

FD DAQ Software Overview and Status

presented by H.-J. Mathes

18.8.2000

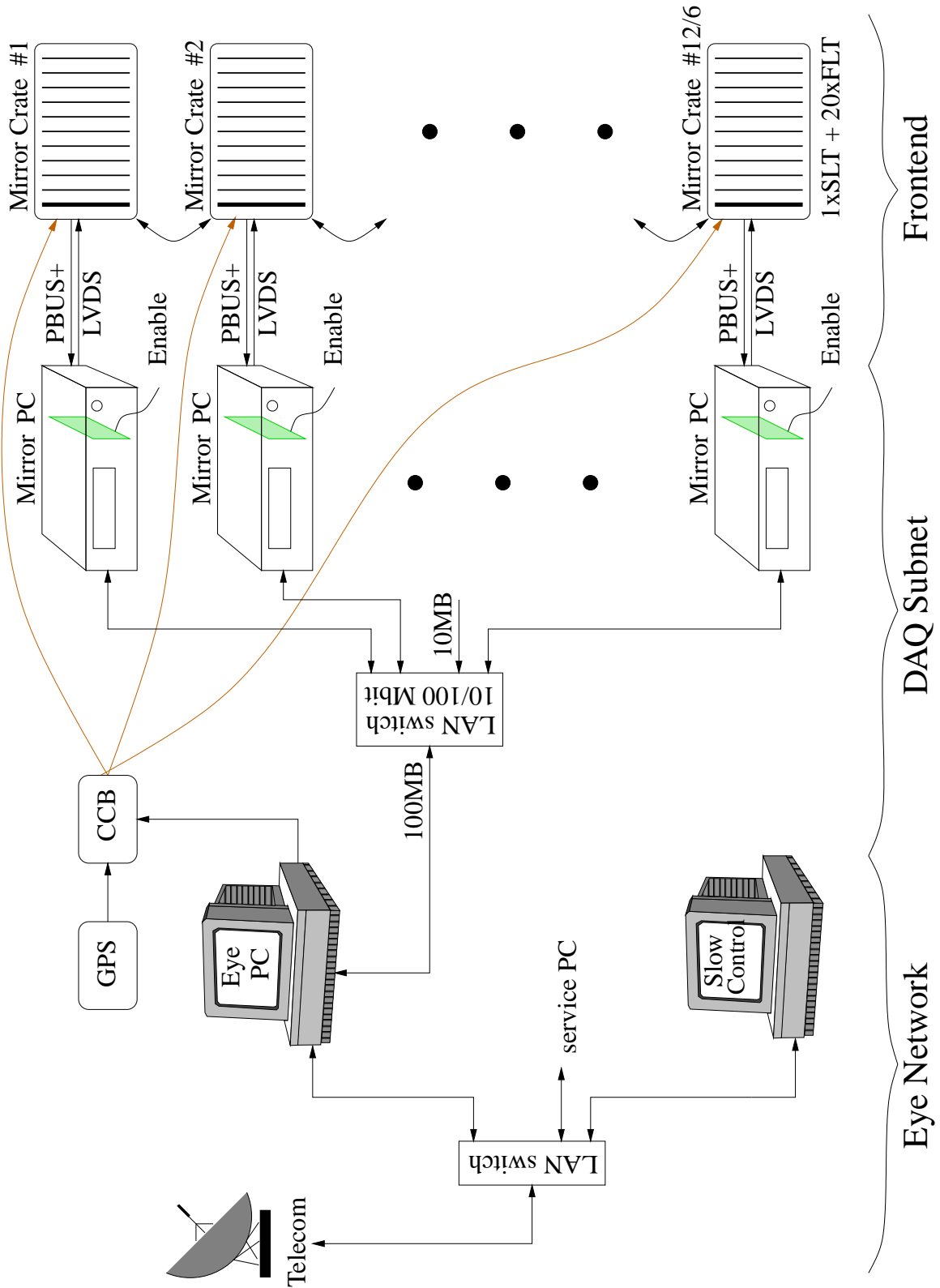
- System Overview
 - requirements
 - system decomposition
- The prototype DAQ system
 - properties and constraints
 - status of different components
- Experiences and performance results
 - designing, coding and testing
 - usage of ROOT
 - first performance measurements

