Guide to the flap_apdcam module

S. Zoletnik, Centre for Energy Research
zoletnik.sandor@ek-cer.hu

8 May, 2022

Contents

Introduction

The flap project is a programming environment in Python for handling and analysing large multidimensional datasets. It can be found on GitHub (https://github.com/fusion-flap) and consists of the core module and a set of data source modules. These data source modules contain the special knowledge on measurement channels, timing, coordinates, calibration, and other information related to a certain data source. They provide a standard interface for integrating into flap, through which data, coordinates and additional information are encapsulated in flap DataObjects. These can be processed in flap regardless of the source of the original data.

The flap_apdcam module implements a data source module for the ADPCAM family devices manufactured by Fusion Instruments. (www.fusioninstruments.com). Additionally to the flap data source, the module contains a graphical user interface (GUI) for performing the most common measurement tasks and another GUI for plotting raw signals and power spectra.

APDCAM data

APDCAM cameras are 32…128 channel fast detector devices capable of sampling all signals directly to computer memory up to several MHz sampling rate, 8…14 bits. Additionally, they have various triggering and timing possibilities. Two camera families area available:

- APDCAM-1G (or simply APDCAM) has 32 channels and transmits data through 1 Gbit Ethernet interface. It uses either a 4x8 pixel Avalanche PhotoDiode (APD) matrix or a same size Multi Pixel Photon Counter (MPPC) detector matrix. With all channels on it has maximum 2 MHz sample rate.
- APDCAM-10G can have 64 or 128 channels and transmits data over a 10 Gbit optical Ethernet interface. It has various pixel configurations: 4x32, 4x16, 8x8, 8x16 or even a 64 channel version with individual fibre coupled detectors. Depending on configuration its maximum sample rate is 5-10 MHz.

The APDCAM devices write data into binary files (2-byte little endian unsigned integer), one file per channel. The camera setup is written into an xml format file which is both human and machine readable. The collection of data files and the xml file are contained in one directory.

The flap_apdcam module reads the configuration and the data and generates a 1, 2 or 3 dimensional flap DataObject.

Using the flap_apdcam data source

To use the flap_apdcam module the following lines must be added to the user program

```
import flap
import flap_apdcam
flap_apdcam.register()
```

The first two lines are standard Python import statements. The third line registers the apdcam data source in flap. After this data can be read using the flap get_data method, e.g. like:

```
get_data(’APDCAM’,name=’APD-*’,coordinates={‘Time’:[1,2]},
         options={’Datapath’:’c\data\m12’}
         )
```

The above example reads all the signals from the measurement directory c:\data\m12 between measurement time 1-2 s. The data object will be either 2D (channel-time) or 3D (row-column-time) depending on the APDCAM configuration.

The flap_apdcam.register() call accepts a data_source keyword. This is used for registering the data source under a different name than APDCAM.

The **name** parameter accepts the following:

- A single string or a list of strings
- ‘APD-<channel>’ for cameras with individual detectors, where <channel> is the channel number (1…)
- ‘APD-<row>-<column>’ means a pixel in the 2D APD matrix. Pixel numbering starts on the upper left corner as looking on to the detectors from the front of the camera.
- ‘ADC-<channel>’ indicates an ADC channel in the camera. The pixel to ADC mapping is contained in the apdcam_channel_map.py and apdcam10_channel_map.py functions.
- <channel>, <row> and <column> can be either a number, * (meaning all values) or [<ch1>-<ch2>] meaning a range of channels

The ‘options’ parameter conforms to flap standards, it is a dictionary. The following keys are understood:

```
        'Scaling':  'Digit'
                    'Volt'
        'Datapath': Data path (string)
        'Camera type': string
                        None: Determine camera type from xml file
                        string: Set camera type as below.
                        The camera type:
                         APDCAM-10G:
                             APDCAM-10G_4x32
                             APDCAM-10G_8x8
```

```
                                APDCAM-10G_4x16
                                APDCAM-10G_8x16
                                APDCAM-10G_8x16A
                            APDCAM-1G:
                                APDCAM-1G : Standard APDCAM-1G (Horizontal array)
                                APDCAM-1G_90: APDCAM-1G with sensor rotated
                                              90 degree CCW
                                APDCAM-1G_180: APDCAM-1G with sensor rotated
                                               180 degree CCW
                                APDCAM-1G_270: APDCAM-1G with sensor rotated
                                               270 degree CCW
        'Camera_version': int
            Only applicable to 10G cameras: 0,1,2
```

APDCAM-10G cameras manufactured after 2016 all have version 1.

