Quick start with the MAST Upgrade BES flap module

S. Zoletnik, 12 August, 2023

Preparing to work

Clone flap and flap_mastu_bes from github.
Set PYTHONPATH so as both are found.

Create a working directory and put a config file, e.g. flap_mastu_bes\test\flap_defaults.cfg there:

```
[Module MAST_BES]
 Datapath = z:/data/MAST/BES
 Scaling = Volt
 Offset timerange = [-0.1,-0.01]
```

Try the tests described below or see

```
 flap_mastu_bes\test\test_mast.py
```

Reading and plotting a single channel

Reading data, a single channel: BES-1-2. This is the pixel second from left on the top row as looking onto the detector:

```
 d = flap.get_data('MAST_BES',exp_id=48193,name='BES-1-2')
```

The result is a 1D FLAP DataObject. Contents can be listed as

```
 flap.list_data_objects(d)
```

The output:

```
<1>(data_source:"MAST_BES" exp_id:"48193") data_title:"MAST BES data, BES-1-2" shape:[6000000][no error]
Data name:"Signal", unit:"Volt"
Coords:
Time [Second](Dims:0) [<Equ.><R. symm.>] Start: -1.000E-01, Steps: 2.500E-07
Sample [n.a.](Dims:0) [<Equ.><R. symm.>] Start: 0.000E+00, Steps: 1.000E+00
ADC Channel [n.a.](Dims:), Shape:[1]) [<R. symm.>] Val:10
Signal name [n.a.](Dims:), Shape:[1]) [<R. symm.>] Val:BES-1-2
```

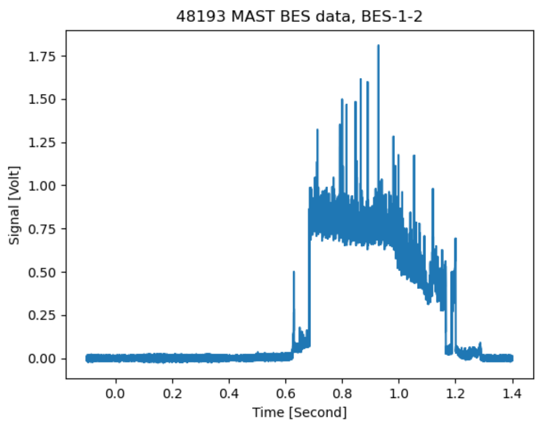
This has 6 million data points and no error is associated. The time coordinate is equidistant, has 0.25 microsec step and starts at -0.1s. It changes along dim 0 of the data array, this is shown by Dims: 0.

There is also a similar Sample coordinate. The “ADC Channel” coordinate has no Dim, which means it is the same for all data points. Also the “Signal name” coordinate is the same for all data.

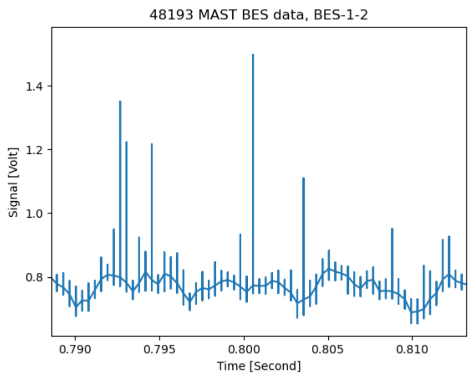
A simple plot:

```
  d.plot()
```

This plots data as a function of the first coordinate, that is Time:



This is fast, as the FLAP plot function does not plot all the 6 million points, but resamples to about 2000. If you zoom in you will see the result:



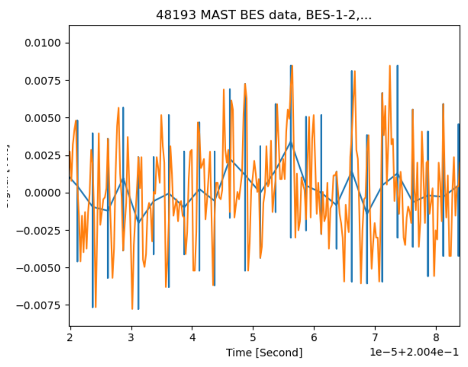
The vertical bars indicate the minimum/maximum at a point. You can also plot part of the signal by taking a slice of the signal in time:

```
 d.plot(slicing={'Time':flap.Intervals(0.2,0.21)})
```

This is 10 ms, 40000 points, therefore flap still reduces the number of points. But it is possible to plot all points as well:

```
d.plot(slicing={'Time':flap.Intervals(0.2,0.21)},options={'All points':True})
```

