Table of Contents

Commissioning tools

Roman Pots

For Roman Pots, the tools are located in SVN:
`svn+ssh://svn.cern.ch/reps/totem/trunk/online/scripts/RP`.

Some of the tools are written in C++, so before using them, they must be compiled. Before compilation, make sure you have a suitable ROOT environment configured, for example by running

```
source /afs/cern.ch/sw/lcg/app/releases/ROOT/5.26.00b/slc4_ia32_gcc34/root/bin/thisroot.sh
```

After that's done, just type

```
make
```

and the C++ tools should compile without problems.

The tools written in Python use the PyROOT interface to access ROOT files. Unfortunately, CERN-provided ROOT is built against python-2.5, which means it doesn't work with the python-2.4 present on SLC4. To work around this, we need to use an external python-2.5 installation, for example the one provided by CERN here: `/afs/cern.ch/sw/lcg/external/Python/`.

To make things easier, I suggest creating a short shell script:

```
#!/bin/bash

PYVER=2.5.4p2

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/afs/cern.ch/sw/lcg/external/Python/$PYVER/slc4_ia32_gcc34/lib
PATH=/afs/cern.ch/sw/lcg/external/Python/$PYVER/slc4_ia32_gcc34/bin/:$PATH
export LD_LIBRARY_PATH PATH
```

and sourcing it (`source python.sh`).

After doing that, you should be able to run all the Python tools described below without problems.

VFAT Channel status analysis

A tool for analysing ROOT files containing VFAT channels pulse check data. Usage:

```
./extract_vfat_channels file.root
```

Latency scan

A tool for processing ROOT files containing trigger data from data-taking runs into intermediate text files and producing latency graphs from those files. Usage:

To process run_xxx.root files:

```
./latency_scan -r run_number -c nclocks -l latency [-o outputfile.txt] inputfile.root
```

NOTE: subsequent runs with the same output file will *append* to that file instead of overwriting it.

To process the resulting txt file and produce graphs:

```
./latency_scan -p -o outputfile_template [-t "title"] outputfile.txt
```

This will produce two graphs (*_45.gif and *_56.gif), each showing trigger efficiency for different latencies for each pot.

Trigger rate analysis

A tool to produce plots (in basically any file format supported by ROOT TCanvas Print method) from Roman Pot position data in CSV format and trigger rate data in text format. Plot type is hit rate vs. time.

Usage:

```
python rate_analysis.py [ -f N ] [ -m ] [ -n ] < -r rates.txt | -P lvdt.csv | -i
run_xxx.info > -S sector -s station -p pot [ -o plot.gif ] [ -v ] [ -t N ] [ -l ]
```

Options:

```
-h, --help
```

Shows options

```
-r, --rates
```

Specifies rate file, i.e. rates.txt, takes one or two files

= -m MASK, --mask=MASK=

Selects which pots to read data for, if 0 (zero), uses the specified pot as mask. This option can be used to manually specify pots as a mask. I.e. "0x3f03f" shows all pots.

```
-o OUTPUTFILE, --output=OUTPUTFILE
```

Specifies output file name, i.e. rates.gif. If this is not used, the output file name will be generated using the name of the rate file.

```
-S SECTOR, --sector=SECTOR
```

Manually specify sector, if not specified, uses the sector found in rate file name. This option is mandatory only when the rate file name does not contain "_45" or "_56".

```
-s STATION, --station=STATION
```

Manually specify station, default=220.

```
-p, --pot
```

Selects individual pots to show rates from, takes less than 12 arguments (as there are no more pots in two sectors). This option can be used with overlapping (-l), then the script overlaps all the selected pots, rather than draws them into different pads.

Example:

```
python rate_analysis.py -r rates_16_May_2011_45.txt -p fr_tp
```

Shows hit rates for pot at far top on sector 45. Works also with . . . `-p 45_fr_tp`, but does not do anything special.

Example2:

```
python rate_analysis.py -r rates_16_May_2011_45.txt rates_16_May_2011_56.txt -p 45_fr_bt 56_fr_tp
```

Shows hit rates for pot at far top on sector 45 and far top on sector 56.

Notice that sector has to be defined in the pot name if used with two rate files.

```
-t TRIGGER, --trigger=TRIGGER
```

Selects which trigger to use, deafult=0:

0 - global trigger (default)

1 - global trigger in coincidence with fork

2 - enables passive trigger data analysis (columns 6-11)

```
-v
```

Enables viewing as a root file.

```
-T INTERVAL, --time=INTERVAL
```

Specifies interval time for data in seconds (i.e. =-T 20). Useful when there's too much data to visualize.

```
-l
```

Enables overlapping of pots. If used with '-t 2', will overlap every pot in specified sector.

```
-n
```

Prints names for all pots.

```
-M, --masklines
```

Adds lines to indicate where mask has changed. Automatically disabled when overlapping different graphs.

NOTE: The current version of rate_analysis.py is working on python 2.6. It is also possible that the program works on python 2.4 and/or 2.5.

hst_batch_wc.py

Generates rates per clock cycles graphs from files containing semicolon-separated values. Accepts multiple input files (UNIX wildcards supported)

Usage:

```
python hst_batch_wc.py ratefile1 ratefile2 ... ratefileN [-v] [-g] [-c outputfilename]
```

`-v|--view` enables viewing the graphs in ROOT.

`-c|--canvas` prints the ROOT canvas with all graphs to specified filename. Supports any file format supported by ROOT TCanvas print method.

`-g|--graphs` prints graphs separately into ROOT .C macro files corresponding to given filenames.

VFAT configuration setting

A tool for displaying and modifying VFAT XML configuration files. It can adjust the latency in all VFATs of a given pot:

```
./vfat_config.py -i file.xml -S sector -s station -p pot -l new_latency -o newfile.xml
```

where sector is 0 or 1, station is 147 or 220 and pot is nr_bt .. fr_tp. Or it can mask the dead and noisy channels based on the output of the `extract_vfat_channels` tool:

```
./vfat_config.py -c chanmask.txt -i file.xml -o newfile.xml
```

NOTE: this tool doesn't require python-2.5, so you can run it without any preparation steps on any SLC4 machine. To run it on newer Python versions, some adjustment of the import statements at the beginning of the script might be necessary.

VMEA to ROOT conversion

A simple shell script to run monitor in batch mode to process raw data files from data-taking runs into ROOT files usable by the aforementioned scripts. Before running, edit it and adjust the parameters at the top of the script.

deadChannelsToXML.py

The purpose of this program is to convert a text file containing dead and noisy channels to XML. The text file is provided by program extract_vfat_channel which is also described in this document. The XML format is provided by Jan Kaspar (XML Schema is not created, example at https://twiki.cern.ch/twiki/bin/view/TOTEM/CompTotemRawData). For more specific documentation please look into source code.

It can be used in following way:

```
python2.6 deadChannelsToXML.py deadAndNoisyChannels.txt
```

NOTE: This script is written for python version 2.6, it is possible that it will not work with other versions. The deadAndNoisyChannels.txt can be obtained by running ./extract_vfat_channels file.root. You can find description of extract_vfat_channels program in this document.

-- TeemuJ - 12-Aug-2011

-- DominikMierzejewski - 27-Jul-2010

-- JakubSmajek - 02-Sep-2011

This topic: TOTEM > CompComTools
Topic revision: r9 - 2011-09-02 - JakubSmajek