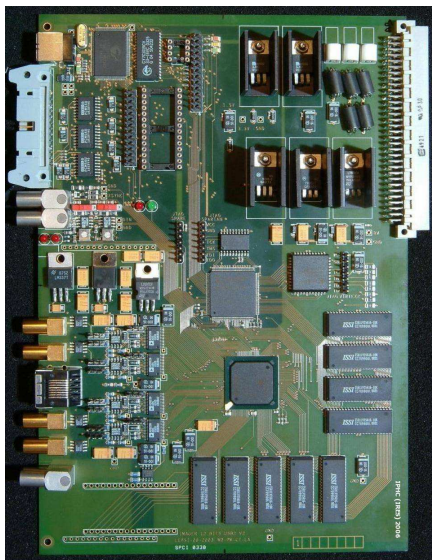


EUDET JRA1 Meeting, Munich October 2006
USB board Firmware & Software Development status

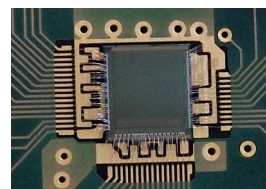


USB Imager Board



OUTLINE

- ▶ USB board firmware validation for EUDET
- ▶ USB board software validation for EUDET
- ▶ IPHC Telescope DAQ upgrade status
- ▶ Analogue signal integrity for MimoTel readout ...



MimoStar 2



USB board firmware upgrade has been done

- ▶ For **EUDET project**
- ▶ For the **upgrade of our beam telescope DAQ from VME to USB**

New features

- ▶ **On board CDS calculation**
 - ▶ **Reduce event size – Increase event rate – Mandatory to reach 40 Events/s with 6 Mi*3M**
- ▶ **Trigger handling**
 - ▶ **Reject trigger on first frame, read one more frame after trigger**
- ▶ **Multiple boards synchronization**
 - ▶ **In order to synchronize 6 MAPS planes readout**

Without these new features the USB board would be useless for EUDET

Status

- ▶ **Implemented on board** : 12 bits signed CDS = Frame 1 – Frame 0 (Hit < 0)
- ▶ **Tested with a marker** injected in analogue input to simulate a hit
- ▶ **Tested with real data** : Mimo*2 + Analysis = same results as with RAW data mode

It works - No problem seen

Availability

- ▶ **DAQ software and board firmware are ready**
- ▶ **Upgrade of data readout libraries** for analysis software (Labview & Mathematica) **done**
- ▶ **Upgrade of Labview analysis software done**

To Do List ... work still to be done

- ▶ **Documentation**
- ▶ A dynamic **switch CDS / RAW** in DAQ SW monitoring to **check base line** level while taking data ...

Status

- ▶ **Implemented** on board firmware – CDS sign handling in DAQ software (use trigger address)
- ▶ **Tested with 2 analogue markers** : Trigger position & Hit simulation (Hit After and Before trigger)
- ▶ We have tried to perform **exhaustive tests** ... **one bug** discovered, **but nothing critical**

It works - No major problem seen

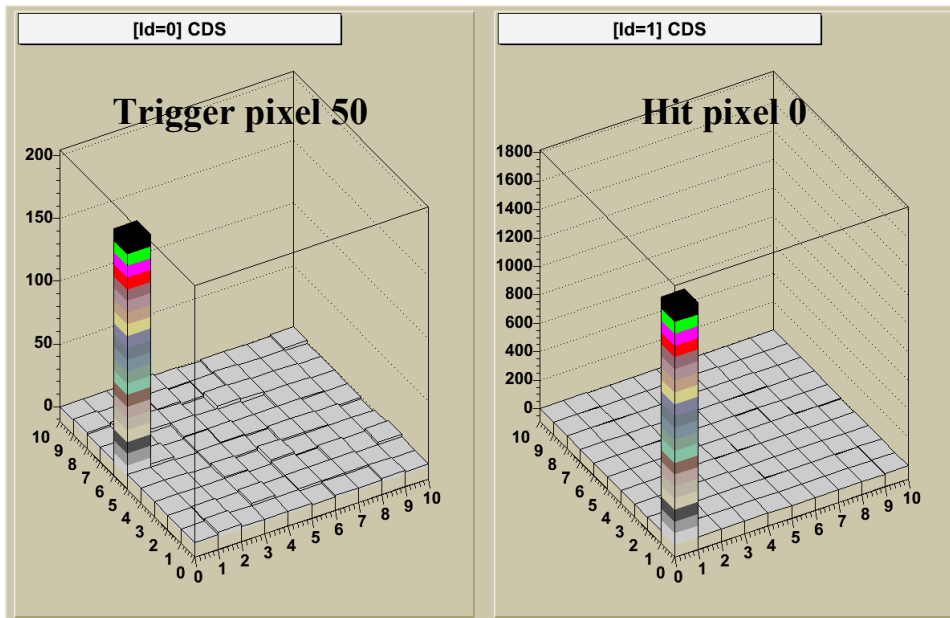
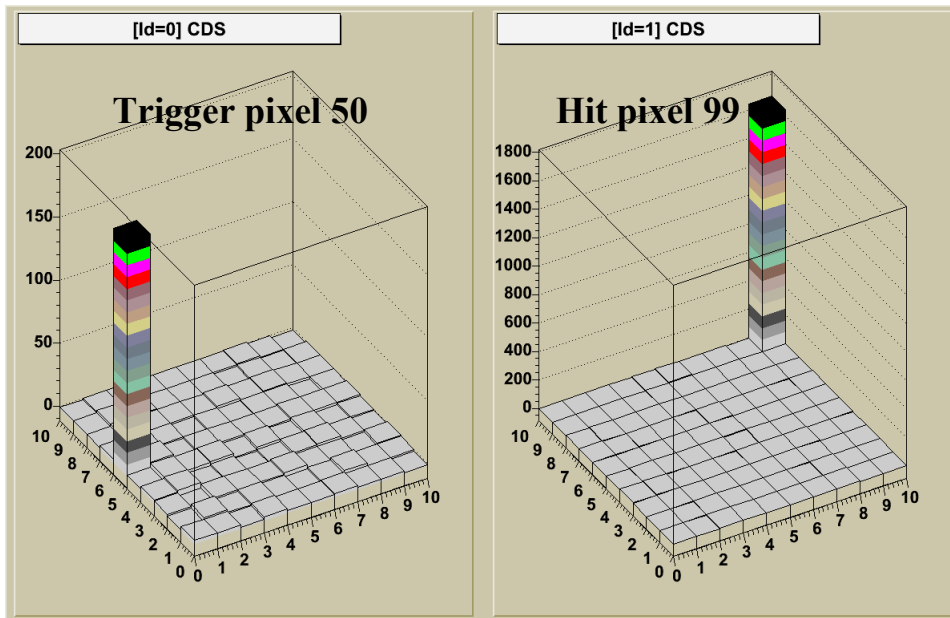
Availability & Limitations

- ▶ **DAQ software & Preliminary** version of **board firmware** are **ready**
- ▶ **Minimum** of **two veto frames** between SYNC and trigger (will increase dead time)
- ▶ Mimosa **matrix size** must be an **even number of pixels** (it should not be a problem)

To Do List ... work still to be done

- ▶ **Documentation**
- ▶ **Fix bug** : **trigger on last pixel not seen** – side effect (**just need time to correct FW and check**)

Firmware validation : Trigger test



Tested with short frame 100 pixels @ 2,5 Mhz

It's easier to check than with 8 k pixels ...

- ▶ Trigger position on left matrix
- ▶ Hit position on right matrix
- ▶ CDS data = Frame (n) – Frame (n-1)

