myana user examples

The myana user guide already has a lot of examples linked in! And here are a few more...

- Acqiris waveform data
- Beamline data (Bld)
- Display images from Princeton camera

This page holds a few example code-snippets for use in myana analysis. This analysis is written in C++ and the examples here use root for plotting. Compare with Pyana user examples to see how the same things can be done using the pyana analysis framework.

Acqiris waveform data

This method is good for any waveform data from the Acqiris digitizers, just replace the *source* name (here *AmoITof* for the AMO Ion Time of flight detector) with some other Acqiris waveform detector.

Include and global variable definitions:

```
#include <TProfile.h>

static int        numChannelsITof;
static int        numSamplesITof;
static double     sampleIntervalITof;
static TProfile*  profileITof = NULL;
```

In `beginjob()`, read configuration data and book a *profile histogram* (for each sample interval (x-axis), draw the average number of samples (y-axis), i.e. the average waveform of all events).

```
  int fail = getAcqConfig( AmoITof, numChannelsITof, numSamplesITof, sampleIntervalITof);
  if ( fail != 0 )
    printf( "begin(): getAcqConfig() failed, code = %d\n", fail );

  profileITof = new TProfile("avg","avg",numSamplesITof,0.0,sampleIntervalITof,"");
  profileITof->SetYTitle("Volts");    //optional
  profileITof->SetXTitle("Seconds");  //optional
```

In the `event()` function, get the Acqiris value for this event and add it to the histogram. The `getAcqValue` function will point `timeITof` and `voltageITo` to arrays of readout values.

```
  double* timeITof;    // array: sample interval
  double* voltageITof; // array: readout values
  int channel = 0;
  int fail = getAcqValue( AmoITof, channel, timeITof, voltageITof);
  if ( fail != 0 )
  {
    printf( "event(): getAcqValue() failed, code = %d\n", fail );
  }
  else
  {
    for (i=0;i<numSamplesITof;i=i+1)
      {
        double t = timeITof[i];
        double v = voltageITof[i];
        profileITof->Fill(t,v);
      }
  }
```

The resulting histogram can be drawn in a root session and would look something like this:

Beamline data (Bld)

To read out energy, charge and position of the beam from the beamline data, use getEBeam(). The function returns 0 if data is available and assignes values to its arguments from the current event. Call from within `event()`.

```
  double ebcharge; double ebenergy; double posx; double posy;
  double angx; double angy;
  int fail =  getEBeam(ebcharge, ebenergy, posx, posy, angx, angy);
  if (!fail) printf("ebeam: %f, %f, %f, %f, %f, %f \n",
                    ebcharge, ebenergy, posx, posy, angx, angy);
```

To read out the energy from the front end enclosure (FEE) gas detector, use getFeeGasDet(). This assigns this event's value to an array of 4 numbers:

```
  double shotEnergy[4];
  int fail = getFeeGasDet( shotEnergy );
  if (!fail) printf("GasDet energy: %f, %f, %f, %f \n ", shotEnergy[0], shotEnergy[1], shotEnergy[2], shotEnergy
[3]);
```

To read out fit time and charge of the phase cavity, use getPhaseCavity() which assigns values to four arguments:

```
  double phaseCavityTime1, phaseCavityTime2, phaseCavityCharge1, phaseCavityCharge2;
  int fail = getPhaseCavity( phaseCavityTime1, phaseCavityTime2, phaseCavityCharge1, phaseCavityCharge2 );
  if (!fail) printf("PhaseCavity: %f, %f, %f, %f \n",
                    phaseCavityTime1,  phaseCavityTime2, phaseCavityCharge1, phaseCavityCharge2);
```

Display images from Princeton camera

Some variables are needed both in `beginjob()` and `event()` member functions. The myana solution is to declare them as global variables right after the necessary include-files.

```
#include <TH2I.h>

TH2I*        h_image;
static int   widthSxrRCI0;
static int   heightSxrRCI0;
static int   orgXSxrRCI0;
static int   orgYSxrRCI0;
static int   binXSxrRCI0;
static int   binYSxrRCI0;
```

In `beginjob()`, get configuration object and book histogram

```
 // Processing Princeton Config Data
  int fail = getPrincetonConfig( DetInfo::SxrEndstation, 0,
                                 widthSxrRCI0, heightSxrRCI0,
                                 orgXSxrRCI0, orgYSxrRCI0,
                                 binXSxrRCI0, binYSxrRCI0 );
  if ( fail == 0 ) {// success
    printf( "Get Princeton config for SxrRCI0: width %d height %d  org (%d,%d)  binX %d binY %d\n",
            widthSxrRCI0, heightSxrRCI0, orgXSxrRCI0, orgYSxrRCI0, binXSxrRCI0, binYSxrRCI0 );
  }

  // Book a histogram
  h_image = new TH2I("h_image","Princeton images (summed)",         // name and title
                     widthSxrRCI0,orgXSxrRCI0,widthSxrRCI0,        // nBinsX, xmin, xmax
                     heightSxrRCI0,orgYSxrRCI0,heightSxrRCI0);     // nBinsY, ymin, ymax
```

In `event()`, fetch data from Princeton camera for the current event. Add to histogram (which will hold the sum of all events). Note, we here use a *root* TH2 histogram. It's rather slow for images, so you may want to use some other display service...

```
unsigned short* pixels;
int fail = getPrincetonValue( DetInfo::SxrEndstation, 0, pixels);

if ( fail == 0 ) {
  // Fill histogram from 2D array
  for(int irow=0; irow<heightSxrRCI0; irow++){
    for(int icol=0; icol<widthSxrRCI0; icol++){
      h_image->Fill(icol,irow,*pixels);
      pixels++; // go to the next pixel
    }
  }
}
```

Once all events have been processed, you can display and/or store the histogram in *myana*'s `endjob()` function.

```
h_image->Draw();
h_image->SaveAs("myana_princ_image.root");

// 1D projections
h_image->ProjectionX()->SaveAs("myana_princ_imageX.root");
h_image->ProjectionY()->SaveAs("myana_princ_imageY.root");
```

At this point, you can open the histogram in root sessions and draw them there and interactively adjust display properties as needed.

```
> root
root > TFile f("myana_princ_image.root")
root > f.ls()
root > gStyle->SetPalette(1)
root > h_image->Draw()
```

Here's the plot: