Demonstration module test program

module test meeting
24 September 2002

talk overview

- introduction
  - what to measure servers , clients
- hardware connections
- starting up the software
- initialization
- settings file
- automatic testing procedures
- data base storage
- software location
- status and to do

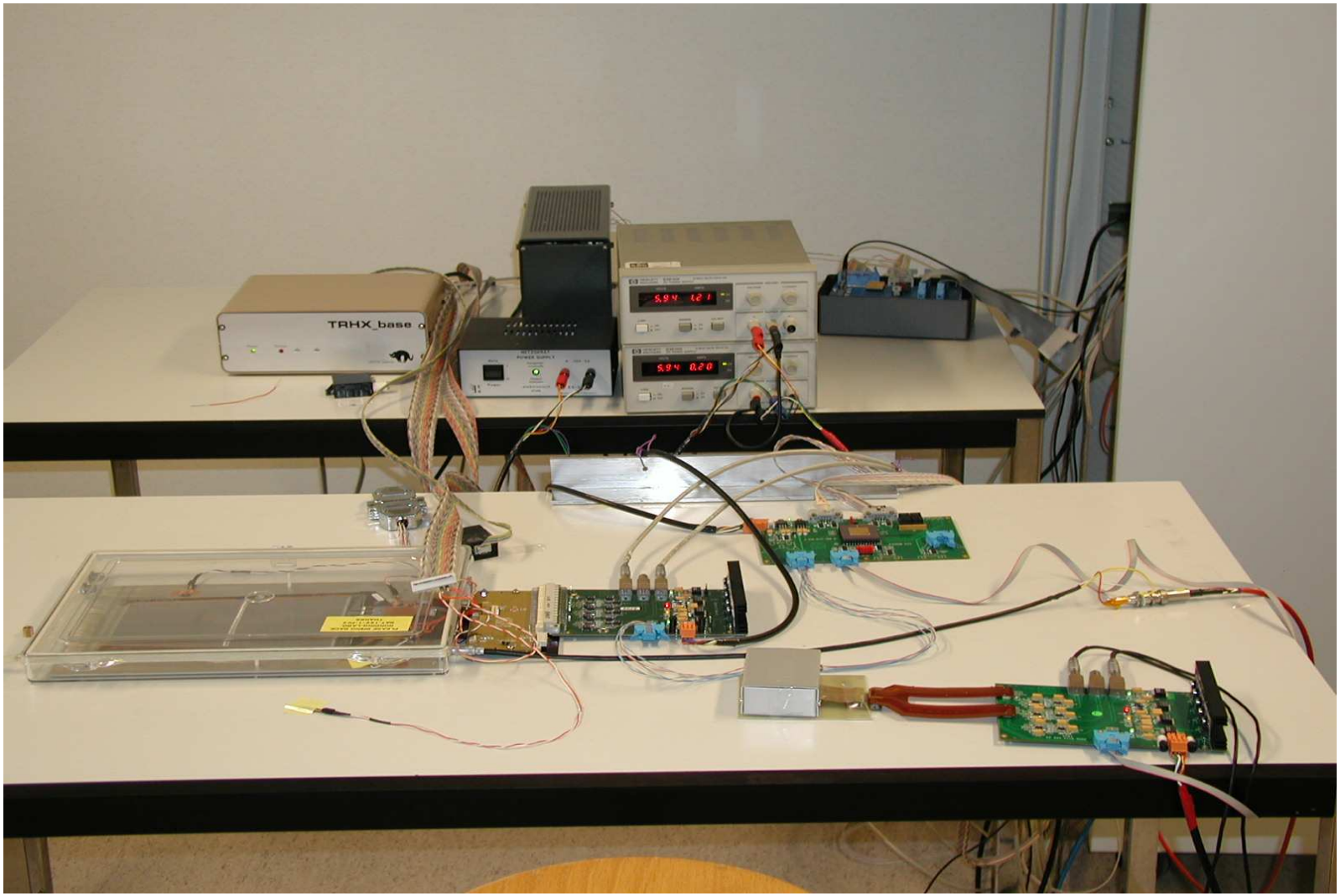
Software Team

- Testing is now more important then "writing"
  For testing (and contributions) "volunteered" :
  - Torino (L. Demaria G.Favro)
  - Pisa (A. Kyriaki)
  - and others (Firenze , Strasbourg)
- Writers :
  - Antwerpen W. Beaumont Vahagn Hovhannisyan
  - CERN L. Mirabito (Basic software)
  - Karlsruhe V. Zhukov

No simple package to install

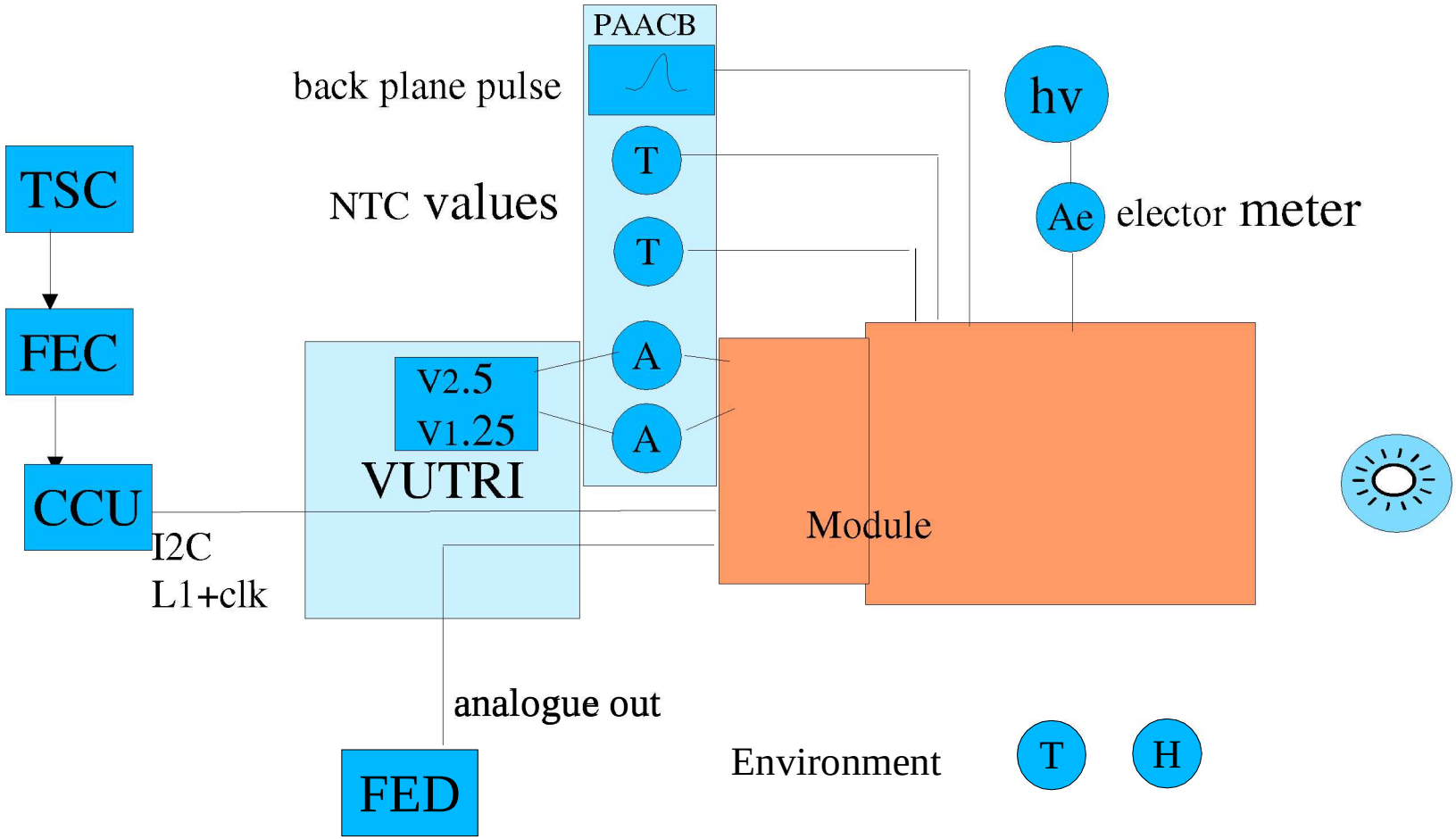
- support of a lot of hardware
- use of a lot of packages (xerces , qt , dim, root)
- not always clear constrains so has to be flexible
- different "applications"
  - hybrid test (because this was anyhow the basic)
  - module qualification test
  - long term test
  - diagnostics

