ADVACAM
Imaging the Unseen

Python API: Frames with subframes examle

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

Contents

3 in 1 examples for subframes 3

3 in 1 examples for subframes

Note: The example is designed for using from commandline, outside of the Pixet program.

```
# device/acquisitions in framemode: Frames with subframes
# (c) 2021 Pavel hudecek, Advacam
#
# This example have frame mode acquisitions:
# 1. use one frame with SFRs after acquisition
# 2. use multiple frames with SFRs after acquisitions sequence (with some processing tests)
# 3. use multiple frames with SFRs while acquisitions sequence is running
#
# Init the device, sensor refresh, acquisitions, using the frames
# Tested on MiniPix Tpx3 CdTe

import pypixet
import code, os, ctypes
from testFrames import *

try: # create dirrectory for flie output
    os.makedirs("out-files")
except FileExistsError:
    pass # already exists

print("pixet core init...")
pypixet.start()
pixet=pypixet.pixet

print("TPX3 devices:")
devices = pixet.devicesByType(pixet.PX_DEVTYPE_TPX3)
# devices = pixet.devices()

# if no physival devices present, virtual device "FileDevice 0" found
if len(devices)<1:
    print("  No devices connected")
    pixet.exitPixet()
    exit()
if devices[0].fullName()=="FileDevice 0":
    print("  No devices connected")
    pixet.exitPixet()
    exit()

devCnt = len(devices)
```

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

print("Selected device index: 0")
# get first Timepix3 device:
dev = devices[0]

parsNotSupp = False
pars = dev.parameters() # different devices has different sets of parameters
try:
    print("FirmwareCpu:", pars.get("FirmwareCpu").getString())
    print("Chip Temp:", pars.get("TemperatureChip").getDouble(), end='')
    print(", CPU Temp:", pars.get("TemperatureCpu").getDouble())
except:
    parsNotSupp = True

try:
    print("FirmwareCpu:", pars.get("FirmwareCpu").getString())
except:
    print()

#for p in pars.params():
#    print(p.name(), "-\t", p.description())

print("================================================================")

print("dev.loadConfigFromDevice", dev.loadConfigFromDevice())
# (Minipix, no Advapix)

opm = pixet.PX_TPX3_OPM_TOATOT
#opm = pixet.PX_TPX3_OPM_EVENT_ITOT # good for imaging / not usable for data-driven
#opm = pixet.PX_TPX3_OPM_TOA
#opm = pixet.PX_TPX3_OPM_TOT_NOTOA
print("mode set:", opm)
# set Timepix3 operation mode: (src_gen\ipixetw.cpp)
dev.setOperationMode(opm)

# Sensor refresh - sequential changing of bias
# First do the sensor refresh: Clean the chip for free charges.
# In data-driven/callbacks mode, some chips can sometimes stop producing data
#    in first measurement, if not refreshed before.
# Depending on the chip condition, default settings may work or not.
```

```
#    Refresh process can be programmed by device.setSensorRefresh("refresh string")
#    or possible to several repeat default refresh sequence.
# Alternatively can be used dummy measurement.
#print("doSensorRefresh start")
#dev.doSensorRefresh()
#print("doSensorRefresh end")

def acqSimple(time): #
================================================================================
    print("acqSimple", time, "-----------------")
    print("dev.doSimpleAcquisition - start")
    # count, time, fileType, fileName
    # count>1: frame data is available only from last frame
    rc = dev.doSimpleAcquisition(1, time, pixet.PX_FTYPE_AUTODETECT, "")
    print("dev.doSimpleAcquisition - end", rc)
    print("acqFrameCount", dev.acqFrameCount())
    #frame = dev.lastAcqFrameRefInc()
    frame = dev.acqFrameRefInc(0)
    print("  frame.frameCount:", frame.frameCount(), "subFrameCount",
frame.subFrameCount())
    data = frame.data() # medipix3 compatible data: simple list of pixel values
    # on Tpx3 in combined modes (OPM_EVENT_ITOT or OPM_TOATOT), there is only processing
artefacts
    # Proper data from comfined modes:
    # - is in the subframes (see SFRs example)
    # - can be saved by frame.save (save multiply files with sfr names in filename)
    print("  type data:", type(data), ", len:", len(data))
    frame.save("out-files\\fr.pmf", pixet.PX_FTYPE_AUTODETECT, 0)
    print("  Main frame preview:")
    testFrameUni(frame, "  ")
    if frame.subFrameCount()>0:
        f0=frame.subFrames()[0]
        print("  SubFrame0:")
        testFrameUni(f0, "  ")
        f0.save("out-files\\f0.pbf", pixet.PX_FTYPE_AUTODETECT, 0)
        #f0.destroy()
        if frame.subFrameCount()>1:
            f1=frame.subFrames()[1]
            print("  SubFrame1:")
            testFrameUni(f1, "  ")
            f1.save("out-files\\f1.pbf", pixet.PX_FTYPE_AUTODETECT, 0)
            #f1.destroy()
    frame.destroy()
    print("")
```

