JUNE 4, 2022

FLI CAMERAS PROGRAMMING

PYTHON PROGRAMMING

PETER OLEYNIKOV

Conventions

The following conventions are used in this manual:

This icon denotes a note that contains an important information.

Bold — **Bold** text denotes the User Interface items in the software (clickable, for example, menu items, dialog box buttons etc.).

Italic — *Italic* text denotes emphasis, cross-references, or an introduction to a key concept. Italic text can also denote a text that that the user must enter.

Contents

Chapter 1 Python programming

The best way to control FLI Kepler cameras is to make calls to the FLI C# **FliSharp.DLL** library.

The **FliSharp.DLL** has dependencies on the following DLLs:

- log4net.dll
- System.ValueTuple.dll
- libflipro.x64.dll
- libflipro.x86.dll

These DLLs should be placed in a Windows folder that is visible to the system.

Chapter 2 ASCOM Python programming

To use the ASCOM FLI driver from the Python programming language the following libraries must be installed:

```
# for win32com.client do:
pip install pywin32
# if it did not work try:
pip install pypiwin32
```

In Python's scientific computing environment, efficient data structures for working with arrays are provided by the NumPy library and it is very convenient use it.

The imports that can be used are

```
import win32com.client              # needed to load COM objects
from win32com.universal import com_error     # for COM exceptions
import numpy as np
import matplotlib.pyplot as plt
import re
import time
```

The following constants can be defined for simplicity:

```
ASCOM_FLI_KEPLER_CAMERA = 'ASCOM.FLI.Kepler.Camera'
ASCOM_FLI_KEPLER_GET_MODE = 'GetMode'
ASCOM_FLI_KEPLER_SET_MODE = 'SetMode'
ASCOM_FLI_KEPLER_GET_MODESLIST = 'GetModesList'
ASCOM_FLI_KEPLER_GET_LOWGAIN = 'GetLowGain'
ASCOM_FLI_KEPLER_SET_LOWGAIN = 'SetLowGain'
ASCOM_FLI_KEPLER_GET_LOWGAINTABLE = 'GetLowGainTable'
ASCOM_FLI_KEPLER_GET_HIGHGAIN = 'GetHighGain'
ASCOM_FLI_KEPLER_SET_HIGHGAIN = 'SetHighGain'
ASCOM_FLI_KEPLER_GET_HIGHGAINTABLE = 'GetHighGainTable'
ASCOM_FLI_KEPLER_GET_BLACKLEVELLOW = 'GetBlackLevelLo'
ASCOM_FLI_KEPLER_GET_BLACKLEVELHIGH = 'GetBlackLevelHi'
ASCOM_FLI_KEPLER_GET_BLACKSUNLOW = 'GetBlackSunLo'
ASCOM_FLI_KEPLER_GET_BLACKSUNHIGH = 'GetBlackSunLo'
```

First, an ASCOM camera object must be created:

```
# get the camera object
cam = win32com.client.Dispatch(ASCOM_FLI_KEPLER_CAMERA)
if not cam.Connected:
    print('Connecting to the camera...')
    cam.Connected = True
    if not cam.Connected:
        print('Unable to connect to the camera')
        exit(-1)
```

At the program exit the camera **must** be disconnected:

```
print('Disconnecting from the camera...')
cam.Connected = False
```

If there is no camera connected, then the above script will produce the following output:

```
Connecting to the camera...
Unable to connect to the camera.
```

