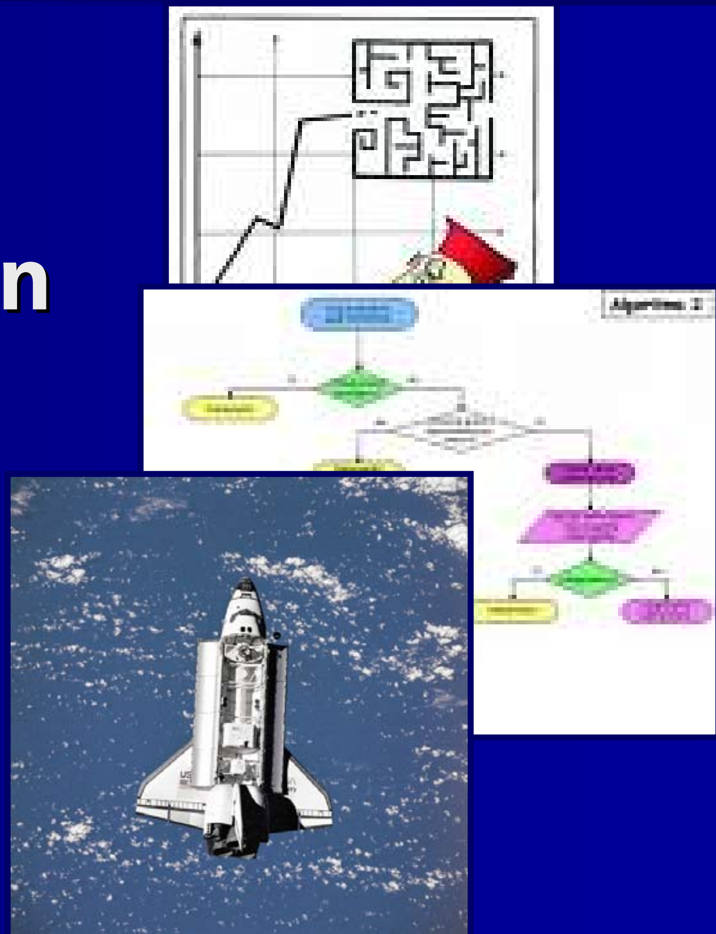
TOF Online Calibration

-Strategy

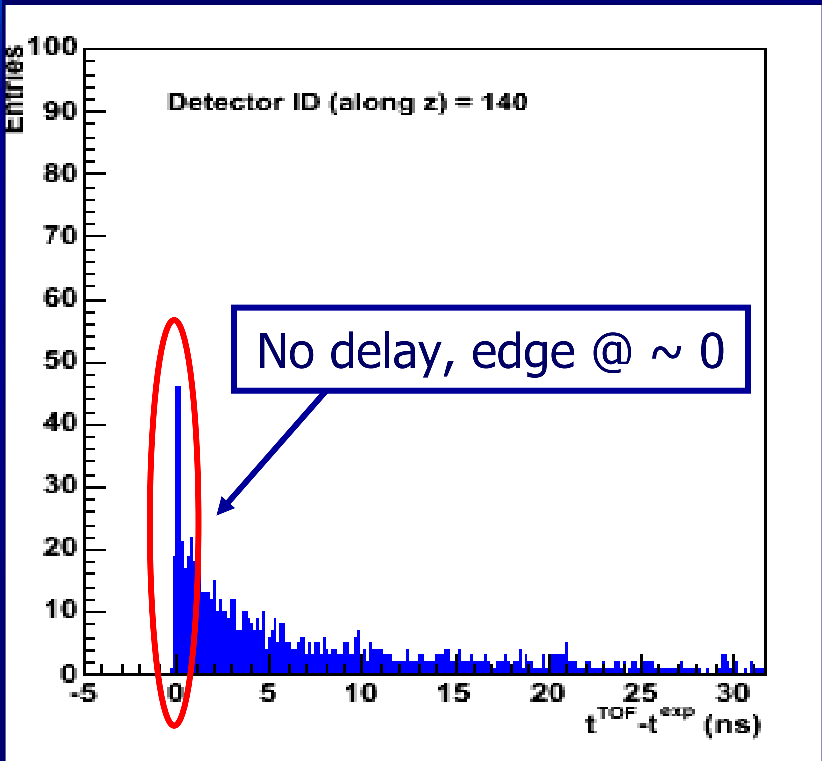
-TOF Detector Algorithm

-TOF Preprocessor



Online Calibration Strategy

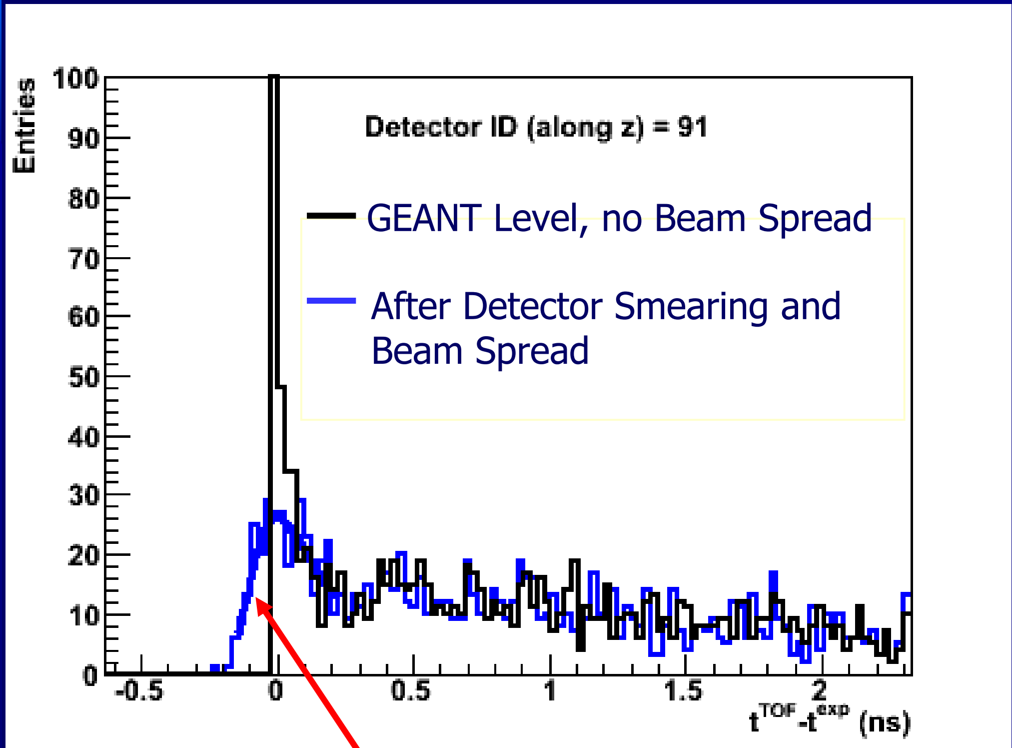
- Determine the relative delays for each TOF channel using the t^{TOF}-t^{exp} spectra



- t^{TOF} = time measured by TOF (T0 subtracted!!)
- t^{exp} = expected time of flight, from TOF geometry, assuming $\beta = 1$ and a straight line trajectory

