TASCA Go4 Analysis

OpenOffice document tascaGo4intro.odt (H.Essel, 26. May 2009)

Setups

Set up account

The tasca account should be customized for more convenience. One should define a variable for the repository path:

```
export SVN=https://subversion:443/goofy/go4/applications/tasca
```

To create a new working copy of the repository, create a directory and

```
mkdir myws
svn checkout $SVN myws
svn info
```

Then one can use svn commands like

```
svn list $SVN
```

to get a listing of the subversion repository. Some useful alias:

```
svndiff='svn diff --diff-cmd /usr/bin/diff -x "-EwbB" '
svndiffl='svn diff --diff-cmd /usr/bin/diff -x "-qEwbB" '
```

On a workspace directory these give a list of files different from repository (second line file list only).
Above has been added to .bashrc file (HE). Other useful alias can be defined here.

Set up working directory

Once the directory is made an svn working directory (by checking out a repository to it) there are few commands to deal with the repository:

`svn info`
: show the repository the workspace belongs to

`svn list $SVN`
: list of repository

`svn update`
: update workspace from repository

`svn commit -m “enter here comment” [file]`
: copies all changed files to repository. If a file is specified, only this file is copied (if modified).

After login

Setup everything for Go4 (**now already done in .bashrc**)

```
. go4login 402-00
```

(Note the space behind the dot.)

The Go4 analysis

To build the Tasca analysis, simply:

```
make
```

The executable made is

```
MainUserAnalysis
```

It can be called from shell or is started from GUI. In principle it does the same in both cases. The analysis is steered by a

ROOT macro file setup.C. You can edit this file before running the analysis. There are the following lines:

```
TString unpackProcess("yes");
TString unpackStore("no");
TString unpackOverWrite("yes");

TString caliProcess("yes");
TString caliStore("no");
TString caliOverWrite("yes");

TString checkProcess("yes");
TString checkStore("no");
TString checkOverWrite("yes");

TString analysisProcess("yes");
TString analysisStore("no");
TString analysisOverWrite("yes");

TString autosave("yes");
Int_t autosaveinterval=0; // after n seconds, 0 = at termination of event loop
```

Batch mode

Examples:

MainUserAnalysis -f file.lmd

processes file

MainUserAnalysis -t r4-4 10000

connects to MBS transport node R4-4 and processes 10000 events.

Usually in batch mode one either writes an auto-save file (containing all histograms, parameters, etc.), and/or any event file. The auto-save file name and the event file names are prefixed by the input file or node name

b_r4-4_AS.root, b_r4-4_Unpacked.root, b_r4-4_Calibrated.root, b_r4-4_Analysis.root

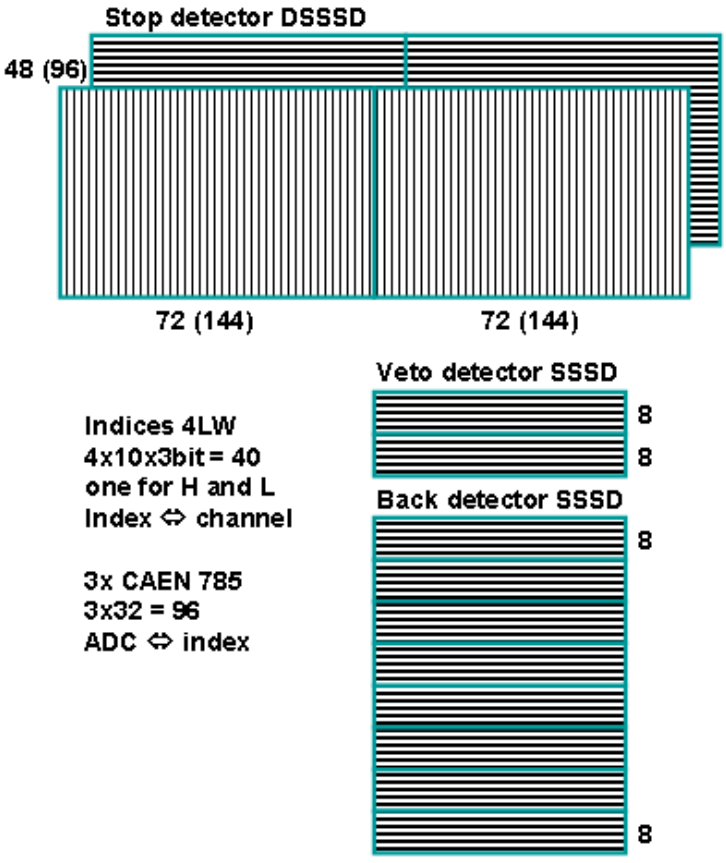
The b_ is added in batch mode only. Any of these can be opened by ROOT or in the GUI.

Interactive mode

In interactive mode the analysis is started by the GUI. In this case, the file name prefix is the analysis name specified in the Start Client panel. This name is saved by Save Settings. In addition the prefix b_ is changed to i_. Further setup is specified in the configuration panel coming up after starting the analysis. Default settings are the ones from setup.C. This setup can be modified interactively and can be stored in

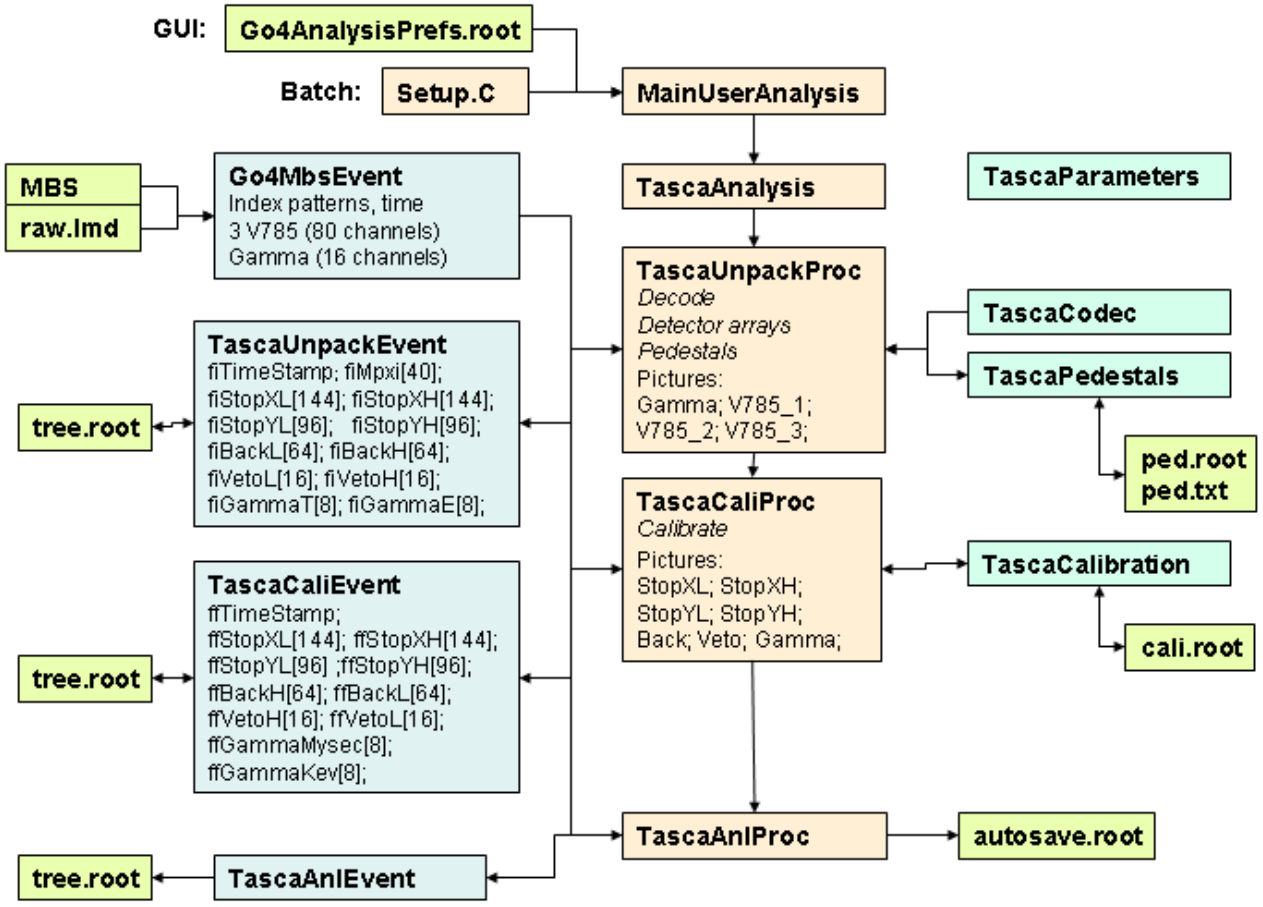
Go4AnalysisPrefs.root

from where it is retrieved next time the analysis is started. If this file is present, the settings from setup.C are overwritten.



	Channels (H+L)	ADCs Multipl: 8
Stop detector DSSSD	96 (192)	12 (24)
	144 (288)	18 (36)
Veto detector SSSD	16 (32)	2 (4)
Back detector SSSD	64 (128)	8 (16)
	320 (640)	40 (80)

The analysis steps



The analysis is divided into three steps as shown in the figure.