Once the current camera is connected the following script can be used to obtain the camera properties:

```
print('Connected to the camera!')
print('Description: ' + cam.Description)
print('DriverInfo: ' + cam.DriverInfo)
print('DriverVersion: ' + cam.DriverVersion)
print('InterfaceVersion: ' + str(cam.InterfaceVersion))
print('Name: ' + cam.Name)

print('CameraState: ' + str(cam.CameraState))
print('PixelSizeX: ' + str(cam.PixelSizeX))
print('PixelSizeY: ' + str(cam.PixelSizeY))
print('CameraXSize: ' + str(cam.CameraXSize))
print('CameraYSize: ' + str(cam.CameraYSize))
print('NumX: ' + str(cam.NumX))
print('NumY: ' + str(cam.NumY))
print('StartX: ' + str(cam.StartX))
print('StartY: ' + str(cam.StartY))
print('BinX: ' + str(cam.BinX))
print('BinY: ' + str(cam.BinY))
print('MaxBinX: ' + str(cam.MaxBinX))
print('MaxBinY: ' + str(cam.MaxBinX))
print('CCDTemperature: ' + str(cam.CCDTemperature))
print('HeatSinkTemperature: ' + str(cam.HeatSinkTemperature))

print('CanAbortExposure: ' + str(cam.CanAbortExposure))
print('CanAsymmetricBin: ' + str(cam.CanAsymmetricBin))
print('CanGetCoolerPower: ' + str(cam.CanGetCoolerPower))
print('CanPulseGuide: ' + str(cam.CanPulseGuide))
print('CanSetCCDTemperature: ' + str(cam.CanSetCCDTemperature))
print('CanStopExposure: ' + str(cam.CanStopExposure))
print('CanFastReadout: ' + str(cam.CanFastReadout))
print('CoolerOn: ' + str(cam.CoolerOn))
print('CoolerPower: ' + str(cam.CoolerPower))
print('HasShutter: ' + str(cam.HasShutter))
print('ExposureMax: ' + str(cam.ExposureMax))
print('ExposureMin: ' + str(cam.ExposureMin))
```

The above code produces the following output:

```
Connecting to the camera...
Connected to the camera!
Description: ASCOM Camera Driver for FLI Kepler
DriverInfo: ASCOM Camera Driver for FLI Kepler. Version: 6.2
DriverVersion: 6.2
InterfaceVersion: 2
Name: FliPro
CameraState: 0
PixelSizeX: 11.0
PixelSizeY: 11.0
CameraXSize: 2048
CameraYSize: 2048
NumX: 2048
NumY: 2048
StartX: 0
StartY: 0
BinX: 1
BinY: 1
MaxBinX: 1
MaxBinY: 1
```

```
CCDTemperature: 20.0
HeatSinkTemperature: -59.9375
CanAbortExposure: True
CanAsymmetricBin: False
CanGetCoolerPower: True
CanPulseGuide: False
CanSetCCDTemperature: True
CanStopExposure: True
CanFastReadout: True
CoolerOn: False
CoolerPower: 16.0
HasShutter: True
ExposureMax: 3600.0
ExposureMin: 0.0
```

The camera readout modes can be obtained through the ASCOM **ReadoutModes** list:

```
try:
    # ASCOM only supports readout modes if CanFastReadout == False
    if not cam.CanFastReadout:
        print('ReadoutMode: ' + str(cam.ReadoutMode))
        print('ReadoutModes:')
        modes = cam.ReadoutModes
        for i in range(modes.Count):
            print('  #' + str(i) + ' ' + str(modes[i]))
except com_error as error:
    _, msg, exc, _ = error.args
    _, _, msg2, _, _, _ = exc
    print('Oops!  ' + msg + ' ' + msg2)
```

Some of the FLI Kepler camera parameters can be accessed through the ASCOM **SupportedActions** list.

```
actions = cam.SupportedActions
print('Actions: ' + str(actions.Count))
for i in range(actions.Count):
    print('  #' + str(i) + ' -> ' + actions[i])

# ASCOM function: string Action(string ActionName, string ActionParameters)
```

The above code produces the following output:

```
Actions: 20
  #0 -> SetShutter
  #1 -> SetMode
  #2 -> GetMode
  #3 -> GetModesList
  #4 -> GetLowGainTable
  #5 -> GetHighGainTable
  #6 -> SetLowGain
  #7 -> GetLowGain
  #8 -> SetHighGain
  #9 -> GetHighGain
  #10 -> SetBlackLevelLo
  #11 -> SetBlackLevelHi
  #12 -> GetBlackLevelLo
  #13 -> GetBlackLevelHi
  #14 -> SetBlackSunLo
  #15 -> SetBlackSunHi
  #16 -> GetBlackSunLo
  #17 -> GetBlackSunHi
  #18 -> SetFramePassChannel
  #19 -> GetFramePassChannel
```