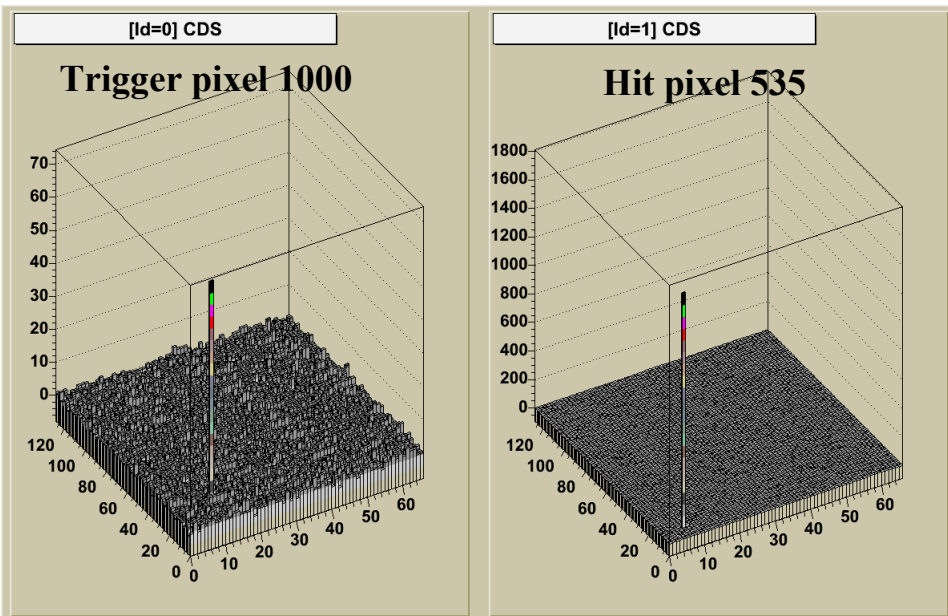
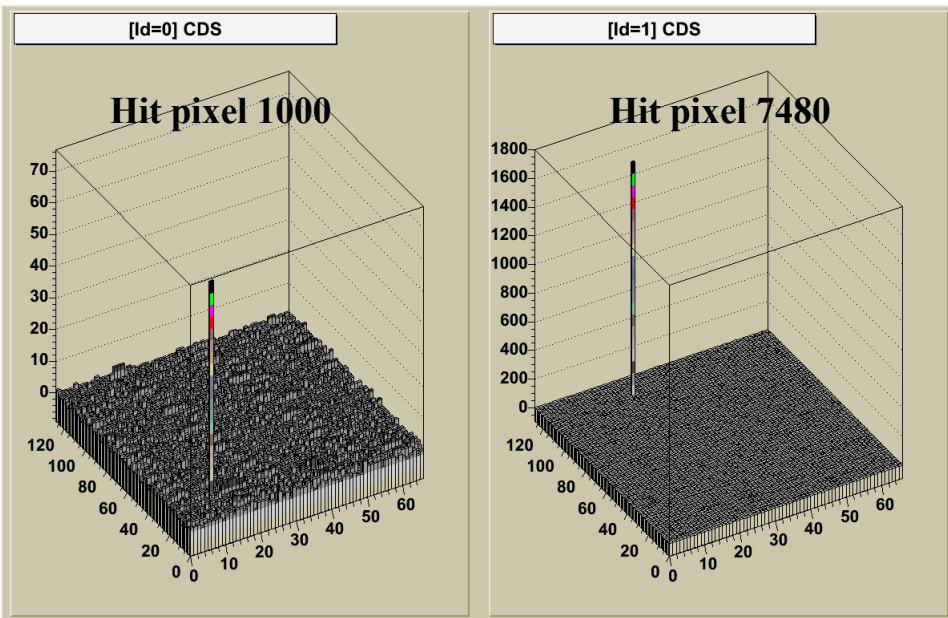
Test done

- ▶ Trigger on Pixel 50 Frame 1
- ▶ Hit after trigger on Pixel 99 Frame 1
- ▶ Hit before trigger on Pixel 0 Frame 1

Result

- ▶ Hit is at the right position
- ▶ Hit sign has been corrected by software
 - ▶ Hit pixel 99 Signal = $Fr(n-1) - Fr(n) > 0$
 - ▶ Hit pixel 0 Signal = $Fr(n-1) - Fr(n) < 0$

Firmware validation : Real world test



Test with $Mi*2$ frame size - 8 k pixels @ 10 Mhz

- ▶ Trigger position on left matrix
- ▶ Hit position on right matrix
- ▶ CDS data = Frame (n) – Frame (n-1)

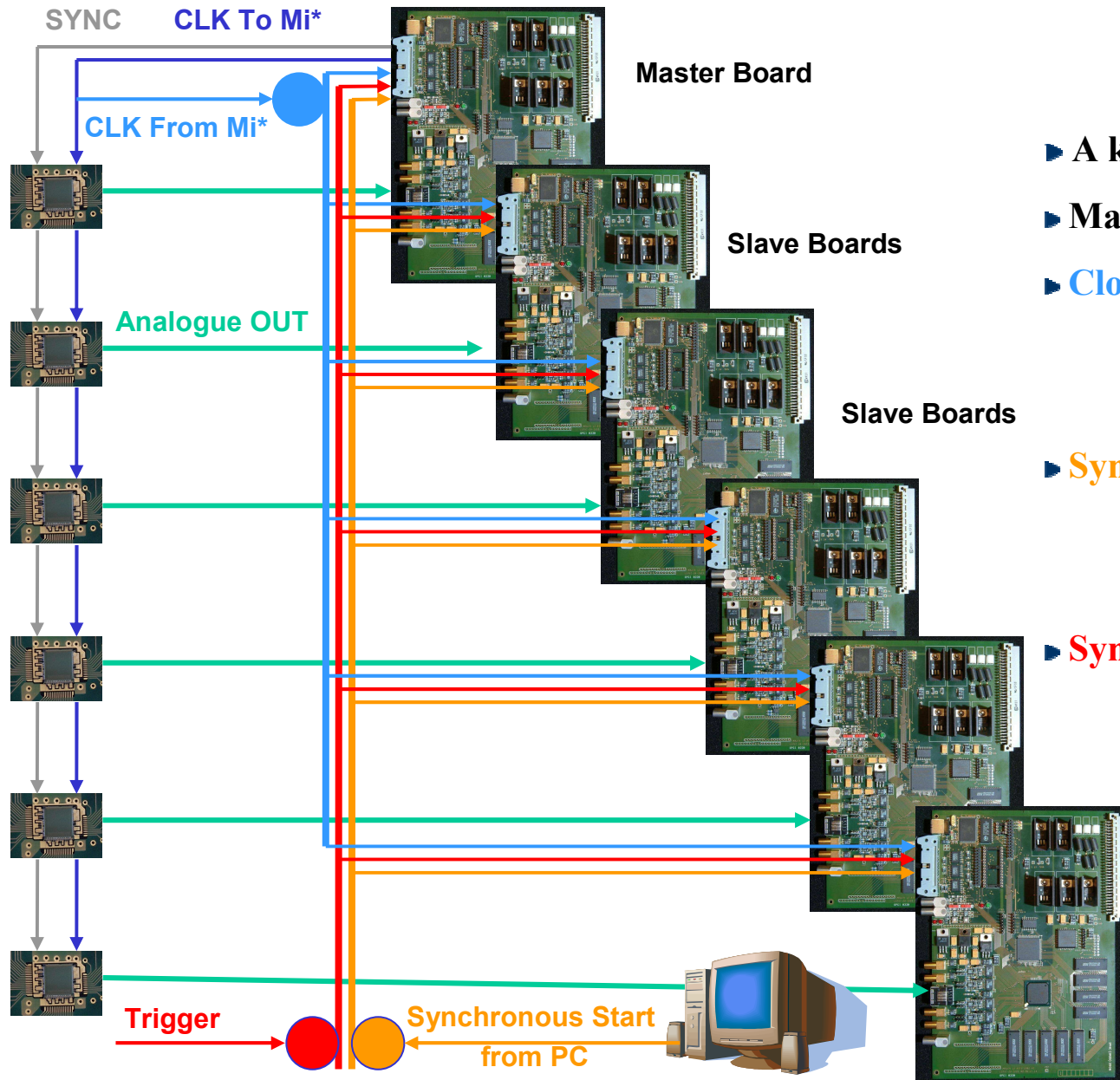
Test done

- ▶ Trigger on Pixel 1000 Frame 1
- ▶ Hit after trigger on Pixel 7480 Frame 1
- ▶ Hit before trigger on Pixel 535 Frame 1




Result

- ▶ Hit is at the **right position**
- ▶ Hit **sign** has been **corrected by software**

Firmware validation : Boards synchronization



The idea

- ▶ A kind of Master / Slave board architecture
- ▶ Master provide **CLK** and **SYNC** to Mimo*
- ▶ Clock go back to DAQ
 - ▶ Distributed to all boards (slaves + master)
 - ▶ Duplicated near the boards 
- ▶ Synchronous start of all boards
 - ▶ From PC parallel port
 - ▶ Duplicated near the boards 
- ▶ Synchronous stop of all boards
 - ▶ From telescope trigger
 - ▶ Duplicated near the boards 

This is not a full master / slave architecture because each board handles trigger

I am not very happy ☹ but it works ☺

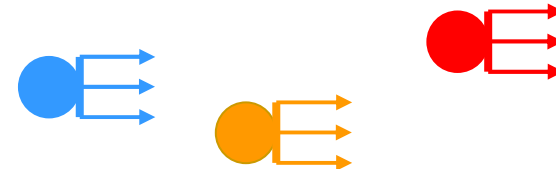
Status

- ▶ **Implemented** on board firmware
- ▶ Test performed with 2 boards (Also tested with 6 boards, but not enough time for exhaustive tests)

It works - No problem seen

Availability

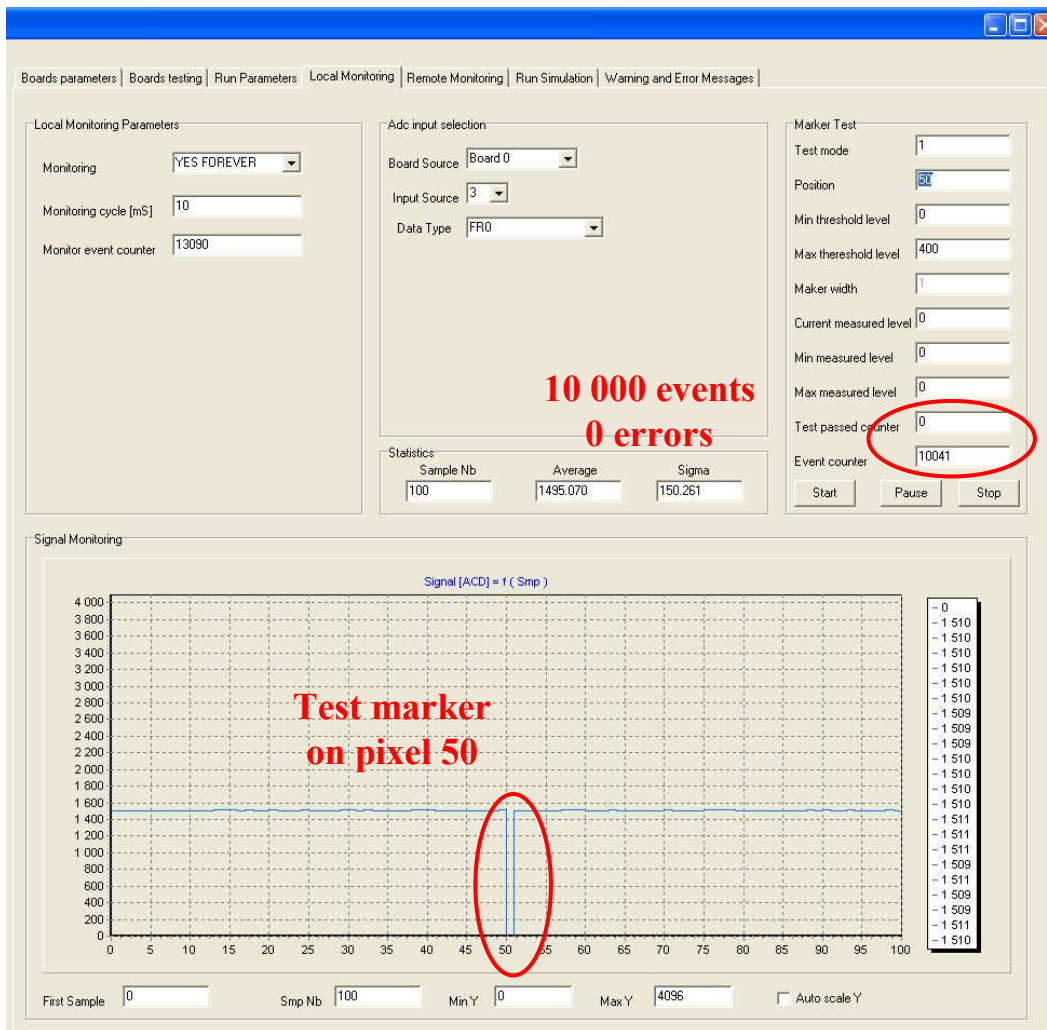
- ▶ **DAQ software & board firmware** are ready
- ▶ **“Special HW”** is **required** to duplicate, synchronize signals ...



To Do List ... work still to be done

- ▶ **Documentation**
- ▶ **Preliminary version** of “Special HW” boards **done by W DULINSKI**

Firmware validation : Boards synchronization test



Test : 2 boards, 100 pixels frame @ 10 MHz

We use “Marker Test” feature of DAQ GUI

- ▶ The same marker is sent to two boards
- ▶ It MUST be at pixel 50 position
- ▶ We count when marker is at a different position

Result

- ▶ More than 10 000 events without error
- ▶ No error :
 - ▶ The marker is at right position
 - ▶ The same position for the two boards

Firmware validation : Trigger handling with N boards

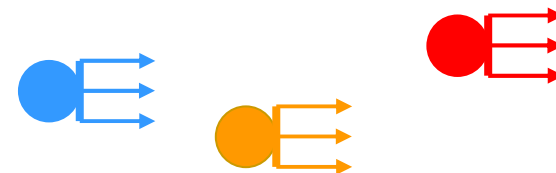
Status

- ▶ Nothing to implement on board firmware : duplicate trigger signal (*synchronized to CLK*)
- ▶ Tested with 2 boards (Also tested with 6 boards, but not enough time for exhaustive tests)

It works - No problem seen

Availability

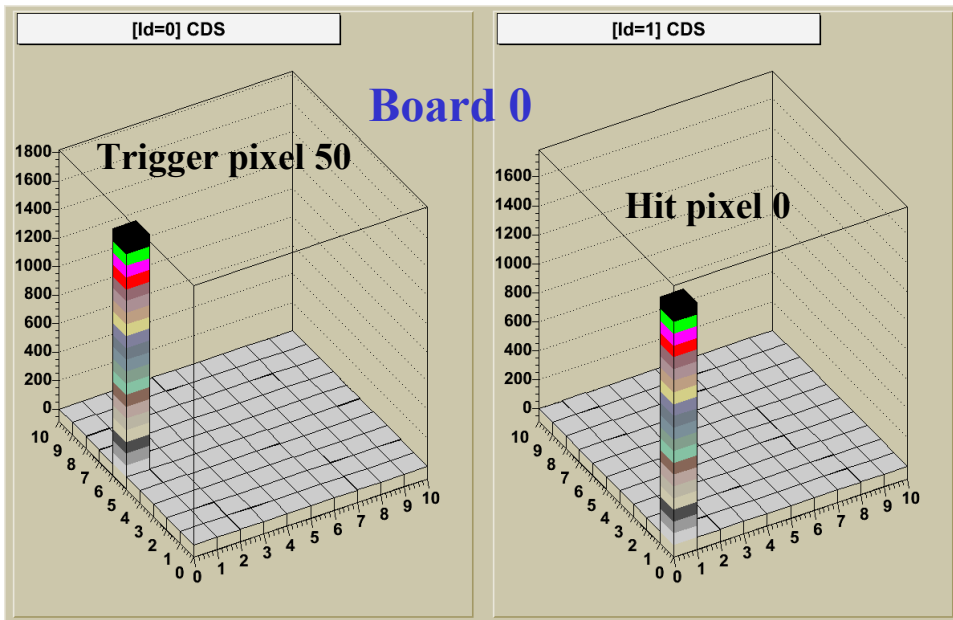
- ▶ DAQ software & board firmware are ready
- ▶ “Special HW” is required to duplicate, synchronize signals ...



To Do List ... work still to be done

- ▶ Documentation
- ▶ Preliminary version of “Special HW” boards done by W DULINSKI

Firmware validation : Trigger handling with N boards

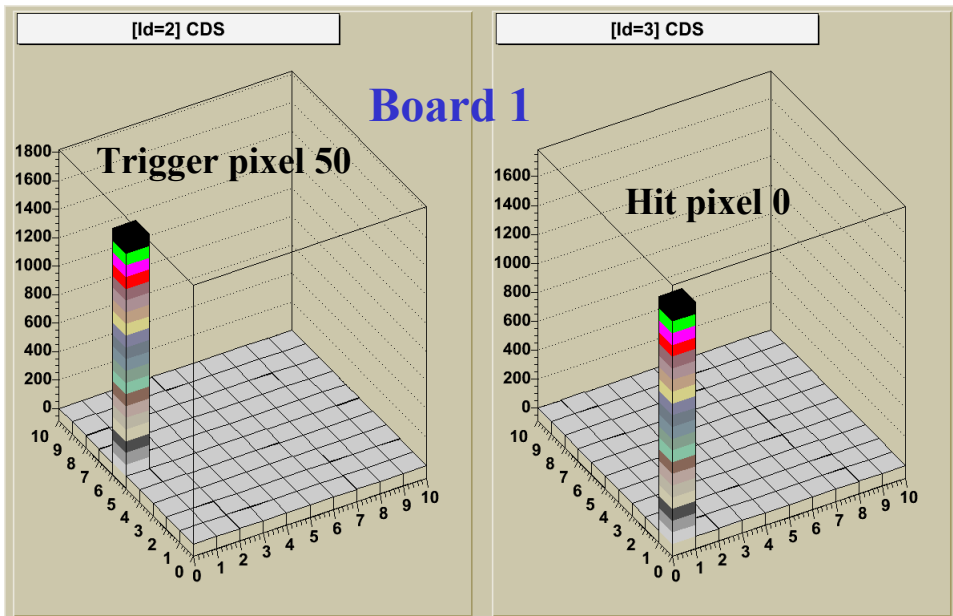


Tested with 2 boards 100 pixels frame

- ▶ The same trigger & hit marker are sent to two boards
- ▶ Trigger position on left matrix
- ▶ Hit position on right matrix
- ▶ Top picture = Board 0 / Bottom picture = Board 1

Result

- ▶ Trigger and hit are at the same position on two boards



The USB board firmware is ready for EUDET demonstrator

- ▶ CDS on board : Ready & Tested
- ▶ Trigger handling : Ready & Tested
- ▶ Multiple boards readout + Trigger handling : Ready & Tested

We have tried to perform exhaustive tests and we have discovered no problem.
But we can't say it's 100 % bug free, we only have done our best in the short time scale we have.

Availability ?

- ▶ CDS on board can be useful for Mi*3M calibrations
 - ▶ Firmware + DAQ Software + LabView Analysis + Doc => End of January 2007
- ▶ Trigger handling for DAQ integration
 - ▶ Very difficult to provide something before January 2007 (I am also involved on others projects ...)

How to do boards firmware upgrade ?