```
# def acqSimple

def testTheFrameSFR(frame, save): #
================================================================
    print("Frame idx:", n, "sfrs:", frame.subFrameCount(), " - - - - - - - - - - - - - - -
")
    frame.incRefCount()
    print("  dataFormat", frame.dataFormat(), "dataID", frame.dataID(), "isIgnoringMasked",
frame.isIgnoringMasked())
    if frame.subFrameCount()>0:
        for sfr in frame.subFrames():
            print("  Subframe:", sfr.frameName())
            print("    metaDataCount", frame.metaDataCount())
            for mdn in frame.metaDataNames():
                try:
                    print("    ", mdn, frame.metaData(mdn).data())
                except:
                    print("    (metadata view error)")
            print("    min", sfr.min(), 'minNonZero', sfr.minNonZero(), 'max', sfr.max(),
'mean', sfr.mean(), 'median', sfr.median(), 'sum', sfr.sum(), 'stdDev', sfr.stdDev())
            sfr.multiplyWithValue(1.123456)
            print("    dataFormat", sfr.dataFormat(), "dataID", sfr.dataID(),
"isIgnoringMasked", sfr.isIgnoringMasked())
            if sfr.frameName()=="ToT":
                dev.calibrateFrame(sfr)
                print("    min", sfr.min(), 'max', sfr.max(), 'mean', sfr.mean(), 'median',
sfr.median(), 'sum', sfr.sum())
            data = sfr.data()
            print("    data sample:")
            print("      ", end="");
            for i in range(256*120+100, 256*120+110):
                print(data[n], end=" ")
            print()
            sfr.save("out-files\\test-{}-sfr-".format(n)+sfr.frameName()+".png",
pixet.PX_FTYPE_AUTODETECT, 0)
            sfr.save("out-files\\test-{}-sfr-".format(n)+sfr.frameName()+".txt",
pixet.PX_FTYPE_AUTODETECT, 0)
            sfr.destroy()
    else:
        print("  min", frame.min(), 'minNonZero', frame.minNonZero(), 'max', frame.max(),
'mean', frame.mean(), 'median', frame.median(), 'sum', frame.sum())
        data = frame.data()
        print("  data sample:")
        for i in range(256*120+100, 256*120+110):
```

```
            print("   ", data[n], end="")
        print()
        frame.save("out-files\\test-{}-frame.png".format(n), pixet.PX_FTYPE_AUTODETECT, 0)
        frame.save("out-files\\test-{}-frame.txt".format(n), pixet.PX_FTYPE_AUTODETECT, 0)
#def testTheFrameSFR(frame):

def acqSimpleSFRs(count, time): #
=====================================================================
    print("acqSimpleSFRs", count, time, "-----------------")
    print("dev.doSimpleAcquisition - start")
    # count, time, fileType, fileName
    rc = dev.doSimpleAcquisition(count, time, pixet.PX_FTYPE_AUTODETECT, "")
    print("dev.doSimpleAcquisition - end", rc)
    print("acqFrameCount", dev.acqFrameCount())
    for n in range(dev.acqFrameCount()):
        frame = dev.acqFrameRefInc(n)
        testTheFrameSFR(frame, True)
        frame.destroy()
    # for n in range(dev.acqFrameCount())
    print("")
# def acqSimple2

# acqCallbackExample
=============================================================================

# Note: Parameter (val) in majority of callbacks is the finished measurement count.
# In the Acq Failed, (val) is error code form the Pixet core.

tics = ctypes.c_int64()
freq = ctypes.c_int64()

def callbackAMS(val): # Acq meas started (val) is meas. count. In this example is 0 (no
finished frames).
    #get ticks on the internal ~2-10 MHz QPC clock
    ctypes.windll.Kernel32.QueryPerformanceCounter(ctypes.byref(tics))
    #get the actual freq. of the internal ~2-10 MHz QPC clock
    ctypes.windll.Kernel32.QueryPerformanceFrequency(ctypes.byref(freq))
    # sys time code details can be found:
    #
https://stackoverflow.com/questions/38319606/how-can-i-get-millisecond-and-microsecond-reso
lution-timestamps-in-python
    t_s = tics.value/freq.value
    callbackAMS.secStart = t_s
    print("clb *** Acq meas started", val)
```


```
    print("  Sys time:", t_s, "sec, resolution:", 1e9/freq.value, "ns")
callbackAMS.secStart = 0