- *Sharp edge* expected at the delay value from fastest particles ($\beta \sim 1$)

t^{TOF}-t^{exp}: a closer look at the edge



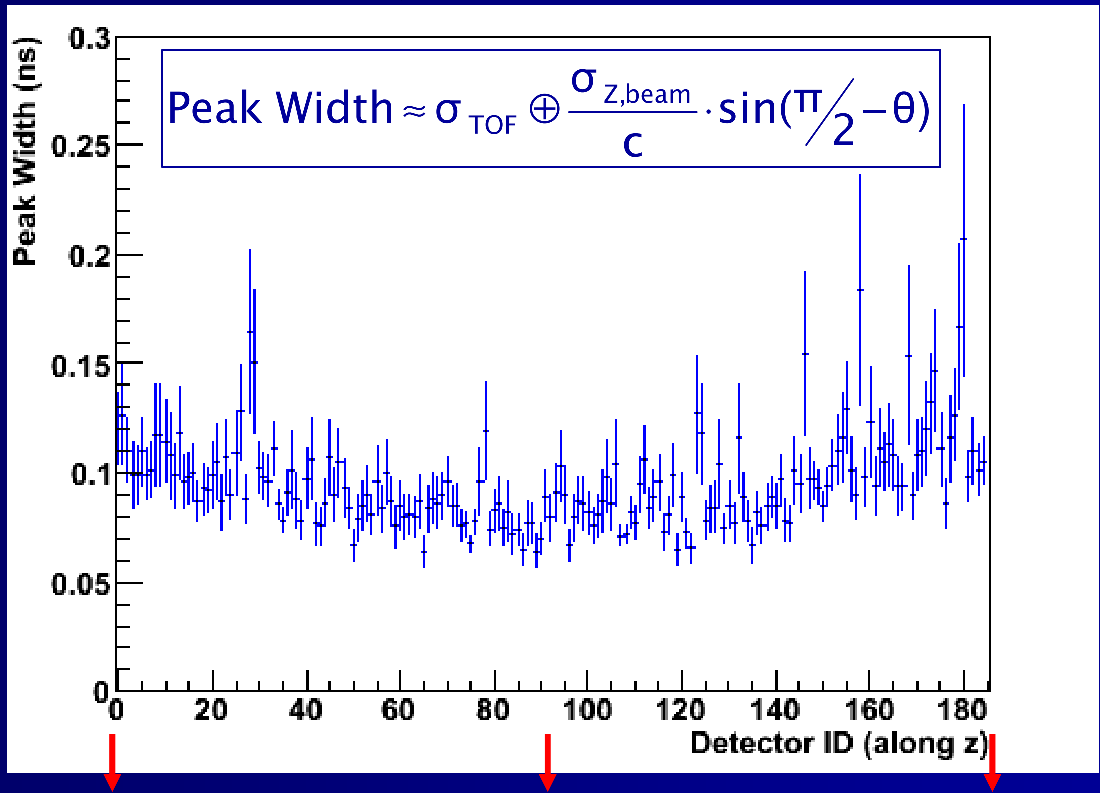
Fitting of the edge with a Landau distribution

At GEANT level the edge is very sharp.

It is then “smeared” by the TOF time resolution and the IP spread along the z direction.

Peak Width

- The Peak Width (averaged over many events) depends on the TOF resolution (σ_{TOF}), on the spread of the interaction vertex along the z direction ($\sigma_{Z,beam}$), and on θ