- ▶ Send boards to Strasbourg at beginning of January 2007 ?

The goal


► Integration test with

- **New board firmware** (CDS, Trigger, Multiples boards synchronization)
- **USB multi-threading readout of 6 boards** (Demonstration of March 2006)
- **IRQ handling** with PC parallel port

► Why ?

- To **check** if the **whole architecture** we have proposed **is working**
- To **measure** maximal **event rate** in real conditions (should be close to 40 Hz)
- **This test will help to avoid bad surprises in final demonstrator DAQ integration**

► Results

- It has been done : Simulation of 6 Mi*3 readout, CDS on board, Trigger handling
- Event rate is **~ 30 Hz** : It's **25 % lower** than **simulation of March 2006** (40 Hz) because :
 - Trigger is rejected during **2 first veto frames** and **not only on first frame** (due to current board FW limitation)
 - For trigger mode : **CDS sign handling** requires **1 more access per board** to read trigger address  cost time

Performances improvement

▶ Minimum veto frames number

- ▶ Requires a FW upgrade to allow trigger rejection only on first frame
- ▶ It will be **done for the demonstrator**

▶ How to get back the time we loose to read trigger address ?

- ▶ **Fast** board control mode **will help**
- ▶ It uses // port signals instead of USB access to restart board after each event
- ▶ It's **developed on FW and SW** sides, it will be **ready for demonstrator**

▶ Performances evaluation with fast board control

- ▶ Trigger happens at the beginning of first frame after veto => close to 40 Hz
- ▶ Trigger happens at the end of first frame after veto => 37 Hz
- ▶ Therefore **realistic event rate** will be **35 – 40 Hz** for 6 MimoTel planes (Readout at 10 MHz)

▶ Performance evaluation of Mimosa 18 (Imager 512 x 512 pixels) DAQ

- ▶ We **can't use the same DAQ** for **MimoTel** planes and **Mimosa 18** => 2 DAQ as proposed in April 2006
- ▶ Mimosa 18 DAQ will be the master (slowest one) and MimoTel DAQ the slave
- ▶ Event rate ~ **12/15 Hz - 1 USB board/plane** ... ~ **18/24 Hz - 2 boards** ... ~ **20/30 Hz - 4 boards** (Clock **10/20 Mhz**)

Development status of our beam telescope DAQ

- ▶ Hybrid system (first step) : Windows USB DAQ / Linux Monitoring
 - ▶ **Virtual mode** implemented in Windows DAQ => Can **run without boards**
 - ▶ **Run simulation** implemented in Windows DAQ => Can **replay an existing run**

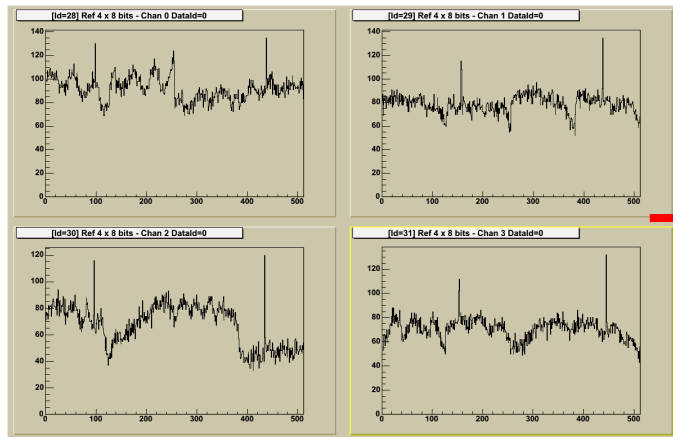
This is very useful tools for whole DAQ chain development and debugging

- ▶ **Interface to Linux monitoring for MAPS and Silicon Strips detectors (with RMP) is ready**
- ▶ **Silicon Strips detectors control HW will be done in February 2007 (2 weeks)**
- ▶ **Conclusion**
 - ▶ **Our telescope DAQ – for fix event size - IS “ virtually ” ready**
 - ▶ We will have **DAQ application** to control 6 boards + **JTAG application**
 - ▶ We will maintain and upgrade this software to follow USB board firmware upgrades

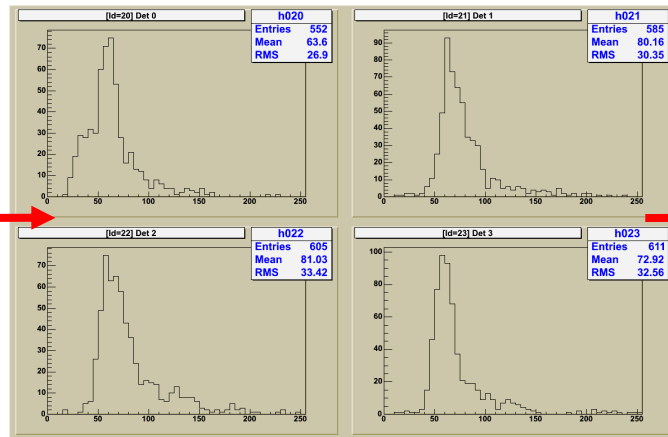
IPHC Telescope DAQ upgrade status

DAQ simulation – Si-Strips detector + 2 Mi*2 = Real software – Without hardware

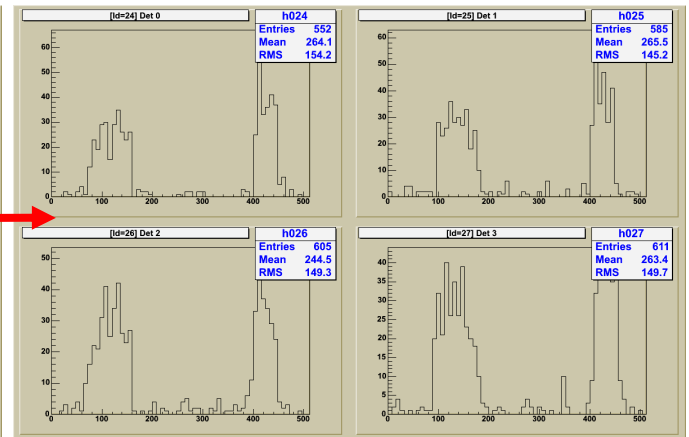
4 planes - Si-Strips on Linux Monitoring



4 planes Si-Strips - Landau

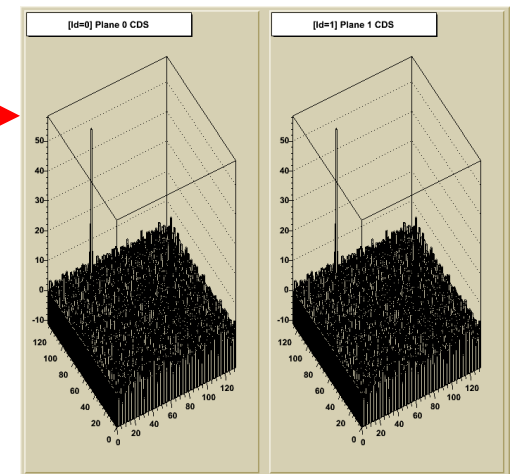
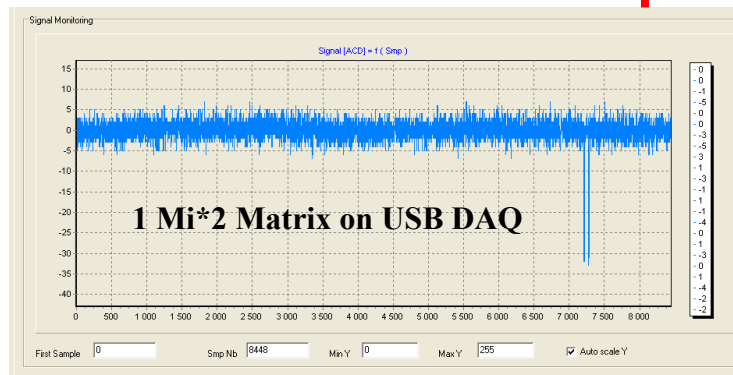
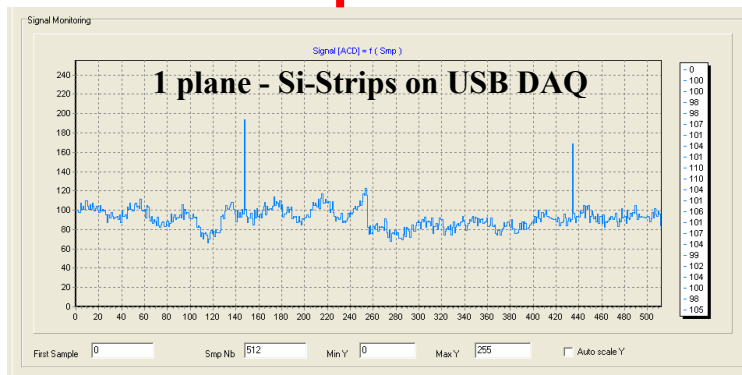


4 planes Si-Strips – Beam Profile



Software “chain” from USB DAQ to Linux Monitoring is ready for Si-Strips & Mimostar

We have DAQ & Monitoring to test Si-Strip + 2 Mi*



2 Mi*2 Plane on Linux Monitoring

What does it mean ?

- ▶ **Windows DAQ** for 6 MAPS plane **is developed**
- ▶ **Exhaustive tests** will be done at **IPHC**
- ▶ We have the **tools** (**Monitoring**) to **integrate it in our Telescope**
- ▶ A **Beam Test** will be possible with **Si-Strip and 2 Mi*** before June (**Spring 2007 ?**)

What can we share with EUDET ?

How can we help EUDET and How EUDET can help us ?

- **Is DAQ software development on both sides** (**IPHC – EUDET**) a **good idea** ?
- It will **consume** a lot of **time** at **each upgrade** (**Documentation, misunderstanding, bug fixing ...**)
- In any case **IPHC will have** to provide a **new driver** at each **board FW upgrade ...**
- Is there a **better idea** – **to save time** – **than the SDK** I proposed in March 2006 ?
- May be **splitting Tasks** (on SW and Manpower point of view) with a **clear interface** can help ?

IPHC Telescope DAQ upgrade status

EUDET Master

EUDET MAPS DAQ Master Application

Boards Configuration: MIMOSTAR2_10MHZ

Run Directory: c:\data\misid_2

Run Number: 14666

Events Number: 100000

Events number / file: 10000

Events Counter: 0

Output Data:
 EUDET (Bohn SW)
 IPHC (DISK - RMP)

Start Run Stop Run

Status: DAQ application stopped - Wait a start request

One Idea ?

- ▶ Provide a remote control of IPHC DAQ
 - ▶ Limited set of parameters under remote control
 - ▶ Master application : Start, Stop, Status
- ▶ The advantages
 - ▶ Reduced commands set (Easy to use)
 - ▶ Specify configuration
 - ▶ Start, Stop Run, get Status
 - ▶ Don't need to handle boards parameters files
 - ▶ Modify and store parameters from IPHC GUI
 - ▶ No need to upgrade GUI to follow board FW
 - ▶ Bugs will be quickly fixed because we use this SW
- ▶ We can apply the same idea for JTAG SW

CMOS USB DAQ V3.1

Boards parameters | Boards testing | Run Parameters | Local Monitoring | Remote Monitoring | Warning and Error Messages

Board 0	Board 1	Board 2	Board 3	Board 4	Board 5
Board Status: RUN	ABSENT	?	?	?	?
Board Type: VFAS	VFAS	?	?	?	?
Selected Board: 0					

ADC USB V2

General Parameters:
Board Type: VFAS
Board Address: 0
Board State: RUN

Digital Parameters:
Samples / Pixels Nb: 8448
Samples Offset: 0
Top Addr: 8447
Clock Divider: 2 to 30,000 kHz
Clock Delay: 5
Spare 1: 0

Analog Parameters:
Chan Width: 16V, CDS: 1E
Read Mode: PROM BEAMS

Board Status:
 Stop Done Pixel Count: 0
 Running Wait

Sequencer Status:
 Stop Post Reset
 Wait Run
 Reset Sync Done

Special Settings:
 Turbo

Diac control:
Dac 1: 0 Dac 3: 0
Dac 2: 0 Dac 4: 0

Save Current Conf

IPHC Slave DAQ "engine" for MAPS

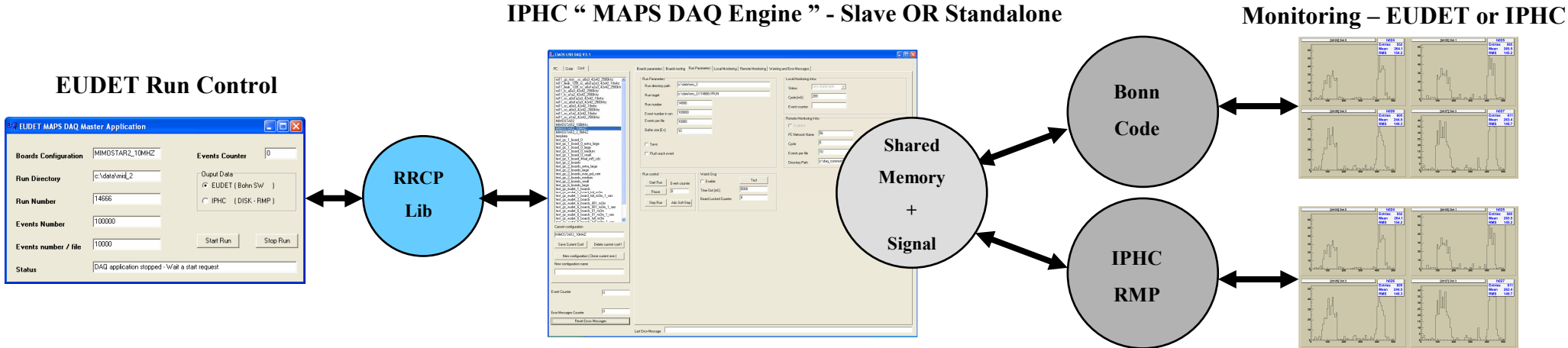
IPHC Telescope DAQ upgrade status

How to do it ?

- ▶ **RRCP** (Remote Run Control Protocol) and **RMP** (Remote Monitoring Protocol) libraries
- ▶ **RRCP and RMP libraries can allow**
 - ▶ A **common interface** for “ **MAPS DAQ engine** ” to **EUDET Demonstrator** and **IPHC Telescope**
 - ▶ **Upper layer** provides **generic functions**
 - ▶ **Lower layer implementation** can be done in a different way for **EUDET / IPHC**
 - ▶ **Upgrades** to improve **lower layer performances** can be done **without consequences on top application**

Advantages

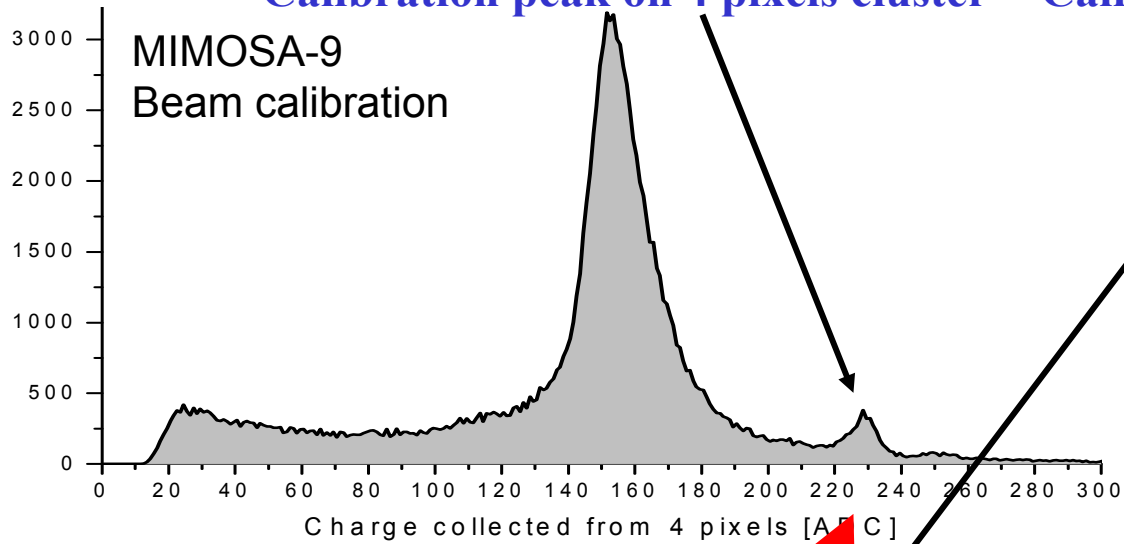
- ▶ We can quickly **Use and Test** the “ **MAPS DAQ Engine** ” in our **Telescope**
- ▶ **Global SW** (Run Control, Monitoring) is **not linked** - we are free on both side : **EUDET / IPHC**



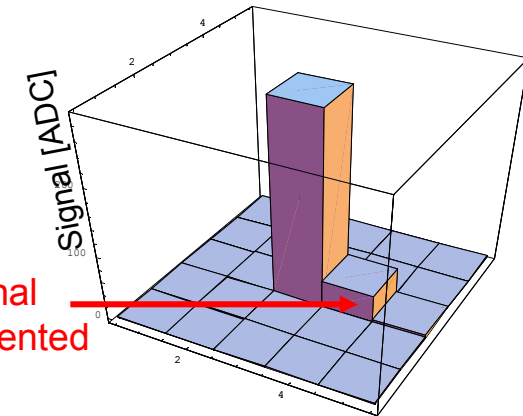
Analogue signal integrity for MimoTel readout – The problem

We have discovered a very strange result (Mimosas 5, 9, 11 ...)