For, example, the list of modes can be obtained by the following code:

```
modes = cam.Action(ASCOM_FLI_KEPLER_GET_MODESLIST, '')
print('Modes list: ')
print_table(modes, ', index=', ', name=')
print('Current Mode: ' + cam.Action(ASCOM_FLI_KEPLER_GET_MODE, ''))
print('Set Mode: res = ' + cam.Action(ASCOM_FLI_KEPLER_SET_MODE, str(0)))
# check the mode
print('Current Mode: ' + cam.Action(ASCOM_FLI_KEPLER_GET_MODE, ''))

# set to HDR mode
print('Set Mode: res = ' + cam.Action(ASCOM_FLI_KEPLER_SET_MODE, str(2)))
# check the mode again
print('Current Mode: ' + cam.Action(ASCOM_FLI_KEPLER_GET_MODE, ''))
```

with the following result:

```
Modes list:
  #0, index=0, name=Rolling LDR
  #1, index=1, name=Rolling LDR - LDC
  #2, index=2, name=Rolling HDR
  #3, index=3, name=Rolling HDR - LDC
```

For the camera gains:

```
# gains
low_gains = cam.Action(ASCOM_FLI_KEPLER_GET_LOWGAINTABLE, '')
high_gains = cam.Action(ASCOM_FLI_KEPLER_GET_HIGHGAINTABLE, '')
# print('Low Gain Table: ' + low_gains)
# print('High Gain Table: ' + high_gains)
# print tables
print('Low Gain Table:')
print_table(low_gains, ', index=', ', gain=')
print('High Gain Table:')
print_table(high_gains, ', index=', ', gain=')

try:
    # NOTE: in LDR modes settings High gain will throw an exception
    print('Low Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_LOWGAIN, ''))
    print('High Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_HIGHGAIN, ''))
    # set gains
    print('Low Gain: ' + cam.Action(ASCOM_FLI_KEPLER_SET_LOWGAIN, str(0)))
    print('High Gain: ' + cam.Action(ASCOM_FLI_KEPLER_SET_HIGHGAIN, str(0)))
    # check gains
    print('Low Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_LOWGAIN, ''))
    print('High Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_HIGHGAIN, ''))
    # set gains back
    print('Low Gain: ' + cam.Action(ASCOM_FLI_KEPLER_SET_LOWGAIN, str(2)))
    print('High Gain: ' + cam.Action(ASCOM_FLI_KEPLER_SET_HIGHGAIN, str(7)))
    # check gains
    print('Low Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_LOWGAIN, ''))
    print('High Gain: ' + cam.Action(ASCOM_FLI_KEPLER_GET_HIGHGAIN, ''))
except com_error as error:
    _, msg, exc, _ = error.args
    _, _, msg2, _, _, _ = exc
    print('Oops!  ' + msg + ' ' + msg2)
```

With the following results:

```
Low Gain Table:
  #0, index=0, gain=0.66
  #1, index=4, gain=1.29
  #2, index=1, gain=1.85
  #3, index=2, gain=2.49
```

```
  #4, index=3, gain=3.68
  #5, index=5, gain=3.7
  #6, index=6, gain=4.95
  #7, index=7, gain=7.25
High Gain Table:
  #0, index=0, gain=0.66
  #1, index=4, gain=1.29
  #2, index=1, gain=1.85
  #3, index=2, gain=2.49
  #4, index=3, gain=3.68
  #5, index=5, gain=3.7
  #6, index=6, gain=4.95
  #7, index=7, gain=7.25
```

A single frame can be obtained from the camera using the following code:

```
# set the full frame
cam.NumX = cam.CameraXSize
cam.NumY = cam.CameraYSize

# start 1.0s light frame exposure
print('Acquire 1s bright image')
cam.StartExposure(1.0, True)

# wait for the frame to be ready
while not cam.ImageReady:
    print('No image yet...')
    time.sleep(0.2)

# get the camera image
buffer = cam.ImageArray
# print the very first pixel
print('First pixel: ' + str(buffer[0][0]))
```

At this point it is convenient to convert the image data into the NumPy array:

```
image = np.array(buffer)
```

The we can save the above data (4-bit integer numbers) into a file:

```
# make a file
newFile = open("image2.bin", "wb")
# write to the file
newFile.write(image)
```