θ = 45°   θ = 90°   θ = 135°

Measuring the Edge

1. Fitting of the edge (over an appropriate range) with a function, e.g. a Landau.

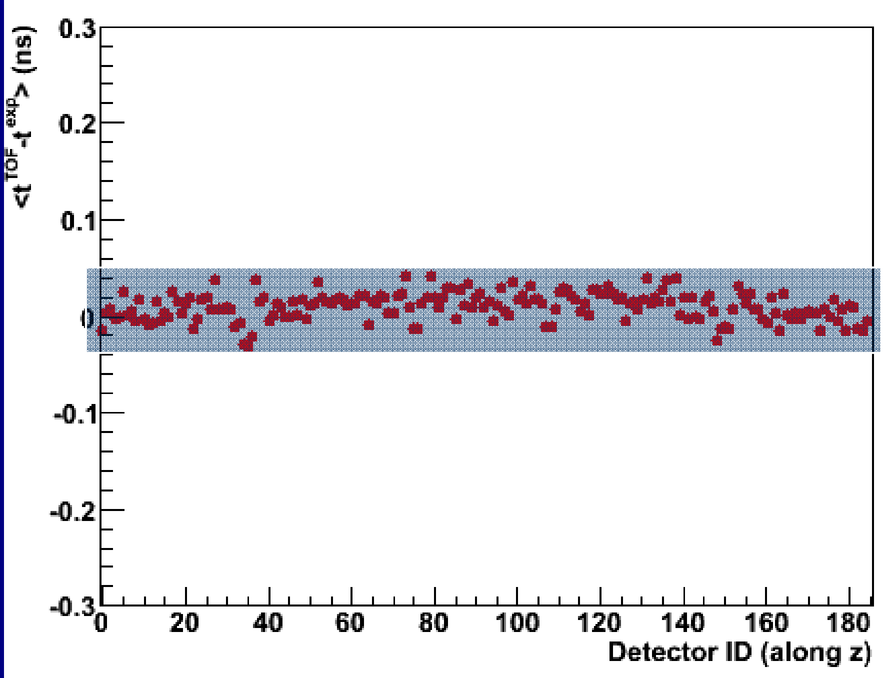
   → may not be robust in case of low statistics!

2. Currently using the mean $<t^{TOF}-t^{exp}>$ over ~ 320 ps (≈ 4 times σ_{TOF}) around the edge: edge $\equiv <t^{TOF}-t^{exp}>$

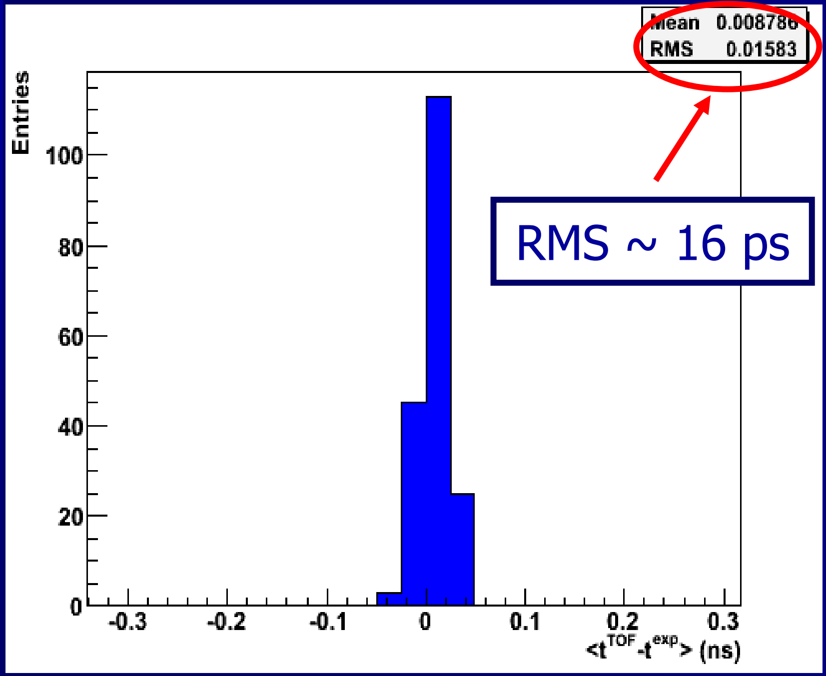
   - the lower limit of the interval is defined by the first four consecutive bins whose total content exceeds the 1% of the total spectrum entries

Determination of the Edge: Results

Edge position vs channel z position (~ 4000 entries per channel)