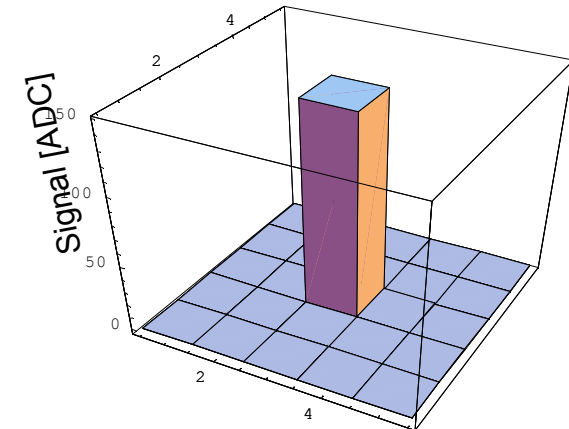
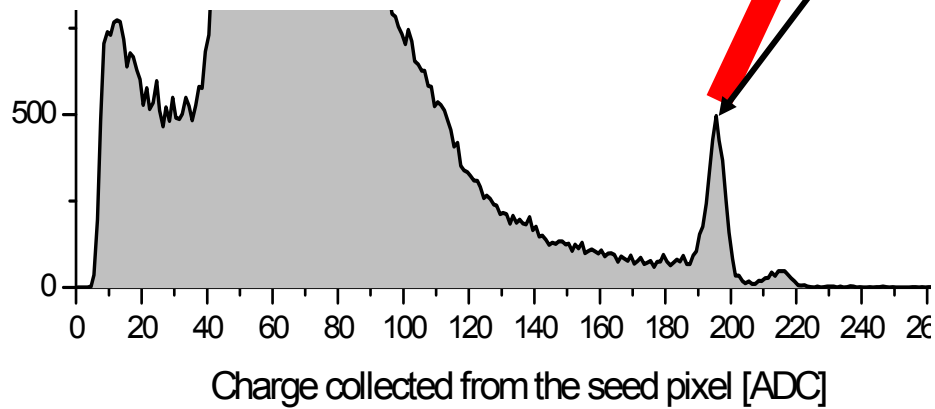
Calibration peak on 4 pixels cluster = Calibration peak seed pixel + **15%**



Here additional charge is invented



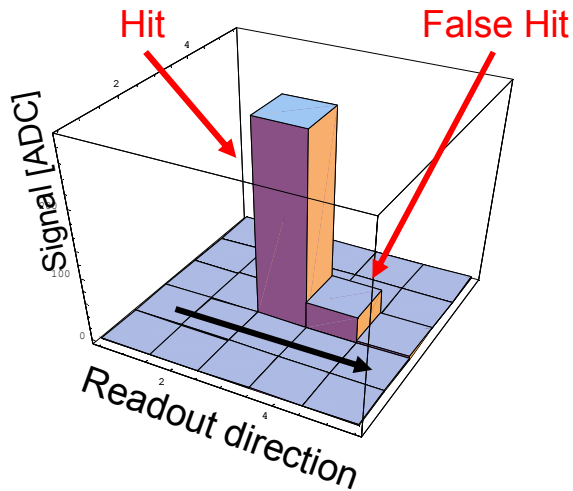
Displacement
(~ 240 e, ~ 15 %)



That's how things should look like

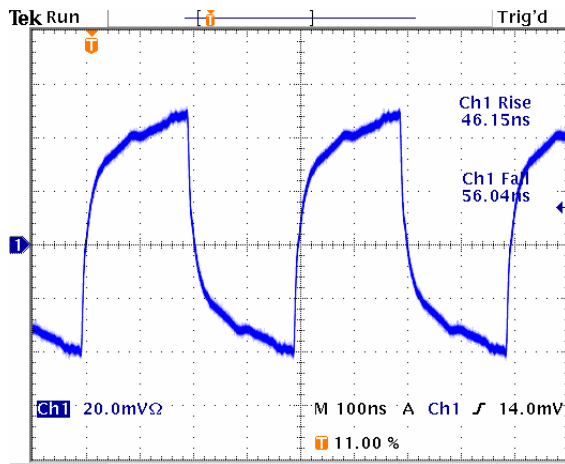
Slide borrowed from Michael Deveaux presentation

Analogue signal integrity for MimoTel readout – Preliminary Ideas

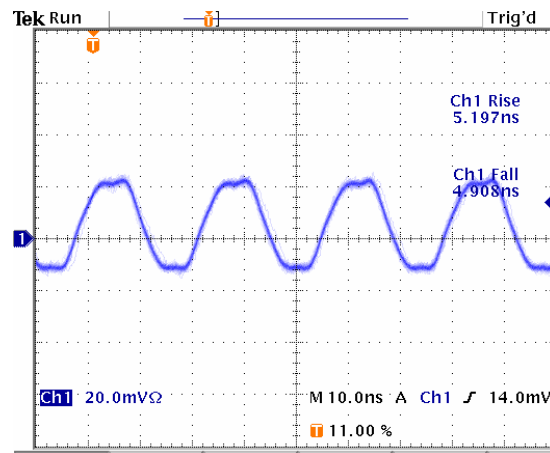


- ▶ It is **worst with long cable** (40 m ~ 15 %, 2 m ~ 2%)
- ▶ It is **not reflexion** from preceding pixel (impedance mismatch)
- ▶ It is connected to **cable bandwidth**
- ▶ After a hit, the signal has **not the time to fall down to base level**
- ▶ If we consider 1° order system, to get 99 % of signal or 1 % error
 - ▶ Rise / Fall time must be less than T Readout / 5
 - ▶ For **10 MHz clock** it gives $100 \text{ nS} / 5 = 20 \text{ nS}$

- ▶ **This problem has never been studied with FTP cables foreseen for EUDET demonstrator**
- ▶ **But it has also been observed with Mi*2 and FTP cables**



48 m FTP cable – 2,5 Mhz



48 m FTP cable – 42 Mhz

Test with 48 m FTP 100 MHz cable

- ▶ At 2,5 Mhz => $Tr \sim Tf \sim 50 \text{ nS}$
- ▶ At 42 Mhz => $Tr \sim Tf \sim 5 \text{ nS}$
- ▶ We must be in a **flat area of cable BW**
- ▶ **Coaxial cable is much better**
- ▶ We may need to **limit cable length**

Detailed investigation must be done

Conclusion

Firmware

- ▶ CDS on board, Trigger handling, Synchronous readout The of N boards are ready
- ▶ Distribution of new boards FW and SW : we hope it will be possible for end of January 2007 ?

Software

- ▶ Test with 6 boards, CDS readout, trigger handling => Done : 30 Hz
- ▶ Integration in our beam Telescope, beam test possible in spring 2007
- ▶ Proposition of Master / Slave architecture for DAQ and JTAG SW

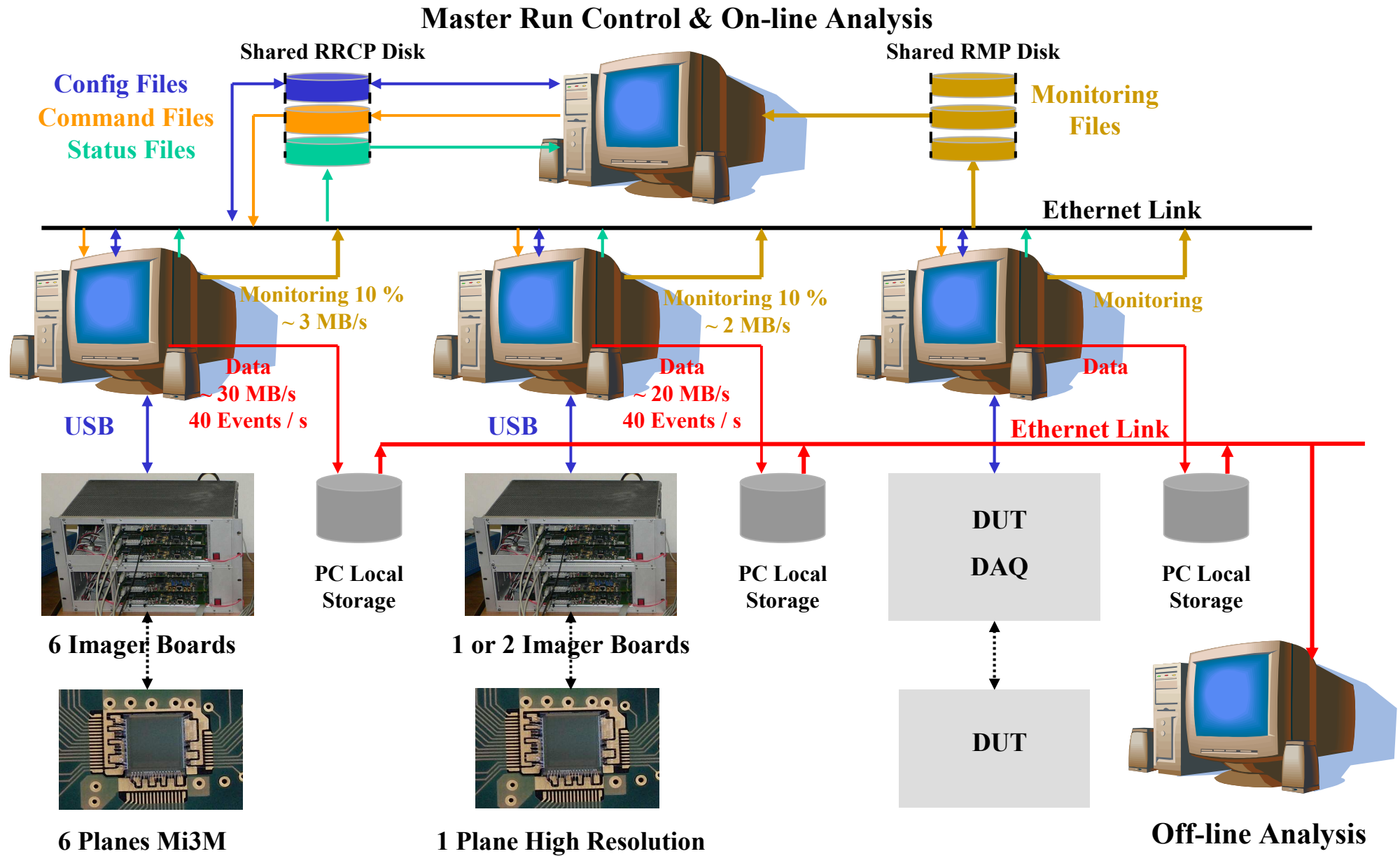
Integration

- ▶ A problem has been discovered on Mimosa readout chain
- ▶ It requires more detailed study (For example : test with Mi*2 at 10 MHz, 2 m, 10 m, 40 m)
- ▶ The consequence can be a cable length limitation between Telescope and DAQ