Requirements for FD-DAQ and Control System

- event rates/data throughput
 - trigger rate approx. $1..2 \text{ s}^{-1}$
 - physics rate 1 min^{-1}
 - max. rate for calibration 20 s^{-1}
- requirements from CDAS:
 - T3 trigger decision: 5 s
 - data buffer for 10 s
 - SD-station list
- operational requirements
 - reliable and failsafe
 - remotely controllable
- maintainability
 - long experiment lifetime, high manpower fluctuation
 - documentation
 - approved methods of software generation

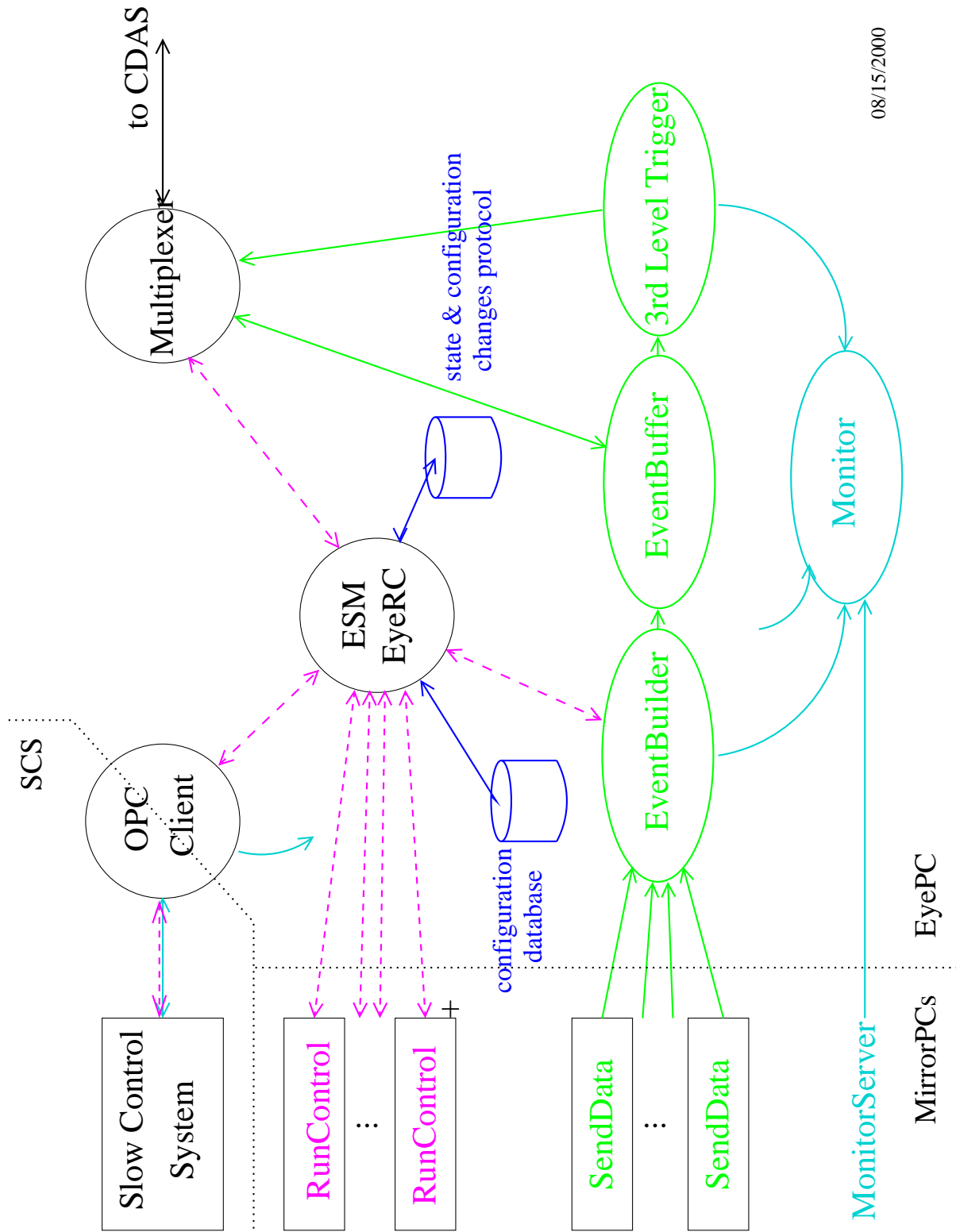
System decomposes into 3 main parts:

- Eye-PC
- 1 ... 6 Mirror-PCs
- SCS → see talk of A. Grindler



Main components of Eye-DAQ

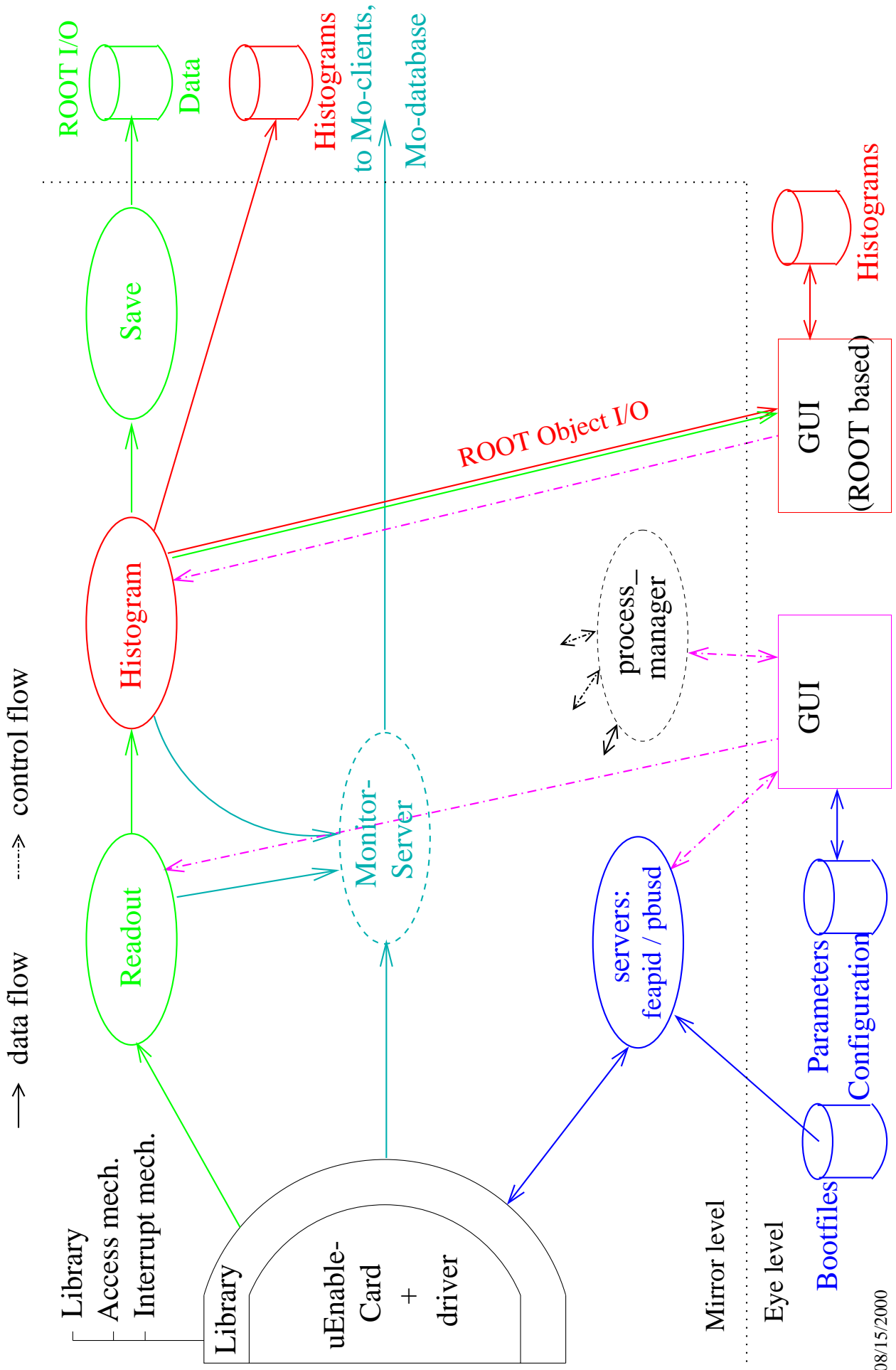
- networking functions
 - gateway: provide access and access control to/for DAQ network
 - boot & file server
- event data and 3rd Level Trigger
 - event building
 - event buffering (10s)
 - 3rd Level Trigger
 - SDP, SD station list,
 - data and trigger transmission to CDAS
- data and parameter storage
 - short term living data
 - temporary calibration data
 - temporary monitoring data
 - configuration variables
 - other experiment parameters
 - program codes
- eye status/state management (together with SCS)
 - ensure operational safety and security of FD
 - automated calibration and DAQ cycles