Doing the two plots into one figure and zooming in you can see the difference:



The plot axis can also be set to Sample:

```
  d.plot(axes='Sample')
```

“Axes” is string or list of strings setting the x,y axes for this simple plot. If only the first axis is given the second is the data. But it is possible to plot one coordinate as a function of another:

```
 d.plot(axes=['Sample','Time'])
```

It is also posible to plot one signal as a function of another:

```
 d1 = flap.get_data('MAST_BES',exp_id=48193,name='BES-1-3')
 d.plot(axes=d1)
```

And also you can plot Time as a function of data:

```
 d.plot(axes=['__Data__','Time'])
```

Data read options

The read timerange can be set using the coordinates argiment:

```
d=flap.get_data('MAST_BES',exp_id=48193,coordinates={'Time':[0.1,0.2]},name='BES-1-3')
```

It is also possible to do resampling already during reading of the data.
E.g. to resample to 1 kHz:

```
d = flap.get_data('MAST_BES',exp_id=48193,name='BES-1-3',options={'Resample':1e3})
```

This is useful when reading all data as 1.5s measurement results in a 3 Gbyte DataObject which takes a lot of memory. If resample is used data will also have error in the DataObject which represents the scatter of the data in one resample interval. This is plot by the plot() method.

Possible signal names:
  BES-<row>-<column>, e.g. BES-1-2
  ADC<number>” e.g. ADC23

To read more than one channel wildcards can be used, or names listed. See in next section.

Some other options for data read:

‘Datapath:
    The directory where the data are found.
‘Test measurement’:
    Reads xbtzXXXXXX.nc files.
‘Scaling’:
    Can be ‘Volt’ or ‘Digit’, sets the signal scaling.
‘Offset timerange’:
    List of two times. The mean signal in this time will be used as offset and subtracted from the signal. Of None, no offset subtraction will be done.

Data can also be stored in the FLAP memory storage:

```
flap.get_data('MAST_BES',exp_id=48193,name='BES-1-2',object_name='BESDATA')
```

This saves the DataObject in FLAP storage under name BESDATA with shot number (exp_id) 48193. The flap.list_data_objects() call lists FLAP storage:

```
BESDATA(data_source:"MAST_BES" exp_id:"48193") data_title:"MAST BES data, BES-1-2" shape:[6000000][no error]
  Data name:"Signal", unit:"Volt"
  Coords:
Time [Second](Dims:0) [<Equ.><R. symm.>] Start: -1.000E-01, Steps:  2.500E-07
Sample [n.a.](Dims:0) [<Equ.><R. symm.>] Start:  0.000E+00, Steps:  1.000E+00
ADC Channel [n.a.](Dims:), Shape:[1]) [<R. symm.>] Val:10
Signal name [n.a.](Dims:), Shape:[1]) [<R. symm.>] Val:BES-1-2
```

From the FLAP storage a plot can also be made:

```
flap.plot(‘BESDATA’)
```

Reading and plotting multiple channels

Multiple channels can be read by listing multiple channel names or using extended wildcards. Here “extended” means that additionally to normal wildcards as used in operating systems, ranges can be defined for multi-digit numbers as well.

The resulting data will be either 2 or 3 dimensional. If the requested channels form a 2D range a 3D object is returned, otherwise a 2D. If any ADC channel is requested a 2D object is returned.

Valid channel specifications:

Channel specification	**Description**	**Data object dimension**
BES-1-*	BES pixels in row 1	2
BES-*	All BES pixels	3
BES-[2-4]-*	BES pixels in rows 2,3,4	3
BES –[2,4]-*	BES pixels in rows 2,4	3
ADC*	All ADC channels	2
A*	All ADC channels	2
ADC[12-23]	ADC channels from 12 to 23.	2

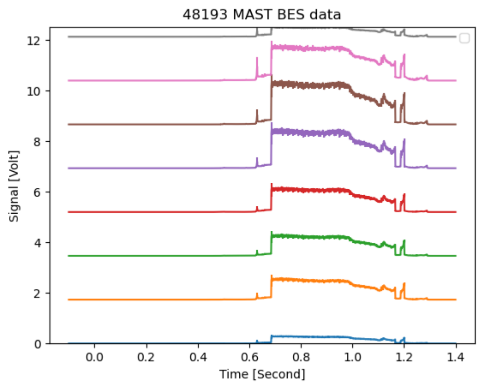
If BES channels are read a “Row” and “Column” coordinate is added to the DataObject.

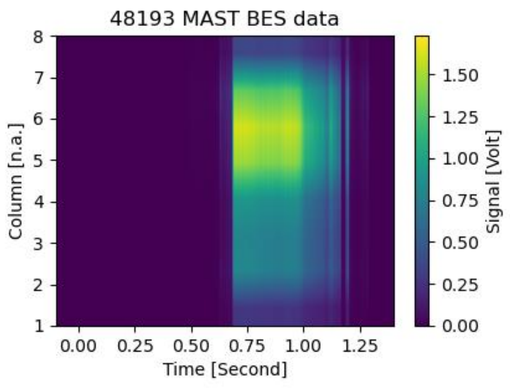
For 2D DataObjects the default plot is “multi xy”:

```
d=flap.get_data('MAST_BES',exp_id=48193,name=['BES-1-*'],options={'Resample':1e3})
d.plot()
```

Another possibility is the “image” plot, as shown on the right side:

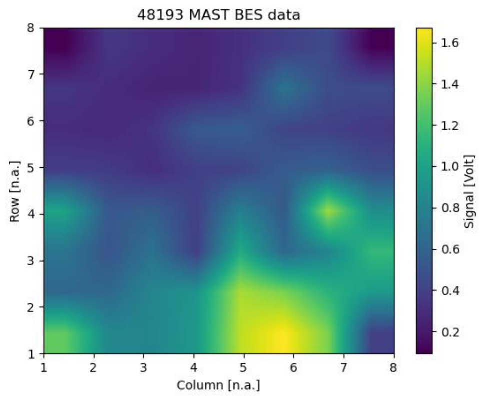
```
d.plot(plot_type='image',axes=['Time','Column'])
```





All the BES channels form a 3D DataObject. The data at a time slice can be plot by slicing in time. Here the axes has to be set, this is why the Row and Column coordinates are included:

```
d=flap.get_data('MAST_BES',exp_id=48193,name=['BES*'],options={'Resample':1e3})
d.plot(plot_type='image',slicing={'Time':0.7},axes=['Column','Row']
```



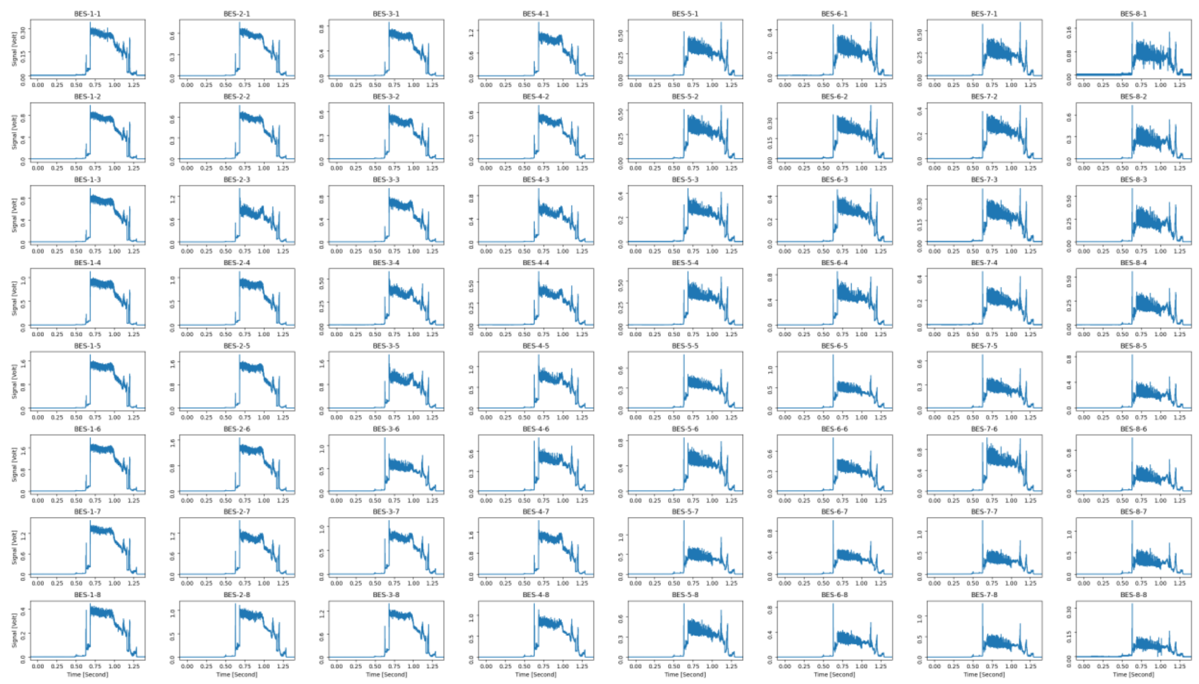
A 3D DataObject can be plot with the “anim-image” plot type, a video. This shows an image and the third coordinate is time:

```
d=flap.get_data('MAST_BES',exp_id=48193,name=['BES*'],options={'Resample':1e2})
d.plot(plot_type='anim-image',axes=['Column','Row','Time'])
```