Backup Slides

- ▶ **Demonstrator DAQ architecture proposition**
- ▶ **RRCP**
- ▶ **RMP**
- ▶ **RRCP & RMP old slides (Design Review 06/04/2006)**
- ▶ **Trigger position & Hit polarity**

One proposition for Demonstrator DAQ






RRCP – Remote Run Control Protocol

The goal

- ▶ Provide a **common set of functions** for DAQ and JTAG **remote control**
- ▶ **Commands functions**
 - ▶ **Load a configuration file** (Boards conf for DAQ – Mimosa operating mode & bias for JTAG)
 - ▶ **Set run parameters** : run number, events number, destination directory ...
 - ▶ **Execution** : Start – Stop Run (DAQ) / Load Mimosa parameters in chip (JTAG)
- ▶ **Status functions**
 - ▶ **Return status of command function call** (passed / failed)
 - ▶ **Return current event number** (DAQ)

Implementation

- ▶ Based on **generic commands & status functions** : parameter bloc, command number, return block
- ▶ Library based on **three levels stack**
 - ▶ **Top**  **RRCP abstraction layer - high level functions** : Start / Stop Run
 - ▶ **Middle**  **Communication layer** – “ pipes ” library
 - ▶ **Bottom**  **Physical implementation** : command / status files with polling – Network protocol (RPC–TCP/IP) ...

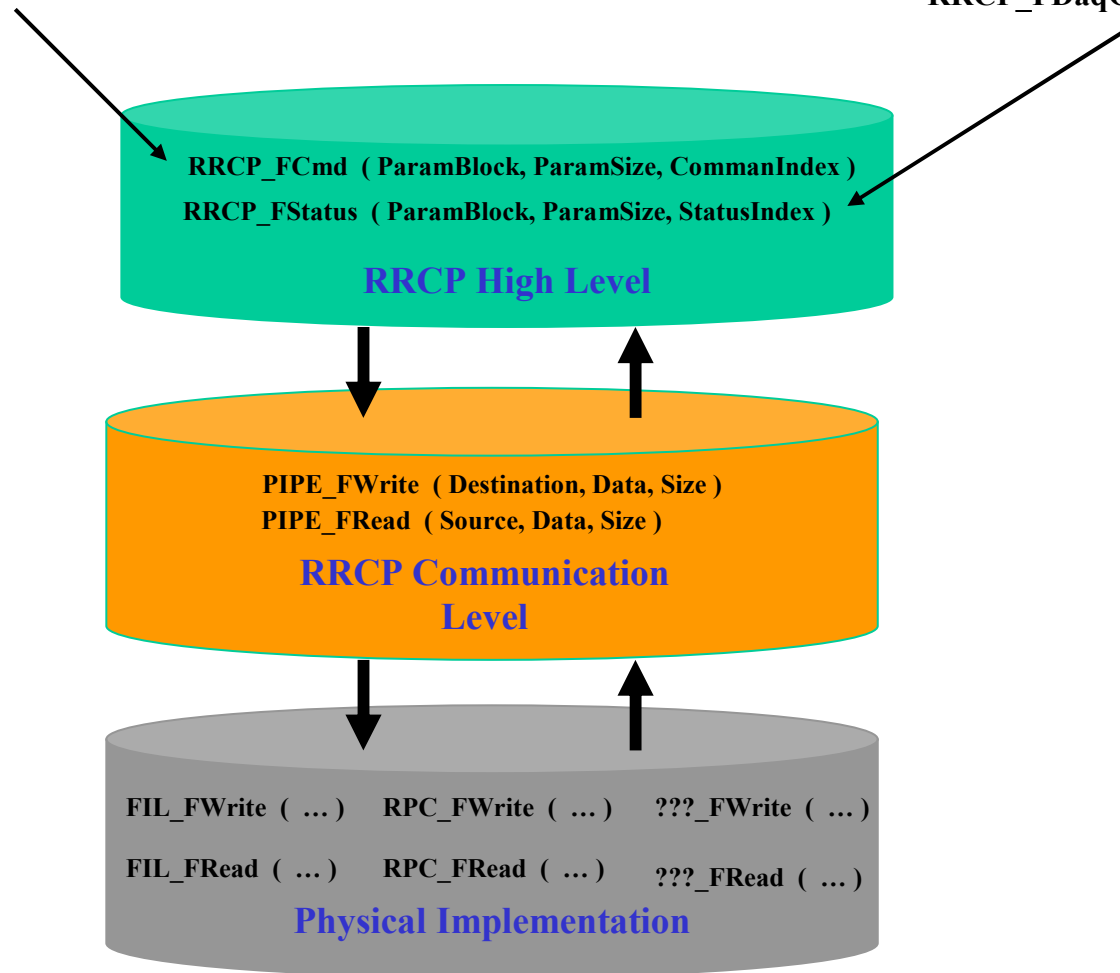
RRCP – Remote Run Control Protocol

Commands requests

RRCP_FDqLoadConf (DaqId, "6Planes_MimoTel")
RRCP_FDqSetRunPar (DaqId, "C:\data\\"", 777, 10000)
RRCP_FDqStartRun (DaqId)

Status requests

RRCP_FDqGetConfName (DaqId)
RRCP_FDqGetRunStatus (DaqId)
RRCP_FDqGetEventsCount (DaqId)






RMP – Remote Monitoring Protocol

Windows USB DAQ / Linux Monitoring

- ▶ **Both are ready** – How to build a bridge between DAQ and Monitoring ?
- ▶ We have **two different operating systems** ...
- ▶ Can we **avoid system programming** ?
- ▶ Can we do something **generic with minimum development efforts** ?
- ▶ Can we **build this bridge step by step** ?
 - ▶ To have **quickly a solution**, of course **not with best performances**
 - ▶ **Upgrade later** to improve transfer rate, flexibility

One idea ... RMP based on files

- ▶ It was the backup solution of our VME DAQ (Main solution = shared buffers)
- ▶ We can adopt the same structure as for RRCP
 - ▶ **Top**  RMP abstraction layer - **high level functions** : GetEvent ()
 - ▶ **Middle**  **Communication** layer – “ pipes ” library
 - ▶ **Bottom**  **Physical implementation** : small data files with polling – Network protocol (RPC–TCP/IP) ...