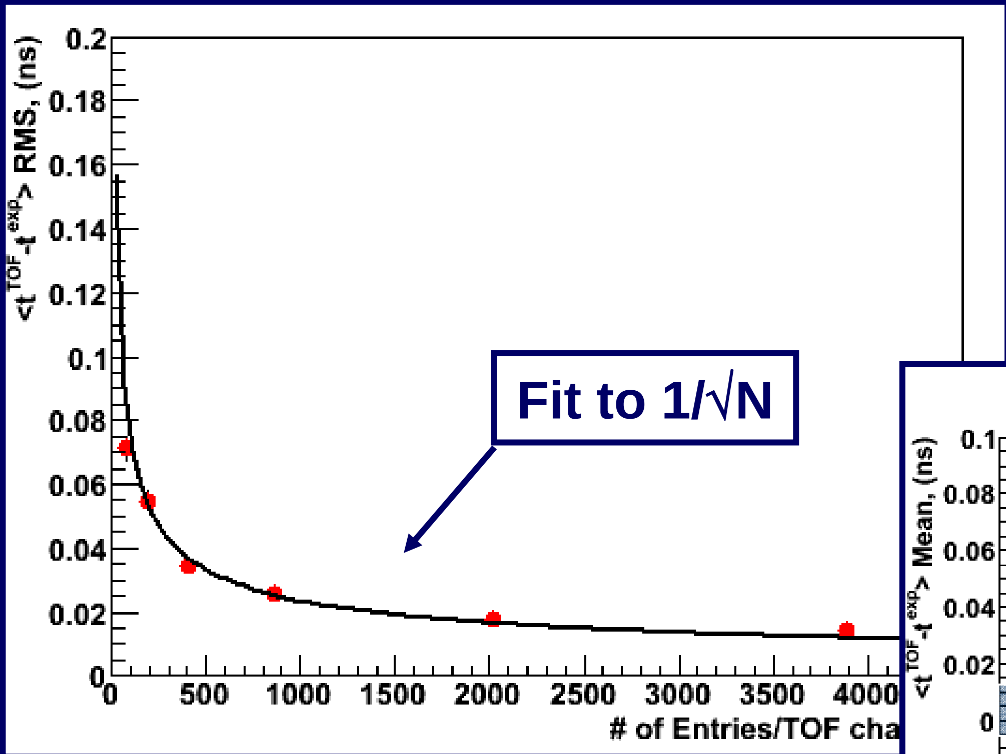


Projection along y-axis (to estimate the uncertainty/channel on the edge determination)

