")
    #pixet.exitPixet() # up to API 1.8.3
    pypixet.exit()
    exit()

for n in range(devCnt):
    dev = devices[n]
    print("  ", n, dev.fullName(), dev.width(), dev.height(), dev.chipCount(),
dev.chipIDs(), dev.sensorType(0))

# do something useful here

print("Exit pixet core...")
#pixet.exitPixet() # up to API 1.8.3
pypixet.exit()
```

Examples

Single examples

General (commandline)

For clarification on various auxiliary directories, configurations and calibrations:

https://advacam.com/examples/dirs+configs+calibs.py

All callbacks possible of Minipix-Tpx3 used:

https://advacam.com/examples/device-tpx3-frames-manyCallbacks.py

Reading a HDF5 (.H5) files:

https://advacam.com/examples/hdf5read.py

Library used in some frames-using examples:

https://advacam.com/examples/testFrames.py

Tpx (commandline)

Tpx version of device-tpx3-frames-manyAcqs.py:

https://advacam.com/examples/device-tpx-frames-manyAcqs.py

Tpx (run from Python IDE in the Pixet)

Example for MiniPIX-EDU in Pixet EDU, but relevant for other Tpx devs in the Pixet Pro

PIXet EDU: Operation modes and calibration

Tpx2 (commandline)

https://advacam.com/examples/device-tpx2-example.py
https://advacam.com/examples/device-tpx2-pixcfg.py

Tpx3 (commandline)

Acquisition frames on Tpx3 and subframes access:

Tpx3 Frames with subframes examle
(usually produces data in two subframes, and the data in the base frame are artifacts of raw frame processing)

Data-driven measuring and using callbacks:

https://advacam.com/examples/device-tpx3-dataDriven.py

Frame measuring by many ways with and without callbacks:

https://advacam.com/examples/device-tpx3-frames-manyAcqs.py
Requires: https://advacam.com/examples/testFrames.py

Mpx3 (commandline)

Mpx3 frames with subframes examle: https://advacam.com/examples/device-mpx3-frames-SFRs.py

Mpx3 synchronization in multidev example: https://advacam.com/examples/device-mpx3-frames-sync.py

Mpx3 integral measuring example: https://advacam.com/examples/device-mpx3-frames-integral.py

Pygui (run from Python IDE in the Pixet)

https://advacam.com/examples/pygui-GridLayout.py

https://advacam.com/examples/pygui-Plot.py

https://advacam.com/examples/pygui-Plot-caldata.py

https://advacam.com/examples/pygui-PropertyTreeView.py

https://advacam.com/examples/pygui-PropertyTreeView+MpxFramePanel.py

https://advacam.com/examples/pygui-MpxFrame-Tpx.py

https://advacam.com/examples/pygui-MpxFrame-Tpx3.py

https://advacam.com/examples/pygui-MpxFrame-Mpx3.py

https://advacam.com/examples/pygui-MpxFrame-Mpx3-multiDev.py

Examples packages

Pxproc (commandline)

Spectral imaging API examples: https://advacam.com/examples/API-Python-pxproc-spectraimg.rar

Clustering API examples: https://advacam.com/examples/API-Python-pxproc-clustering.rar

Auxilliary scripts

Single scripts

Single T3PA from Advapix-Quad to four "single-chip" files:

https://advacam.com/examples/multichip/t3pa-quad-to-4f.py

Converting T3PA to frrames TXT/PMF

https://advacam.com/examples/t3pa-to-frames.py

Can be used with or without measurement and measuring from system Python or Pixet Py console.

Export ABCT calibration files from multichip devs:

https://advacam.com/examples/multichip/device-multichip-export-abct.py

Spectral imaging offline parallel process of multichip data:

https://advacam.com/examples/multichip/pypxproc-spectraimg-tpx3-gFfE-parallel-offline.py

Documentation

The documentation of the Python API is in a PDF file located in the API package.
Download at: https://advacam.com/downloads/

Related

- Pixet SDK
- Files and directories: Main directory of the API-using programs