08/15/2000

Main components of Mirror-DAQ

- libraries for hardware access → Kopmann
- tools to setup the hardware → Kopmann
- tools for various hardware tests → Kopmann, Menchikov
- Readout
- Framework for data transport
- Histogram and Event server
(INFN Rome contribution)
- Monitor Server
- System setup
- Logger daemon (not shown)



08/15/2000

Features of the prototype DAQ

- Prototype DAQ \equiv mainly single mirror based DAQ !!!
- intensive test and use of test and calibration capabilities
- operation partly independent of CDAS
- test of libraries and packages
- support of several design decisions

Readout

Features

- two levels of readout
 - 1st: pixel data (SLT)
 - 2nd: FADC data
- 2nd LT decision
- controlled by user (run control)

Status

- design and coding finished
- 2ndLT only with dummy implementation but interface specification clear
- **test and integration with hardware library required**

Framework for data transport and event format(s)

Background: different "DAQ" scenarios possible on Mirror-PC:

- readout of physics data
- readout of test or calibration events
high rate, (eventually) feedback to hardware needed

Features:

- philosophy: producer and consumers
- in-band messaging:
different buffer types need to be transported:
(DATA, BEGIN_RUN, END_RUN)
⇒ need for more flexibility
BEGIN_CAL, END_CAL, CM_DATA, ...
- examples provided:
consumer, producer, load, save
- transparent creation, mapping and formatting of data structures (TMirrorEvent)
- state machine like behavior for user applications

Status

- **verification and test** with realistic data structure (describing FD mirror event)
- **first mirror event format fixed**
→ TMirrorEvent and TShmEvent classes
- **loading/saving** as ROOT tree is working
ackn. to S.Argiro for support
- follows **FD numbering conventions** (docu. available)
- **documentation in work ...**
- **scheme sufficient for calibration RUNs ?**
- **slow DAQ:** Current Monitor readout

Overall Status of Mirror DAQ

- data transport chain is working
- several integration tests required
 - DAQ will be possible !
- test tools not done (on Linux), but just on the way
- pbusd/server starts soon
 - required for any kind of user interface
- no user interface (GUI) at the moment
- link between state control, SCS, ... not done yet