def callbackFin(val): # Acq finished (val) is number of finished measurements.
    ctypes.windll.Kernel32.QueryPerformanceCounter(ctypes.byref(tics))
    t_s = tics.value/freq.value
    print("clb *** Acq finished", val)
    print("  Sys time from start:", t_s - callbackAMS.secStart, "s")
    print("  Shutter start:", dev.getShutterStartTime(), "s, Shutter end:",
dev.getShutterEndTime())
    frame = dev.lastAcqFrameRefInc()
    testTheFrameSFR(frame, False)
    #print("  dev.calibrateFrame:", dev.calibrateFrame(frame))
    #print("  frame.dataID:", frame.dataID())
    #print("  frame.frameCount:", frame.frameCount())
    data = frame.data() # medipix3 compatible data: simple list of pixel values
    # on Tpx3 in combined modes (OPM_EVENT_ITOT or OPM_TOATOT), this is only first data
(EVENT or TOA)
    # This data is saved to simple image formats
    if val==1:
        print("  type frame:", type(frame))
        print("  frame.metaDataCount:", frame.metaDataCount())
        print("  frame.metaDataNames:", frame.metaDataNames())
        print("  type data:", type(data), ", len:", len(data))
    #print("  data min:", min(data), ", avg:", sum(data)/len(data), ", max:", max(data))
    fnam = "out-files\\frames-callback{}".format(val)
    print("  fnam:", "\""+fnam+"\"")
    frame.save(fnam+".png", pixet.PX_FTYPE_AUTODETECT, 0)
    frame.save(fnam+".txt", pixet.PX_FTYPE_AUTODETECT, 0)
    #testFrameUni(frame, "  ")
    frame.destroy()
# def callbackFin

def callbackFail(val): # Acq failed (val) is error code form the Pixet core.
    print("clb *** Acq failed", val)
    print("  ", dev.lastError())
def callbackDSCh (val): # Device status changed
    print("clb *** Dev status changed", val)
    print("  Status:", dev.deviceStatus())
    print("  StatusText:", dev.deviceStatusText())

def callbackACQ_SERIE_STARTED(val):
    print ("clb *** ACQ_SERIE_STARTED", val)
def callbackACQ_SERIE_FINISHED(val):
```

```
    print ("clb *** ACQ_SERIE_FINISHED", val)
def callbackACQ_MEAS_FINISHED(val):
    print ("clb *** ACQ_MEAS_FINISHED", val)
def callbackACQ_SWTRG_READY(val):
    print ("clb *** ACQ_SWTRG_READY", val)
def acqCallbackExample(count, time): #
============================================================
    print("acqCallbackExample", count, time, "-------------------------")
    print("registerEvent")
    # registerEvent(name, (reserved), CallFunc)
    # (only PX_EVENT_ACQ_FINISHED is necessary)
    dev.registerEvent(pixet.PX_EVENT_ACQ_MEAS_STARTED, 0, callbackAMS)
    dev.registerEvent(pixet.PX_EVENT_ACQ_FINISHED, 0, callbackFin)
    dev.registerEvent(pixet.PX_EVENT_ACQ_FAILED, 0, callbackFail)
    dev.registerEvent(pixet.PX_EVENT_DEV_STATUS_CHANGED, 0, callbackDSCh)

    #dev.registerEvent(pixet.PX_EVENT_, 0, callback)
    dev.registerEvent(pixet.PX_EVENT_ACQ_SERIE_STARTED , 0, callbackACQ_SERIE_STARTED)
    dev.registerEvent(pixet.PX_EVENT_ACQ_SERIE_FINISHED, 0, callbackACQ_SERIE_FINISHED)
    dev.registerEvent(pixet.PX_EVENT_ACQ_MEAS_FINISHED, 0, callbackACQ_MEAS_FINISHED)
    dev.registerEvent(pixet.PX_EVENT_ACQ_SWTRG_READY, 0, callbackACQ_SWTRG_READY)
    print("doAdvancedAcquisition start")
    #doAdvancedAcquisition(count, time, type, mode, fileType, fileFlags, filePath)
    rc = dev.doAdvancedAcquisition(count, time, pixet.PX_ACQTYPE_FRAMES,
pixet.PX_ACQMODE_NORMAL, pixet.PX_FTYPE_NONE, 0, "")
    print("doAdvancedAcquisition end", rc)
    #print("doSimpleAcquisition start")
    #dev.doSimpleAcquisition(count, time, pixet.PX_FTYPE_NONE,  "")
    #print("doSimpleAcquisition end")
    print("unregisterEvent")
    dev.unregisterEvent(pixet.PX_EVENT_ACQ_MEAS_STARTED, 0, callbackAMS)
    dev.unregisterEvent(pixet.PX_EVENT_ACQ_FINISHED, 0, callbackFin)
    dev.unregisterEvent(pixet.PX_EVENT_ACQ_FAILED, 0, callbackFail)
    dev.unregisterEvent(pixet.PX_EVENT_DEV_STATUS_CHANGED, 0, callbackDSCh)
    print()
# def acqCallbackExample


print("================================================================")
print("examples:") #
================================================================================
print()

testTime = 2
```

```
acqSimple(testTime)                     #(time)
acqSimpleSFRs(3, testTime)              #(count, time)
acqCallbackExample(5, testTime)         #(count, time)
print("All examples complete")
print("=====================================================================") #
======================

if not parsNotSupp:
    pars = dev.parameters()
    print("Chip Temp:", pars.get("TemperatureChip").getDouble(), end='')
    print(", CPU Temp:", pars.get("TemperatureCpu").getDouble())
print("pixet core exit...")
pixet.exitPixet()
pypixet.exit()
```