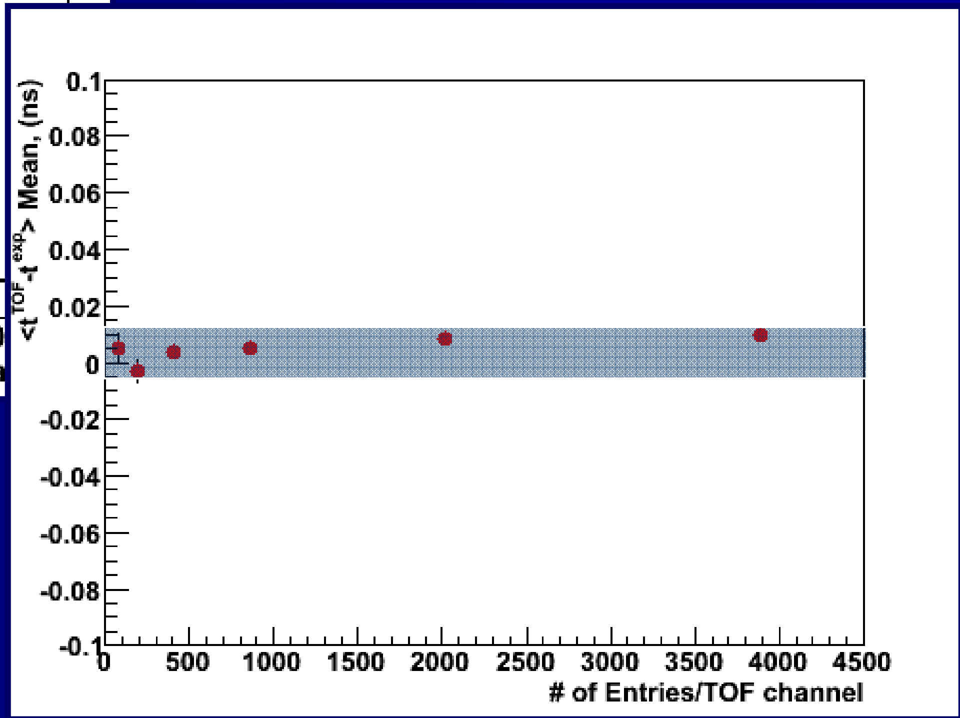


RMS ~ 16 ps

Dependence on the Number of Entries/Channel

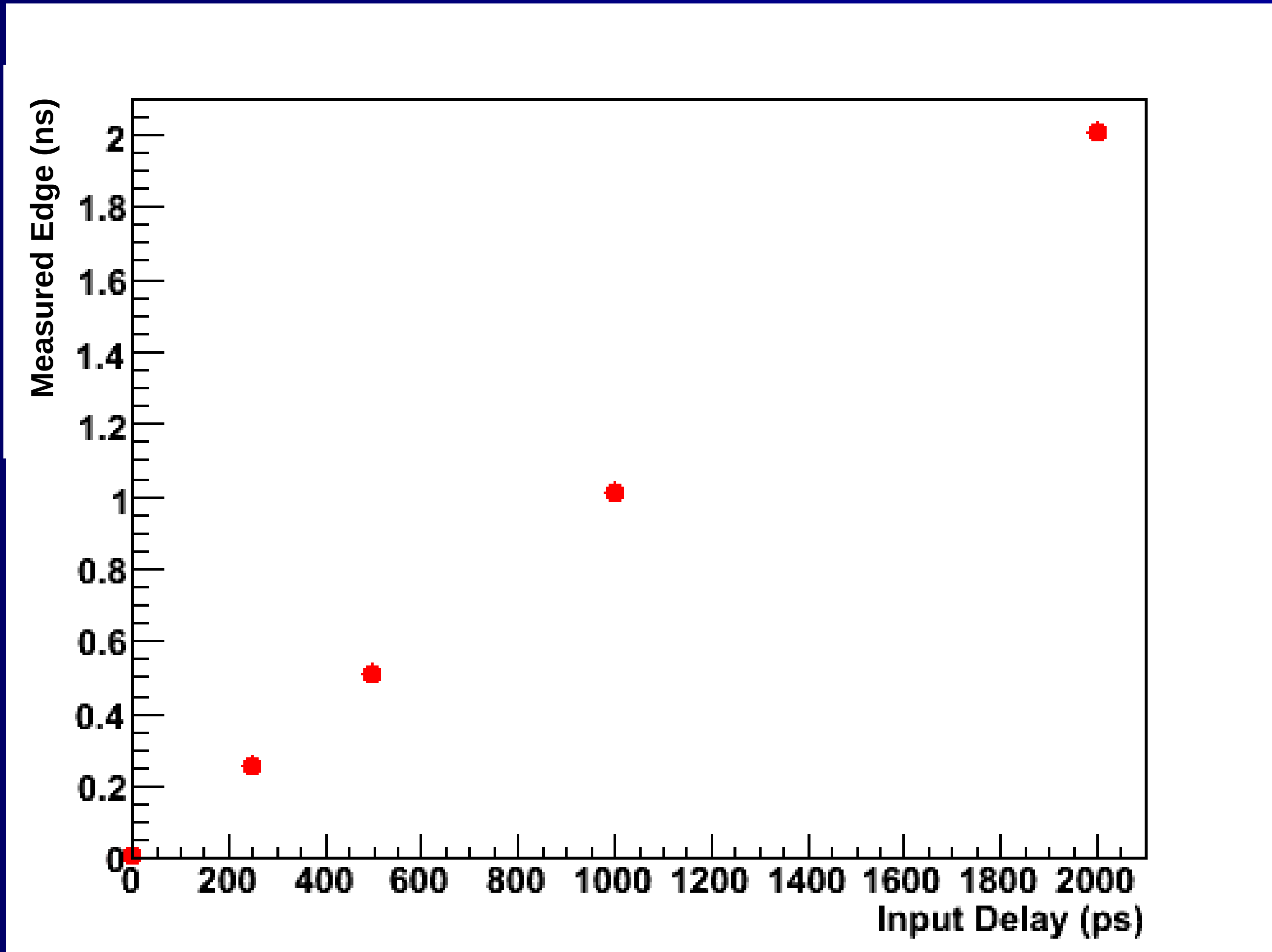


Uncertainty (RMS) on the time