Histogram and event server

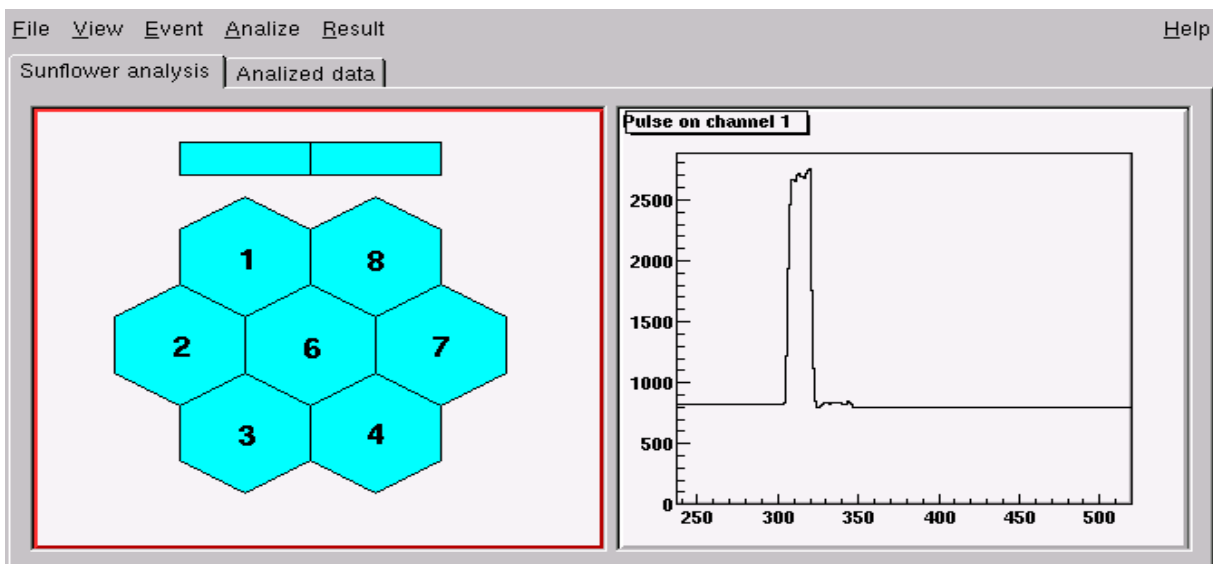
= INFN Rome contribution

Motivation

- analysis of sunflower test data
- monitor tool for 1st measurements in Malargue
- support definition process of FD monitoring during next years operation

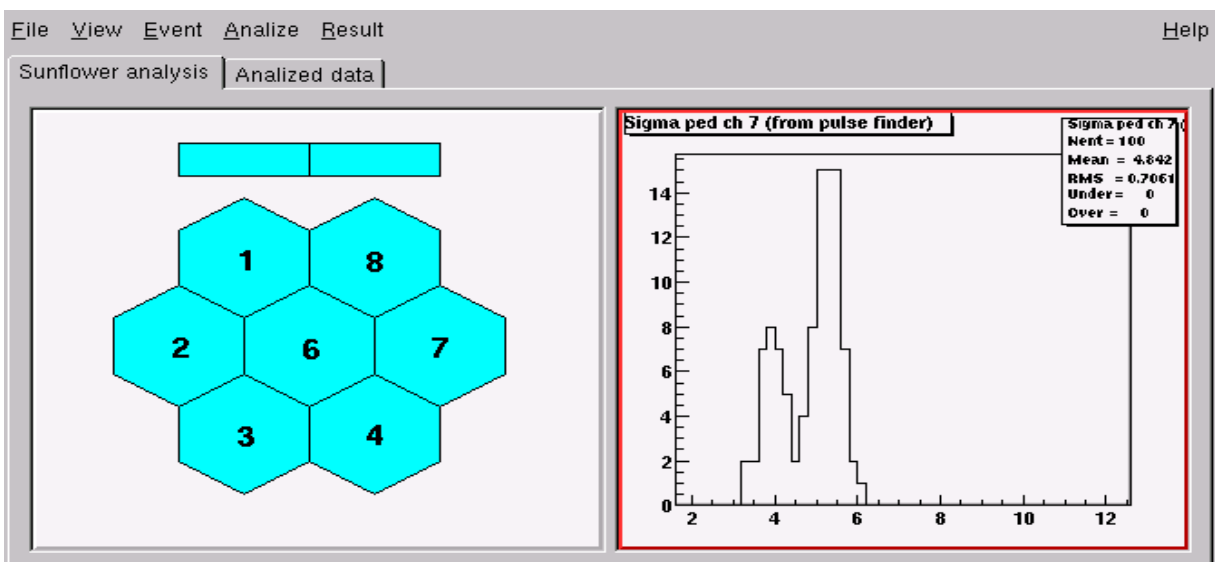
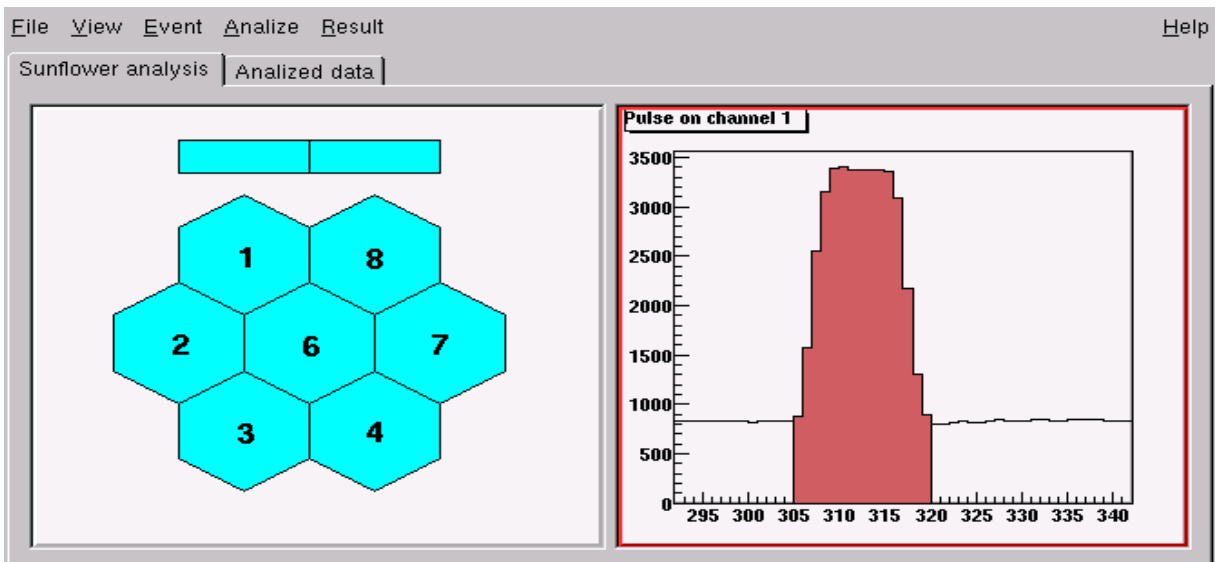
Features