APDCAM programming interface

At present the programming interface is available only for the APDCAM-10 camera family. It is contained in file APDCAM10G_control in the apdcam_control directory of the module. The APDCAM10G_regCom class implements the camera interface. Some example programs are included in the module showing basic use cases.

For measurement the APDTest_10G scriptable console program is also needed. It is available only for Linux systems, as on Windows the 10G Ethernet interface throughput is limited. The program is part of the flap_apdcam package in directory apdacam_control/APDTest_10G. It must be compiled on the given Linux machine. To compile enter the directory and use:

```
make clean
make all
```

Nothing else needs to be done, the Python programs use the APDTest_10G program from the package.

The Graphical User Interface (GUI)

The graphical user interface is implemented in the tkinter Python package, the Python implementation of the Tcl/Tk standard.

The graphical user interface is present in the apdcam10g_control_gui.py file. It can be started in Python with the following commands:

```
import flap_apdcam
flap_apdcam.gui()
```

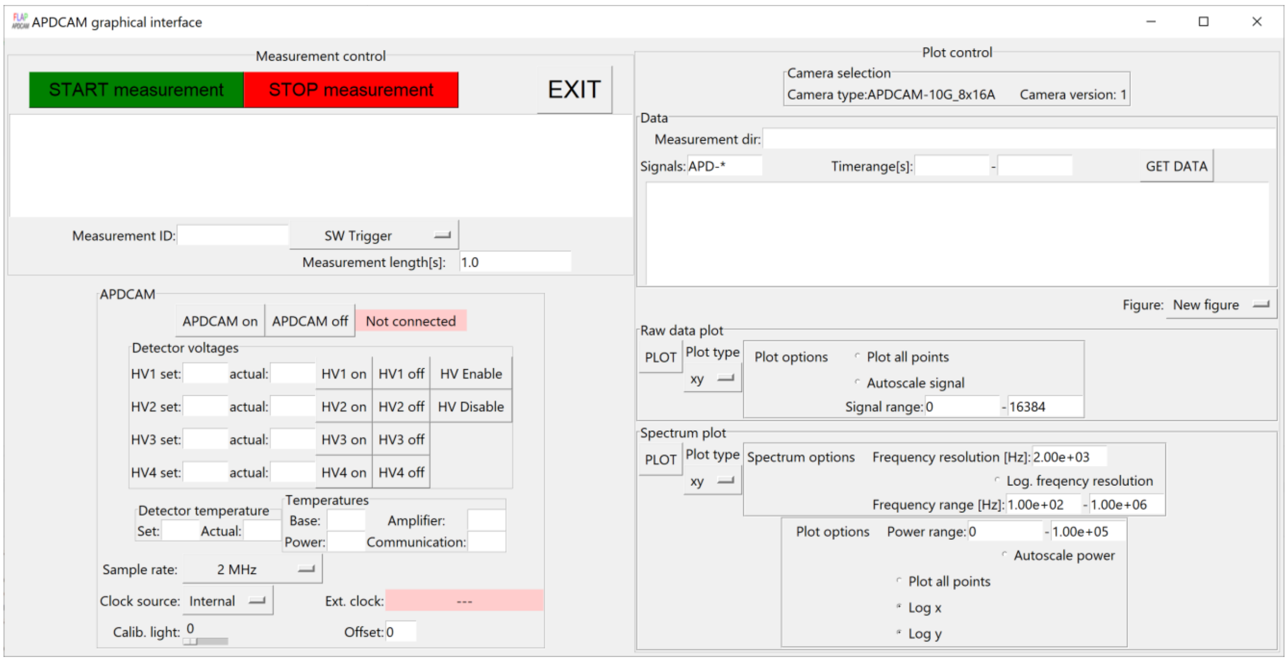
A configuration file APDCAM_GUI.cfg should be present in the working directory. The contents is the following:

```
[General]
 Address = 10.123.13.102
 Datapath = data
 CameraType = APDCAM-10G_8x16A
 CameraVersion = 1
[Trigger]
 APDCAMStartTime = 0
 TriggerTime = 0
 InternalTrigger_APD-1-3 = (ENABLE,POSITIVE,12300)
```

The interpretation of the parameters is the following:

`Address:` The APDCAM network address.
`Datapath:` The name of a directory where data will be collected. Measurements are numbered sequentially with the date and a sequency number and each measurement will be placed into a subdirectory under the data directory.
`CameraType:` The actual camera type.
`CameraVersion:` The camera version.
`APDCAMStartTime:` The delay [s] of the APDCAM data acquisition relative to trigger.
`TriggerTime:` The time of the trigger pulse in seconds.
`InternalTrigger_`<channel name>: This entry sets up internal trigger for one channel. A trigger condition can be setup for as many channels as necessary, all should be a different entry. The channel name can be APD-<row>-<column> for 2D APDCAM, APD-<channel> for the APDCAM-10G_FC, or alternatively ADC<channel> when directly an ADC channel is named. The format of the entry is (<enable>,<polarity>,<level>), where
   <enable>: ENABLE or DISABLE or any abbreviation
   <polarity>: POSITIVE or NEGATIVE or any abbreviation
   <level>: Is the trigger level in bits assuming 14 bit resolution, that is between 0 and 16383.

The window of the GUI is shown below.



Measurement ID:
SW Trigger
Measurement length[s]: 1.0
Not connected
HV1 set: actual:
HV2 set: actual:
HV3 set: actual:
HV4 set: actual:
Detector temperature Set: Actual:
Temperatures Base: Amplifier: Power: Communication:
Sample rate: 2 MHz
Clock source: Internal
Ext. clock: ---
Calib. light: 0
Offset: 0
Camera type:APDCAM-10G_8x16A Camera version: 1
Measurement dir:
Signals: APD-*
Timerange[s]: -
Figure: New figure
Plot type: xy
Signal range: 0 - 16384
Frequency resolution [Hz]: 2.00e+03
Frequency range [Hz]: 1.00e+02 - 1.00e+06
Power range: 0 - 1.00e+05

The left panel is for the measurement control, while the right panel is for plotting.

After starting the GUI a connection should be established to the camera by the “APDCAM on” button. When the connection state switches to green the camera is connected and the value of various camera parameters are continuously shown in the respective fields. The detector voltages should be entered in the “HVx set” fields. Enter should be pressed to set the value in the camera. The detector voltages can be enabled by the “HV Enable” button and can be switched on by the “HVx on” buttons. The detector temperature can be set (pressing Enter) in the “Set” field while the actual detector temperatures are shown in the “Actual” field. Other camera temperatures are shown in the Temperatures block.

The camera clock source can be internal or external, as selected in the “Clock source” droplist. If external clock is selected the actual frequency of the clock is shown in the Ext. clock field. This is a reference clock to the camera and not sampling clock. 10 MHz should be used when using this GUI.

The Measurement length can be given in seconds above the APDCAM block. The measurement can be started with the “START measurement” button. If “SW trigger” is selected the measurement starts immediately. If “HW trigger” is selected the camera waits for a positive edge on the TTL trigger input of the camera. The third option is “Internal trigger”. In this case the measurement will start if any of the internal trigger conditions listed in the configuration file becomes valid.

The camera calibration light can be regulated at the bottom of the panel.

The “Offset” field shows the offset setting of all ADC channels. The standard offset value is set in the camera, in most cases it need not be modified. If the camera is operated at high detector voltage, high detector temperature or otherwise the detector leakage current is higher than normal this value can be modified to shift the signal offset into the ADC measurement range.

The plot Graphical User Interface

The plot panel of the GUI is part of the measurement control GUI, but it can also be started with the following command:

```
flap_apdcam.plot_gui()
```

The plot gui can handle both APDCA-1G and APDCAM-10G camera data. At the top of the panel the camera type and version can be selected. If the panel is part of the measurement control GUI (as shown above) the camera version is taken from the configuration file and cannot be modified. If the panel is standalone the camera type and version can be selected.