Unpack step

Input: LMD file or MBS (transport, stream server, event server)

Output: ROOT tree with values of all detector channels and detector hit lists. Details in TascaUnpackEvent.h

Autosave: Controls, Parameters, Pedestals and Codec
Histograms in directory Unpack: Adc_nn GammaE_n GammaT_n Pedestals Contents AdcAllRaw AdcAllCal TraceRaw_nn TraceE_nn Hist_nn Pileup_nn

Processing: TascaUnpackProc constructor creates the parameters, histograms and pictures. Method *TascaUnpack* uses parameter class TascaCodec to decode Adc values, gamma values, and fills the data fields of TascaUnpackEvent. TascaCodec also contains the mapping tables for the multiplexed channels.

Calibration step

Input: TascaUnpackEvent (from Unpack step or from file)

Output: ROOT tree with calibrated values of all detector channels and gammas. Hit indices of all detectors and their values.
Details in TascaCaliEvent.h

Autosave: Controls, Calibration
Histograms in directory Cali: All detector channels, gamma channels, Sum of detector channels.

Processing: Filling histograms and TascaCaliEvent data fields.

Analysis step

Input: TascaAnlEvent (from Calibration step or from file)

Output: ROOT tree with data from TascaAnlEvent.h (currently none)

Autosave: Creates parameters Controls, Parameters
Histograms in directory Anl: 2d histograms of stop detector (Energy-Xstripe) for each Ystripe.

Processing: Fills histograms

Calibration

An automated generation of calibration coefficient files is done by macro
makecali.C(prefix, rootfile)
root -b -l "makecali(\"test\",\"test_AS\")"
where prefix is a string used as prefix for all file names generated, rootfile is the name of the ROOT file containing the histograms (given without trailing .root). The macro should be adjusted. Several parameters can be set inside.

Histograms/Cali/StopXL:	prefix_StopXL[144]
Histograms/Cali/StopYL:	prefix_StopYL[96]
Histograms/Cali/StopXH:	prefix_StopXH[144]
Histograms/Cali/StopYH:	prefix_StopYH[96]
Histograms/Cali/BackH:	prefix_BackH[64]
Histograms/Cali/BackL:	prefix_BackL[64]
Histograms/Cali/VetoH:	prefix_VetoH[16]
Histograms/Cali/VetoL:	prefix_VetoL[16]
Histograms/Unpack/GammaE:	prefix_GammaE[8]
Histograms/Unpack/GammaT:	prefix_GammaT[8]

The format of the calibration files is:
name value
The format of the generated files is:
name index a0 a1 a2 : NOF ChiSquare

Class TascaCalibration is the parameter class holding the coefficients. This parameter is used in the TascaCaliProc processor of the second step.
To enable/disable the calibration the macro
setcali.C
must be edited. If enabled, it reads the files produced by makecali. For these the prefix string must be set.

Class TascaCaliFitter is a parameter class with the purpose of doing the calibration interactively. This might be necessary if the automatic calculations do not work for a histogram. This parameter is used in the TascaCaliProc processor of the second step. Calculating calibration parameters is done in two steps. First we need a histogram with the measured lines and a text file with the energies of these lines. These are present in arrays inside the parameter. First fitter LineFitter is used to find out true channel numbers for corresponding lines in calibration spectrum. This fit should be done interactively on the GUI side:

- Get parameter CaliFitter from analysis (Doubleclick)
- Display calibration spectrum.
- Double click on the LineFitter fitter in the parameter editor. Fit panel will open showing the current settings of the fitter. Press Use pad of the fit panel to assign this fitter to the view panel containing the calibration spectrum and Rebuild button.
- Use peak finder 3 to find the peaks. Enlarging the noise factor removes peaks as well as minimum noise.
- Do Fit. If the positions of the lines are fitted correctly, copy the fitter back to the calibration parameter: right mouse button click on LineFitter, select Get from FitPanel .
- Check if the name of the calibration file is correct.
- Set DoFit variable to 1 (will be set back to 0 after the fit).
- Now press left arrow button. This will perform fit of the calibration curve (polynomial of order 2) in the *UpdateFrom()* method of TascaCaliFitter on the analysis side.
- Pressing right arrow button will get the results of the calibration, present in the polynomial coefficients fdA[0]...fdA[2] and in the Calibrator fitter.
- The corresponding TGraph is UserObjects/CaliGraph and is displayed by double click. Then double click on the Calibrator fitter in the parameter editor to open in a fit panel, press Use Pad, Rebuild and Draw. This will draw the

calibration polynomial over the points which indicate the energy/channel of the calibration lines.