This plot looks like the one above, but moving in time. There are many options for this plot type, like the frame rate, video filename. This plot can generate an AVI file, but openCV is needed to be installed on the computer.

The full BES data can also be plot as time evolution of all the signals. This can be done with the “gid xy” plot type:

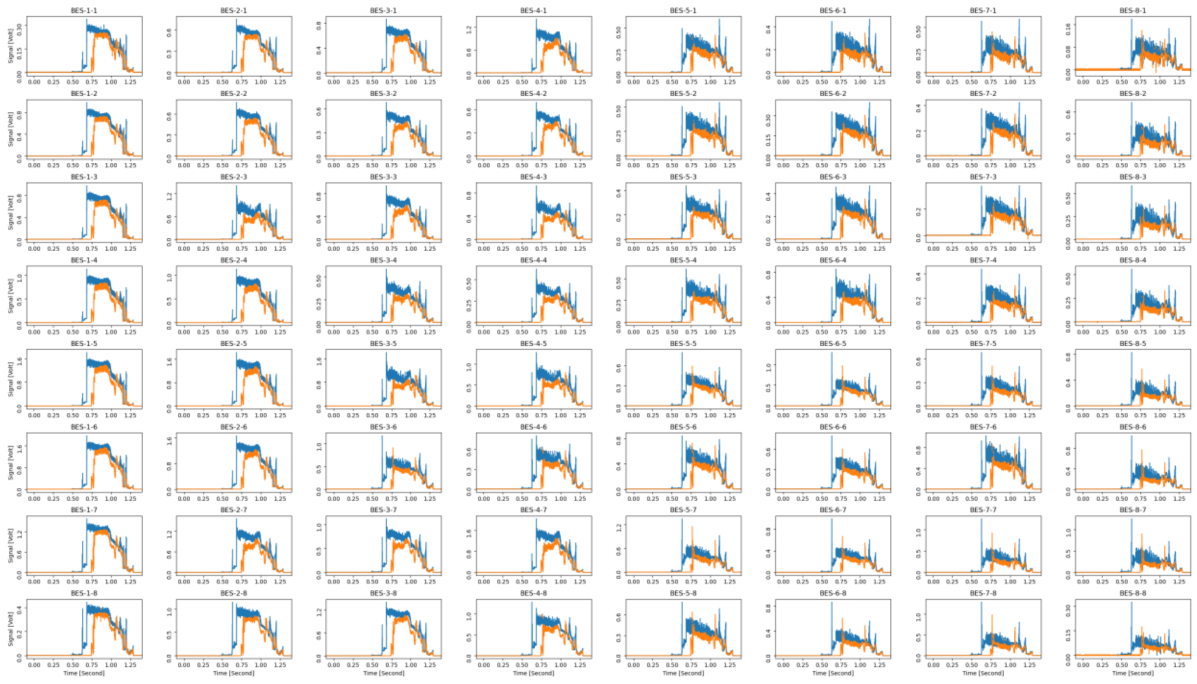
```
d=flap.get_data('MAST_BES',exp_id=48193,name=['BES*'],options={'Resample':1e3})
d.plot(plot_type='grid xy',axes=['Column','Row','Time'])
```



The y scale of the individual plots can be identical or autoscaled to data. In this plot type the x (and in some cases the y axes as well) are coupled. Zooming into a timerange zooms in all plots.

It is also possible to overplot these flap plots with other data. E.g. the above grid plot can be overplot for two shots.

```
d=flap.get_data('MAST_BES',exp_id=48193,name=['BES*'],options={'Resample':1e3})
d.plot(plot_type='grid xy',axes=['Column','Row','Time'])
d=flap.get_data('MAST_BES',exp_id=48198,name=['BES*'],options={'Resample':1e3})
d.plot(plot_type='grid xy',axes=['Column','Row','Time'])
```



FLAP plots has a lot of options. The state of a plot can be also stored in a plot_id returned by the plot method. This can be supplied to another plot to overplot.