P06 Nexus NXS processing in GeoPIXE

Overview of pipeline

The two main methods for access to the data are (i) reading "header" metadata to extract parameters like the pixel size of an image (nx,ny), energy, sample name, etc. and also pull out data for the stage encoders, dwell (exposure time) as a function of sequence number, and (ii) reading the Xspress3 NXS data files to get spectral data and associating that with pixel address x,y to form vectors of photon events (E,x,y, …).

More detail

The methods in the device objects generally include (i) a few that handle "options", (ii) 'get_header_info', which calls the 'read_petra_p06_h5_header' routine, (iii) 'update_header_info', which copies the read header info into the 'self' structure of the object (can avoid unnecessary re-reading of long header files), (iv) 'flux_scan', which is usually used to look for the PV names and settings for flux (it also calls 'read_petra_p06_h5_header'), (v) 'read_setup', setup photon event vectors for each read spectral data NXS file, (vi) 'read_buffer' read next buffer of data from the spectral data NXS file, (vii) a few that handle "import" approaches and (viii) 'init', which is the object initialize method, which is only called when a new instance of the object is created.

"Option" GUI elements

"Options" are GUI elements that can appear across GeoPIXE (e.g. "device" parameters in the *Import spectra* and *Sort EVT* windows) for control of internal "device" parameters. These are not really used at this time for this P06 NXS device object. They are place holders for later features, and currently just save a default 'version' value.

Import spectra

"Import" methods provide parameters to use for (i) the *Import Spectra* call (Menu: "Import→Spectra" in the *spectrum display* window), and (ii) custom local spectral data reads. In this device, we are only using the first, which is handled via the *Import Spectra* approach (see routines "import_select" and "spectrum_load" called from *spectrum display* window), which scans all raw spectra/event data in Xspress3 NXS files to accumulate spectra for 'E' and the projection of all events onto X,Y axes. The definition for this is …

```
opt_39 = define(/import)                                    ; XFM new HDF5 file read as a list-mode
        opt_39.name =         'petra_p06_h5_evt'            ; unique name of import
        opt_39.title =        'Extract E,X,Y from P06 map HDF5 file as list-mode'
        opt_39.in_ext =       '.nxs'                        ; input file extension
        opt_39.request =      'Select P06 map HDF file to scan for all spectra, X,Y'
        opt_39.preview =      0                             ; allow spectrum preview
        opt_39.raw =          1                             ; flags use of separate Raw data path
        opt_39.multifile =    1                             ; denotes multi-file data series
        opt_39.separate =     '_'                           ; char between file and run #
        opt_39.spec_evt =     1                             ; uses call to 'spec_evt' to extract events
        opt_39.use_IC =       1                             ; pop-up the 'flux_select' PV selection panel
        opt_39.IC_mode =      1                             ; default to using PV for IC
```

This tells GeoPIXE that NXS data is found in a series of files (multifile=1) with each named with a sequence number after an underscore separator (separate='_') and a file extension of "nxs" (in_ext=".nxs"). 'Title' is for the *Import Spectra* popup title, 'request' is a title for the *File-Requester* popup, 'raw' flags a separate path for raw data and analyzed data, 'use_IC' flags using a popup 'flux_select' to choose the PV to use for flux values, 'IC_mode'=1 means a default mode of using a PV selection for flux with gain settings.

Header read (“read_petra_p06_h5_header.pro”)

Reading “header” metadata is needed to extract parameters like the pixel size of an image (nx,ny), energy, sample name, etc. and also pull out data for the stage encoders, dwell (exposure time) and flux as a function of sequence number. It also “manufactures” names for flux variables (GeoPIXE uses the Epics jargon of PV names), such as “adc_1/value1”, … and “qbpm_1/value1”, ...

If you look at the code for ‘read_petra_p06_h5_header’, it starts with comments showing the main parameters stored across the dir tree. On entry, the first Xspress3 data file is already open and ‘fstat’ is used to get its filename. This is done so we can navigate to associated directories to find “adc_*” and “qbpm_*” dirs and the files therein.

It starts looking for any dirs named “pilctriggergenerator_”. The main goal here is to dig out all data for “encoder1” and “encoder2” (from all files in this dir). This gives pixel position as a function of sequence number. From these we calculate pixel address “pixel_x, pixel_y” versus sequence number using these lines.

```
pixel_x = round( (encoder1 - min(encoder1)) / abs(step) )
pixel_y = round( (encoder2 - min(encoder2)) / abs(step) )
nx = max(pixel_x) + 1
ny = max(pixel_y) + 1
```

We also pull out dwell time per pixel (“exposuretime” stored in ‘dwell_array’) as a function of sequence number across all NXS files in any dirs named “adc_*”. We then populate a 2D array of dwell time (“maia_dwell”) making use of the pixel address and sequence number. The array “maia_dwell” is stored crudely in a common block for now, so it can be accessed when scanning data later in ‘read_setup’ and ‘read_buffer’.

```
maia_dwell = fltarr(nx,ny)
maia_dwell[pixel_x,pixel_y] = dwell_array * 1000.                    ; dwell in ms
```

A common/representative dwell “common_dwell” is also estimated.

Processing spectra/image data

Reading Xspress3 data is done in the methods “read_setup” and “read_buffer”. ‘read_setup’ is called for each new Xspress3 NXS file and then ‘read_buffer’ is called, normally in a loop until all data is read. In this device object, this looping is used access all “channelxx” detector data found in the NXS files (indexed using ‘i_petra_channel’).

On entry into ‘read_setup’, the next Xspress3 NXS data file is already open and ‘fstat’ is used to get its filename so we can open that using HDF5 routine ‘H5F_OPEN’. We first read sequence number vector for this NXS file and determine the pixel address vectors ‘x,y’ for this sequence number vector (‘sequence’) using the saved ‘pixel_x, pixel_y’. Note that P06 sequence number start at 1, while vector indices need to start at 0.

‘read_setup’ then determines how many ‘channlxx’ detectors are present and sets ‘n_petra_channel’. It then creates some arrays that will be used in ‘read_buffer’ to assemble vectors for the events stream for pixel address (x,y), “station” or channel number (ste), and photon energy (e).

‘read_setup’ then looks for the *same named* NXS file in the dirs “adc_*” or “qbpm_*’ to find the vector of flux values (‘values’) as a function of sequence number. The chosen PV is given by ‘flux_ic.pv’. Its gain units are set in ‘flux_ic.val’ and ‘flux_ic.unit’, which for now are set to unity (1 nA/V, i.e. nsls_flux_scale=1.0). It then sets the corresponding pixel flux values in an array ‘flux’ across the image. Here we distinguish between imaging mode, where we need to maintain a 2D flux array and spectrum mode, where we just accumulate a total flux.

```
nsls_flux_scale = flux_ic.val * flux_ic.unit
tflux = nsls_flux_scale * values                    ; flux PVs are counts
if self.spectrum_mode then begin
    flux = flux + total(tflux)
endif else begin
    flux[x,y] = tflux                               ; distribute flux by sequence # into flux by x,y
endelse
```

'read_buffer' reads the spectra data from the Xspress3 "histogram" for each channel 'i_petra_channel'. It then builds vectors of photon events 'e,ste,x1,y1,multiple for return to GeoPIXE (e.g. to 'da_evt' or 'spec_evt' routines). The vector 'multiple' makes it simple to convert a spectrum into a pseudo photon event stream by setting 'multiple' to the counts in each bin of the spectrum histogram.