edge vs Number of Entries/Channel



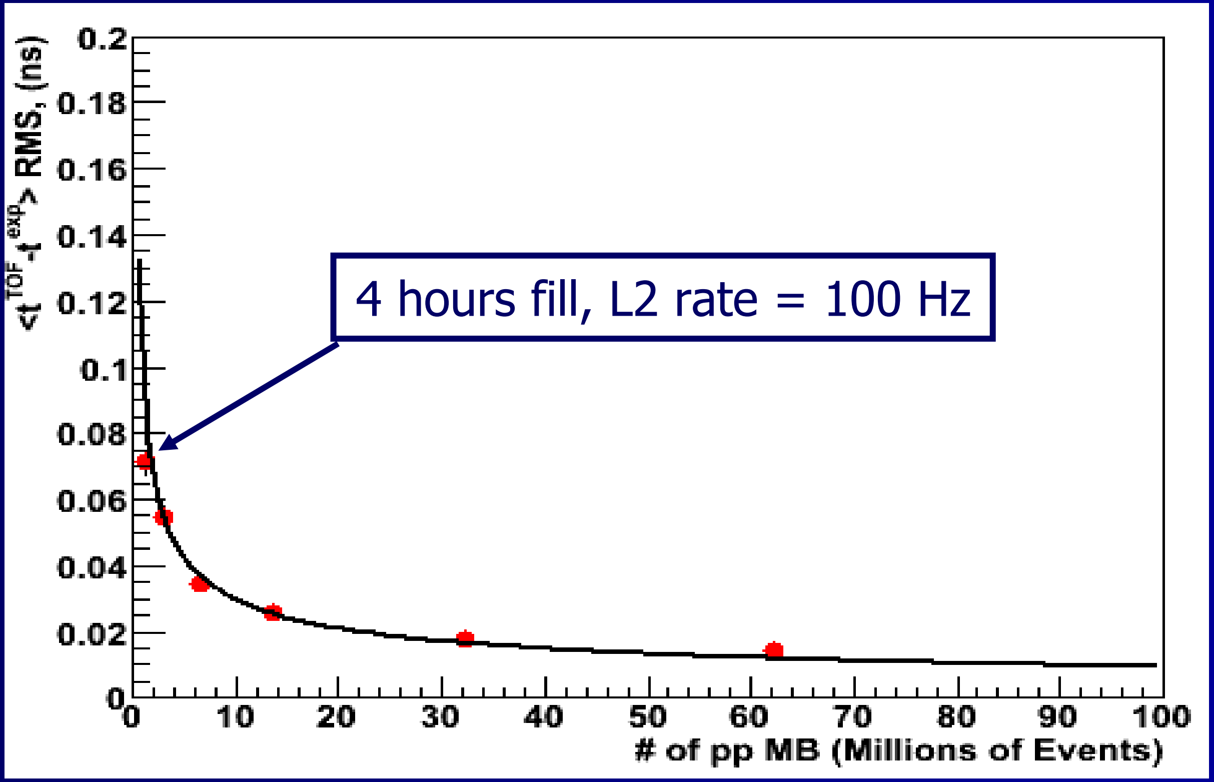
Mean Value of the Time Edge vs Number of Entries/Channel (no delay!). Stable within 10 ps.