RMP – Remote Monitoring Protocol

Event requests

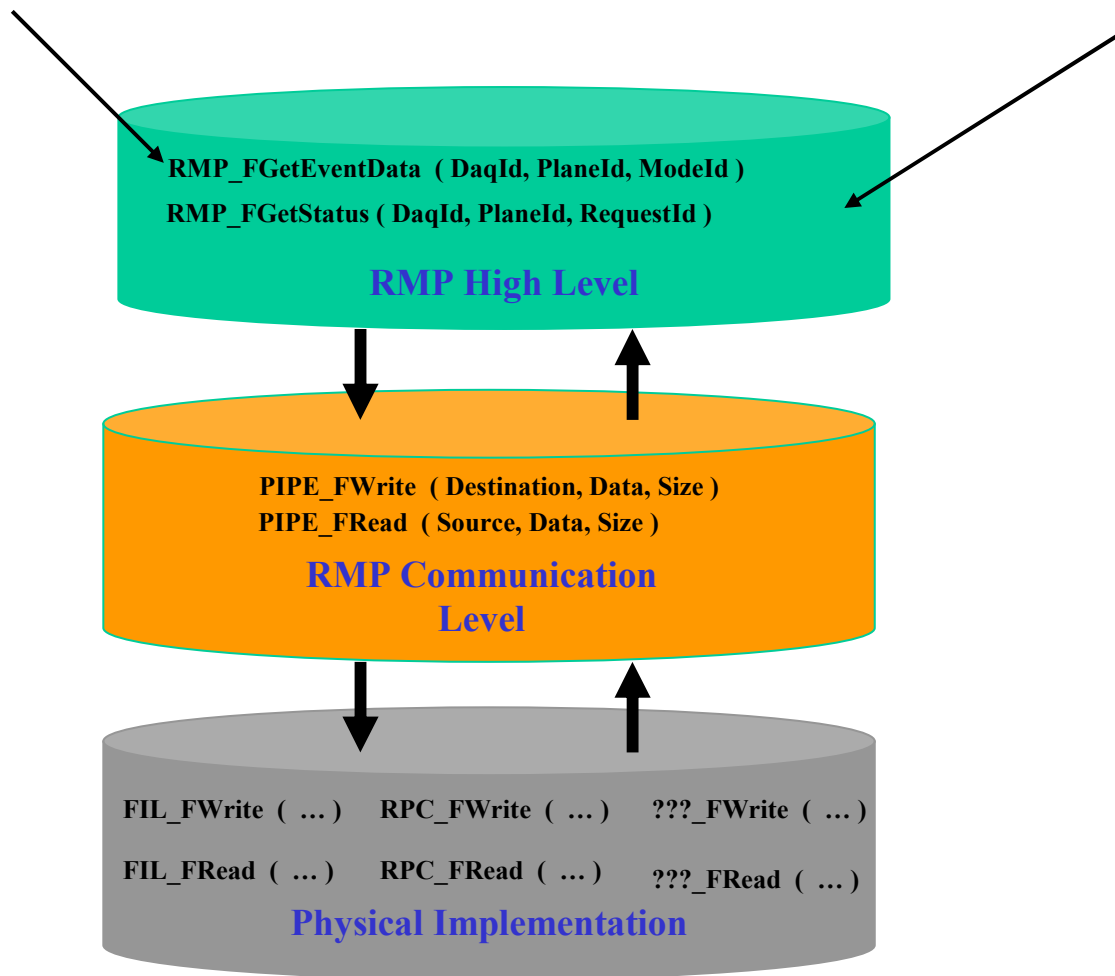
RMP_FGetRawData (DaqId, PlaneId, EventId)

RMP_FGetHitsList (DaqId, PlaneId, EventId)

Status requests

RMP_FGetEventRate (DaqId, PlaneId)

RMP_FGetHitsRate (DaqId, PlaneId)



How to synchronize MAPS, DUT, ... DAQ ?

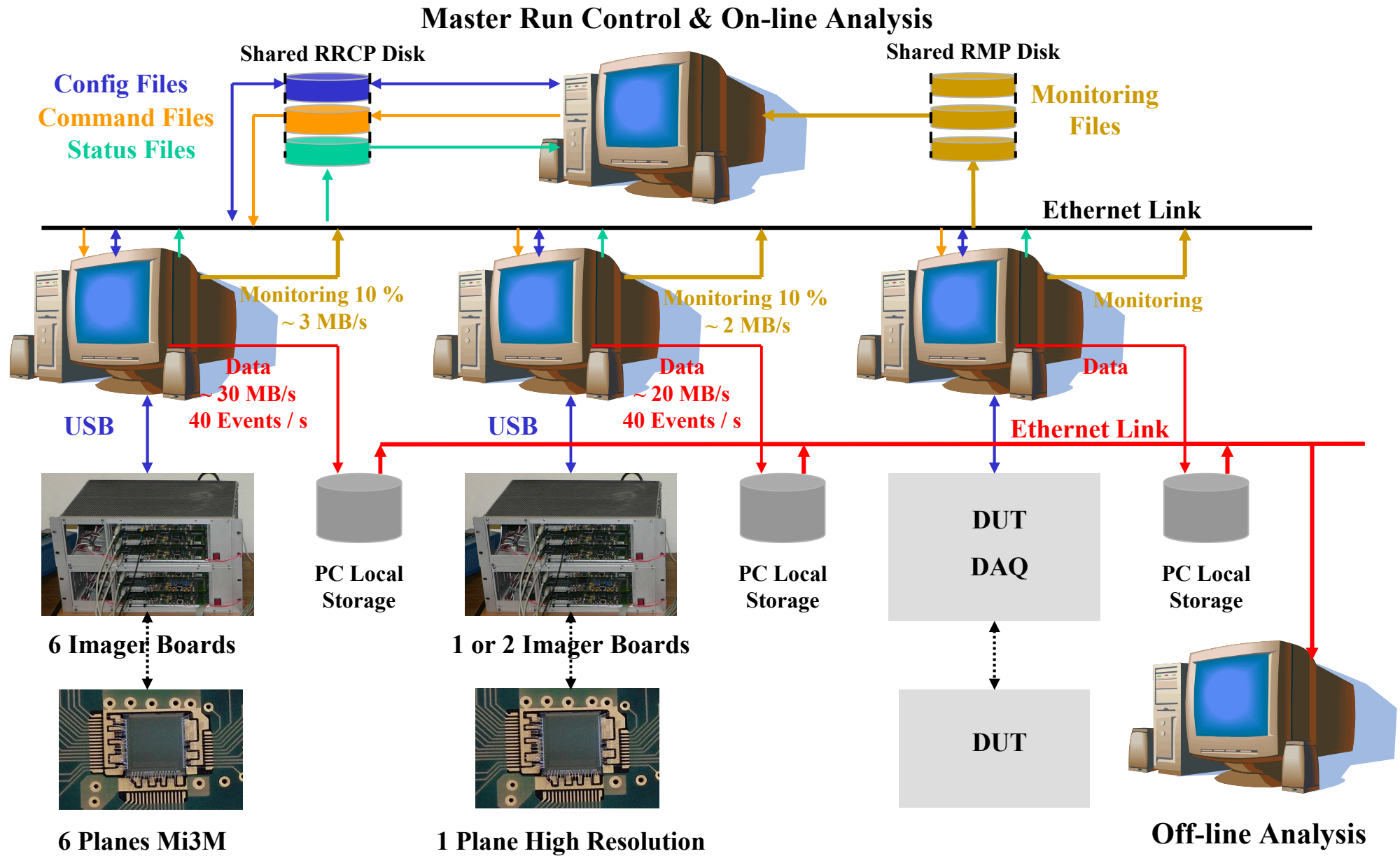
Remote Run Control Protocol (RRCP)

- ▶ Each DAQ is **controlled** either
 - ▶ In **local**, GUI **acts as a master** => Stand alone application in lab or User Telescope
 - ▶ By a **commands interpreter**, GUI **acts as a slave** => EUDET Beam Telescope
- ▶ Each DAQ handles his **configuration files** (stored in a working directory)
- ▶ How it works ?
 - ▶ EUDET Master Run Control application (MRC)
 - ▶ Copy **configurations** in each DAQ **working directory**
 - ▶ Send **command Start, Stop** ... with parameters by RRCP (write command file)
 - ▶ Slave DAQ applications
 - ▶ Interpreter receive **commands** by RRCP (read command file in polling)
 - ▶ **Execute** command
 - ▶ Return **status file** to MRC (write status file)

Remote Run Control Protocol ... RRCP ?

- ▶ This RRCP is not a genial idea ;-)
 - ▶ It's the standard way to control DAQ systems
 - ▶ But implementation can be done in a basic way
- ▶ The implementation
 - ▶ Using **command** and **status** files on a shared network drive
 - ▶ **Command** files from **Master Run Control to slave DAQ**
 - ▶ **Status** files from **slave DAQ to Master Run Control**
 - ▶ Checking command / status files by slow polling – remove file after processing
- ▶ A library provides RRCP function calls and files format
 - ▶ Only need to **add Remote Control & Status report features** on each **DAQ software**
 - ▶ Each team can **maintain his own DAQ software**

One proposition for Demonstrator DAQ



Conclusion

- ▶ Remote Run Control & Remote Monitoring Protocols = **Top layer** + 2 implementations
 - ▶ First - **basic version with files** : easy to program & debug
 - ▶ Second - **with standard network Protocols** : RPC, Socket pipe ...
- ▶ **No need to develop a global DAQ system Application**
 - ▶ **Master Run Control application**
 - ▶ **Each DAQ is independent BUT can be controlled as a slave system**
 - ▶ Allow to use the **same DAQ in User or EUDET context**
 - ▶ It will be easier for **DAQ upgrade => few modifications of MRC**
- ▶ **On-line Monitoring and Off-line analysis**
 - ▶ Provide libraries for **event building (EVB) from all Data Sources (MAPS, DUT ...)**
 - ▶ The **same EVB libraries** should be used for **On-line & Off-line analysis**

Two ideas to build Telescope global DAQ

▶ Idea N° 1 : DAQ HW API & Global DAQ Application

- ▶ Each group provides HW API libraries (eg : USB Board SDK, JTAG SDK)
- ▶ Need a global DAQ Application (“ Active Application ” : Real Time, SDR, EVB ...)

▶ Idea N° 2 : Slave DAQ & Master Run Control Application

- ▶ Each group implements **Slave Remote Control** in his DAQ Application
- ▶ Need a **Master Run Control Application** (“ Passive Application ” : Control, Supervision)

▶ Idea N°2 ... How to do : EVent Building (EVB) & Software Data Reduction (SDR) ???

- ▶ Pseudo on-line EVB from data files
- ▶ SDR ?
 - ▶ in each DAQ Application => 2 Outputs : RAW (keep as reference) + after DR
 - ▶ after final event building

Hit "After" trigger