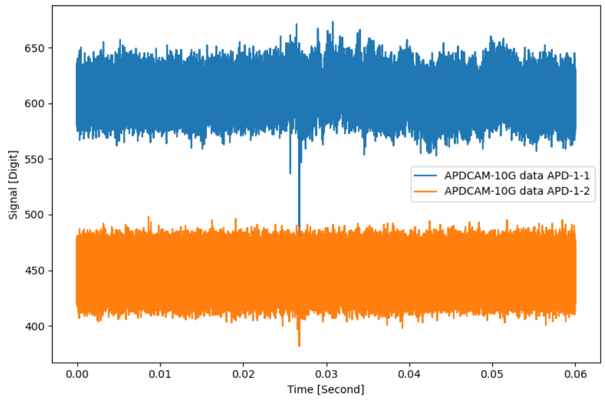
In the “Data” block the data can be read into memory. The “Measurement dir” contains the data directory (the directory of the actual measurement). This field is automatically filled at the end of a measurement, but can also be entered manually. The “Signals” field contains the channel description as described above in the “name” parameter of get_gata(). The “APD-*” in the example means all APD pixels. In the two “Timerange” fields the start and end time of the required data read can be entered on the timescale of the measurement. (Taking into account TriggerTime in the configuration file.) If left empty, all data are read. Pressing the “GET DATA” button reads the requested data into memory, into a flap DataObject. This data object can be plot using the “Raw data plot” and the “Spectrum plot” blocks below.

The “Figure” droplist is used for handling figures, where various flap plots can appear. An arbitrary number of figures can be crated using the “New figure” selection. If no figure is present when the fist plot is created, a figure will be automatically generated. The figure numbers appear in the droplist, any of them can be selected. If the figure already contains a plot a new plot will be overplot, if the axes and plot types are the same. A figure cannot be erased, but it can be closed and it will be regenerated as an empty plot when selected. The figures are Matplotlib interactive figures, axes can be modified, points read, etc.

The “Raw data plot” shows raw signals, while the “Spectrum plot” block shows autopower spectra of the signals. In each block the following plot types can be selected:

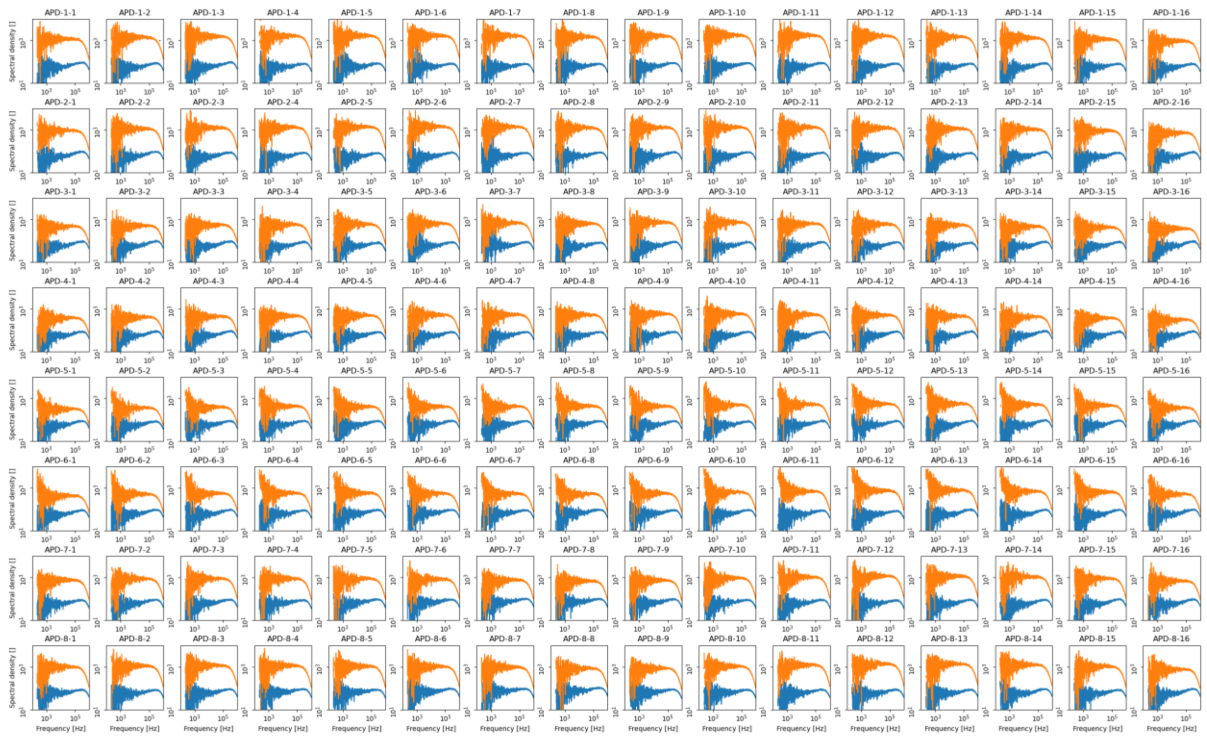
- **xy**

A simple time or frequency plot of one signal Another signal can be overplot. An example is shown below.