- sitting on event data stream, but cannot crash DAQ
- some rough preliminary analysis
- fills and serves online histograms
→ characterize and monitor the quality of operation and data
- periodic saving of histograms
- acting also as event server
- compiles (and serves) a list of interesting events
- interactively controllable (GUI - ROOT based)



Status

- **monolithic application ...**
→ break down into client (= GUI) and server part
- **used successfully for data analysis of the Sunflower test setup**
- **Analysis and histograms:**
 - pulse finder
→ pulse integral, pulse maximum, pulse length, start time, pedestal, sigma pedestal,
 - pedestals, sigma of pedestals



FD Monitoring

Overview:

- online monitoring possible at eye and/or mirror level
- hardware functional monitoring
- data quality monitoring
- incomplete state:
 - monitoring items not complete
 - margins, references, what to do with it
 - flexibility very much needed !
- definition of monitoring items
 - all FD hardware/analysis groups
 - definition of the items
 - place of their generation
 - how often
 - margins, thresholds, rules
 - storage issues
- collection/transport (→ under discussion with IPN-Orsay)
- displaying (→ IPN-Orsay contribution)
- short/long term storage requirements (→ needs to be discussed)

⇒ **Mirror Monitor Server**

= conceptual study (like "monitoring kit")

- **configurable** via a simple file
- MonitorItem class with abstract interface
 - ⇒ model for the **basic behaviour**
 - periodic checking of data with rules
 - alarm action possible
 - different alarm levels possible
 - other periodic activity programmable
- templates for **rules, margins, ...**
 - different data types possible
- using ROOT classes and I/O
- server and client classes provided
 - **replaced by CORBA ?**

Coding & Documentation → Feedback ?

Experience from recent design and coding phase:

⇒ Feedback from potential user of software libraries and the software system is very essential !!!

- are specification and design complete ?
ex.: is the provided scheme sufficient for a calibration scenario ?
- documentation understandable and complete ?
- support of testing, completeness of testing
- help in debugging:
 - debugging build process
 - debugging and understanding setup and initialization

Usage of ROOT at Mirror DAQ

Observations:

- rather **large executables** (15 - 20 MBytes in memory)
 - well, shared libs are necessary
 - is it possible to skip some features ?
- **bad run-time behavior** (memory consumption)
of non tree I/O ⇒ use ROOT classes only for
 - event I/O via files or sockets
 - histograms (filling and I/O)
 - Monitor Server

Results:

	CPU: P166 100 evts. inline		CPU P500	
			macro	inline
P	15s	6.6 s ⁻¹	47 s ⁻¹	20 s ⁻¹
P + C	30s	3.3 s ⁻¹	20 s ⁻¹	10 s ⁻¹
P + 2C	45s	2.2 s ⁻¹	12 s ⁻¹	7 s ⁻¹

Conclusions:

- high rate shots only for short time
- need to specify which pixels to read out
software decision based on SLT data !
- statistical analysis of pedestals in hardware (V 3.0)
- rearrangement of FADC register map (V 3.0)