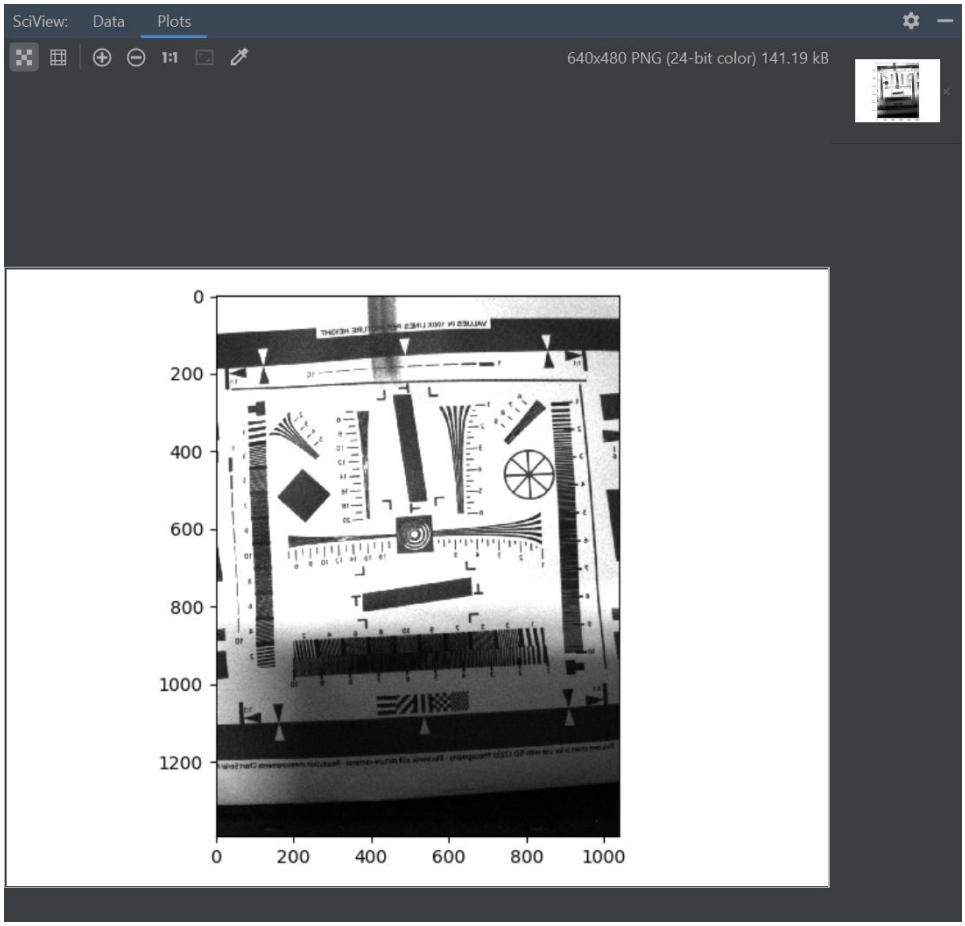
The NumPy array can be created as an unsigned 16-bit integer from the image buffer as:

```
image = np.array(buffer, dtype=np.uint16)
```

Alternatively, we can display the image:

```
plt.imshow(image, cmap='gray', vmin=0, vmax=4096)
plt.show()
```

In the **JetBrains © PyCharm** Python programming environment we can see the image directly in the **SciView** tab:



Actions description

The following table describes all the names and parameters that ASCOM Action function accepts as arguments:

Action name	Action parameter	Description
SetShutter	'open' / 'close'	Opens/Closes the shutter
SetMode	0 … N–1	Sets the mode (N – number of modes in the table returned by **GetModesList**)
GetMode		Gets the current mode
GetModesList		Gets the modes list as a multiline string. Each line has the format of: *<mode name>*;*<mode index>*
GetLowGainTable		Gets the modes list as a multiline string. Each line has the format of: *<gain value>*;*<gain index>*
GetHighGainTable		Same as Low gain table
SetLowGain	0 … N–1	Sets low gain index (N – number of gains in the table returned by **GetLowGainTable**)
GetLowGain		Gets low gain index
SetHighGain	0 … N–1	Sets high gain index (N – number of gains in the table returned by **GetHighGainTable**)
GetHighGain		Gets high gain index
SetBlackLevelLo	0 … 16383	Set the low Black Level
SetBlackLevelHi	0 … 16383	Set the high Black Level
GetBlackLevelLo		Get the low Black Level
GetBlackLevelHi		Get the high Black Level
SetBlackSunLo	0 … 63	Set the low Black Sun
SetBlackSunHi	0 … 63	Set the high Black Sun
GetBlackSunLo		Get the low Black Sun
GetBlackSunHi		Get the high Black Sun
SetFramePassChannel	0 – low, 1 – high, 2 – merged	Set the frame pass channel type
GetFramePassChannel		Get the frame pass channel type