Dependence of the Measured Edge on the Input Delay



Dependence on the Number of Events

pp Runs, assuming 10 hits on TOF/MB event



TOF DAQ Detector Algorithm (DA)

- The TOF DA will run on the monitoring machines after the events have been built in the GDCs.
- It will run over whole events, or only on sub-events (i.e. fragments of events – discussed with DAQ team) containing the TOF and the T0 data.
- It will run during physics runs, during data taking.
- The output file will be a persistent file which will be "updated" run by run.
- It will export the output file via the DAQ FES, which will be then processed by the SHUTTLE for calibration.

TOF DA: Software Development

- The TOF DA being developed in DATE.
- The TOF DA will need ROOT libraries.
- At present, the TOF DA framework is stand-alone wrt to AliRoot: dedicated classes are being written reflecting the TOF geometry and RawData decoding.
- Better to use AliRoot libraries.
- Currently using a TObjArray of TH1C – starting point to test the DA.

TOF DA: Test-Benchmarks

- ~ 160000 (TObjArray of TH1C, nbins = 1500):
  - size in memory needed: ~ 400 MB (processing) ÷ ~ 800 MB (when writing to a file, but only at the end of run!)
  - time requirements:
    - Time to access the data not yet estimated, dependent on the type of events (whole events/subevents, type of collisions..).
    - Histo filling time/hit on TOF = ~ few μs
  - FES space usage: ~ 7 MB to store the TObjArray

TOF DA: Test-Benchmarks with TH1

	TH1C	TH1S	TH1I (500 bins)
Memory during processing	~ 500 MB	~ 500 MB	~ 200 MB
Memory during writing	~ 800 MB	~ 2 GB	~ 1.6 GB
Disk space	~ 7 MB	~ 11 MB	2.4 MB
Time during writing on a file	< 10 s	~ 15 s	~ 10 s

– 1 TOF pp (PbPb) event ~ 20(75) kB
– 3h Data Taking, Rate = 100 Hz: ~ 20 GB

TOF DA: work in progress

- Under study the possibility to replace the TObjArray with a TTree with a simple C++ 2D array of chars (short ints/ints) [nbins][160000]:
  - less memory needed, less space usage (< 1 MB ÷ ~ 20 MB), easier access in both write and read mode.

A special thank to Rene Brun...

- nbins (both in the case if TH1 and of the C++ array) determined by the maximum relative difference expected between the channels, strongly dependent on the electronics and the hardware components (cables, lengths of the signal lines on the PCBs…): could be substantially reduced firmware (tentative guess: less than 100 bins).

SHUTTLE: TOF Preprocessor

- A TOF preprocessor has been implemented in the Test Suite provided by the Offline Core, both for:
  - USER CASE 1:
    - The algorithm (see previous slides) finding the edge of the $t^{TOF} - t^{exp}$ distribution has been implemented.
    - ~ 1.2 s to process all the ~ 160000 channels.
    - The output file (~ 300 kB) is written in $ALICE_ROOT/SHUTTLE/TestShuttle/TestCDB/TOF/Calib/OnlineDelay/
  - USER CASE 4:
    - The algorithm gets input data from DCS, and process them.
    - ~ 1.5 s to process all the 10442 channels.
    - The output file (~ 240 kB) is written in $ALICE_ROOT/SHUTTLE/TestShuttle/TestCDB/TOF/Calib/DCSData/
    - Currently the DCS map is simulated as suggested in the Test Suite.
    - Dedicated classes written.

TOF Preprocessor: Reference Data

- We would like to save the ~ 160000 histos from DAQ as the TOF Reference Data set.
- Size of Reference Data considerably less than Raw Data size (few MB wrt 20 GB $\rightarrow$ ~ 0.1%, assuming 3h of pp-MB Runs).
- Stored for the time being in $ALICE_ROOT/TOF/DELAYS/MON/

TOF: The Calibration GUINEA PIG

- Online
- TOF D/ e as well.
- TOF Pr er Case 1
  and Us
- Referer Still
  some s
  on.

Back-Ups