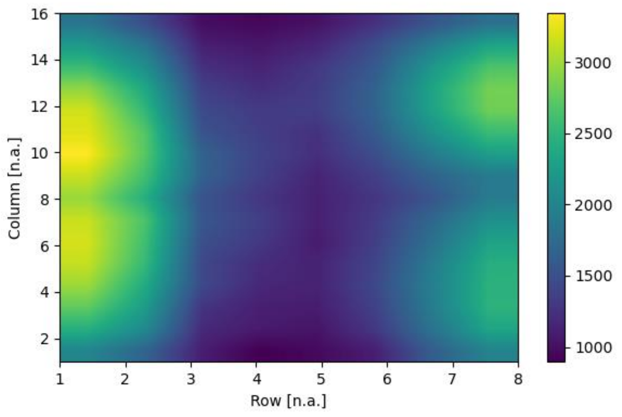
- **grid xy**

  A 2D matrix of xy plots following the APD pixel layout. The x axis of the sublots are connected. If fixed signal range is used the y axis are also connected. Other signals can be overplot. An example is shown below. In this case noise power spectra of a 8x16 channel APDCAM are plot. The blue curves are without light, the orange with light.



- **image**

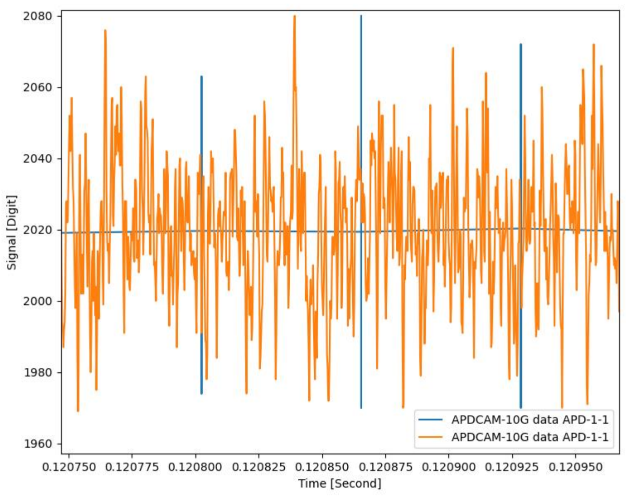
An image of the 2D mean APD signals in the full time interval (raw data plot) or the mean power in the frequency range (spectrum plot). An example for a raw signal image plot is shown below.



- **anim-image**
  The 2D image of the APD signals or the power is plot as a video. The time in the video is either the measurement time (raw data plot) or frequency (spectrum plot).

Most of the plot parameters are standard settings, some of them needs some explanation:

- **Plot all points**
  APDCAM signals often contain millions of samples. If that many points are plot it takes too long time and Matplotlib cannot handle it. To overcome this problem, flap by default reduces the number of data points to 2000 and at each point plots a vertical bar between minimum-maximum in the time range of the plotted points. This enables to plot long signals, but when the x axis is zoomed one can see the vertical bars and



  reduced number of points. This behaviour can be avoided by clocking the ‘plot all points’ checkbox. The figure below shows a small part of a signal plot without (blue curve) and with (orange) this option on.

- **Log frequency resolution**
  For calculating power spectra one needs a frequency resolution and a frequency range. Power spectra are often plot on logarithmic frequency scale. In this case the fixed frequency resolution results in a small number of points at the start of the axis and large

number of points at the end. This also means the scatter of points will be unnecessary high towards the end of the frequency scale. Flap can optimize the plot by calculating power spectra with frequency resolution changing relative to the frequency. This means on a log f-axis plot the density of frequency points is constant and the scatter is optimal. An example is shown in the sample plot for grid xy.

The default value of the plot options are loaded from the [PS] section of the flap_defaults.cfg configuration file in the working dierctory. This also contains other settings (e.g. Hanning window) which cannot be changed from the GUI. For more detailed control of the plots and power spectrum calculation the flap plot method should be used in a program.

Example programs

In the "examples" directory some example codes can be found:

- flap_apdcam_example.py
  Example code how to use the flap_apdcam module.
- flap_apdcam_control_example.py
  Example of measurement program with APDCAM. Opens camera, switches detector HV on and does a measurement. Finally plots a channel data.
- apdcam_plot_gui.py
  Starts the APDCAM plot Graphical User Interface.
- apdcam_gui.py
  Starts the APDCAM measurements and graphical interface.