The Antwerpen Setup



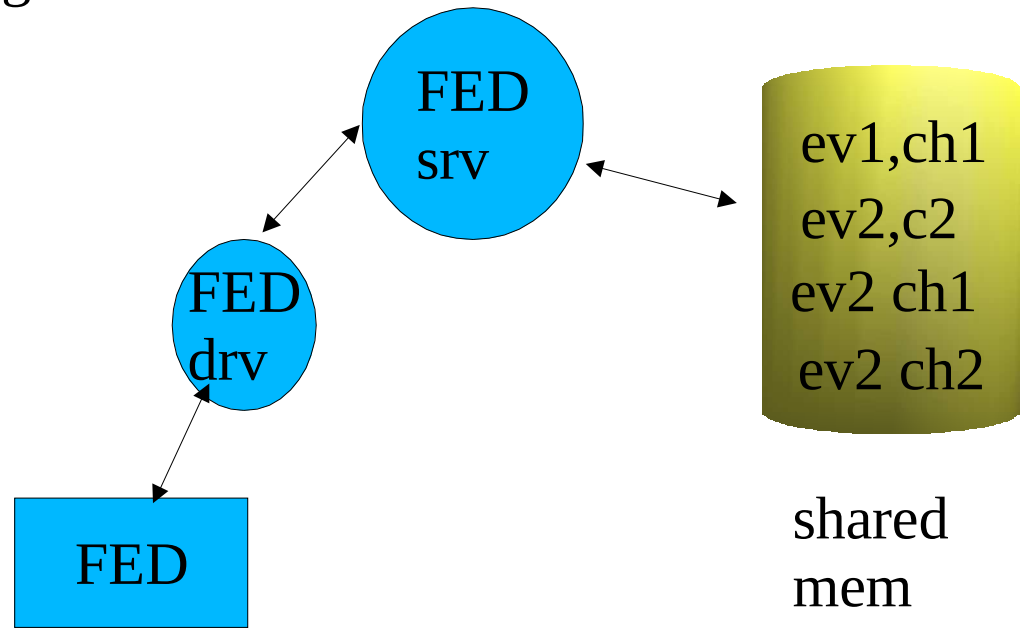
The module connections

(What can be measured)

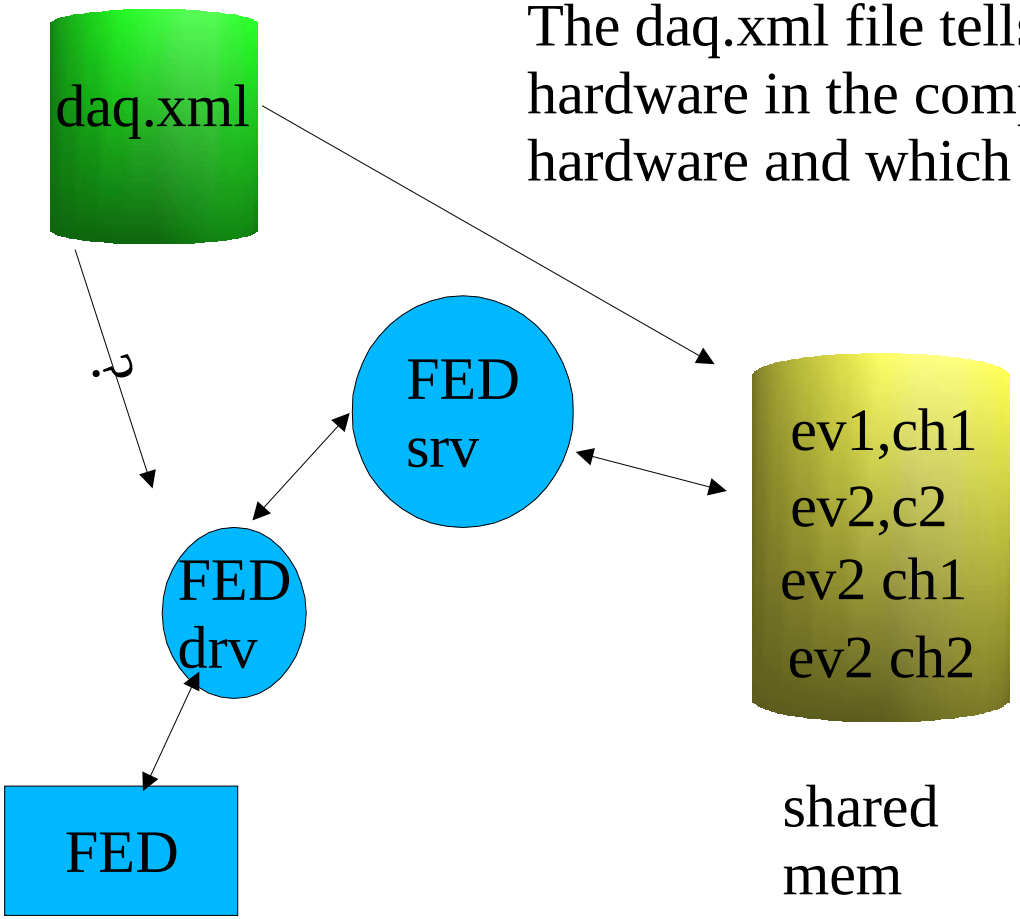


The server

The (FED) server reads via the feddriver permanent from the FED the events and store the result in a shared memory segment



The daq.xml file for the servers



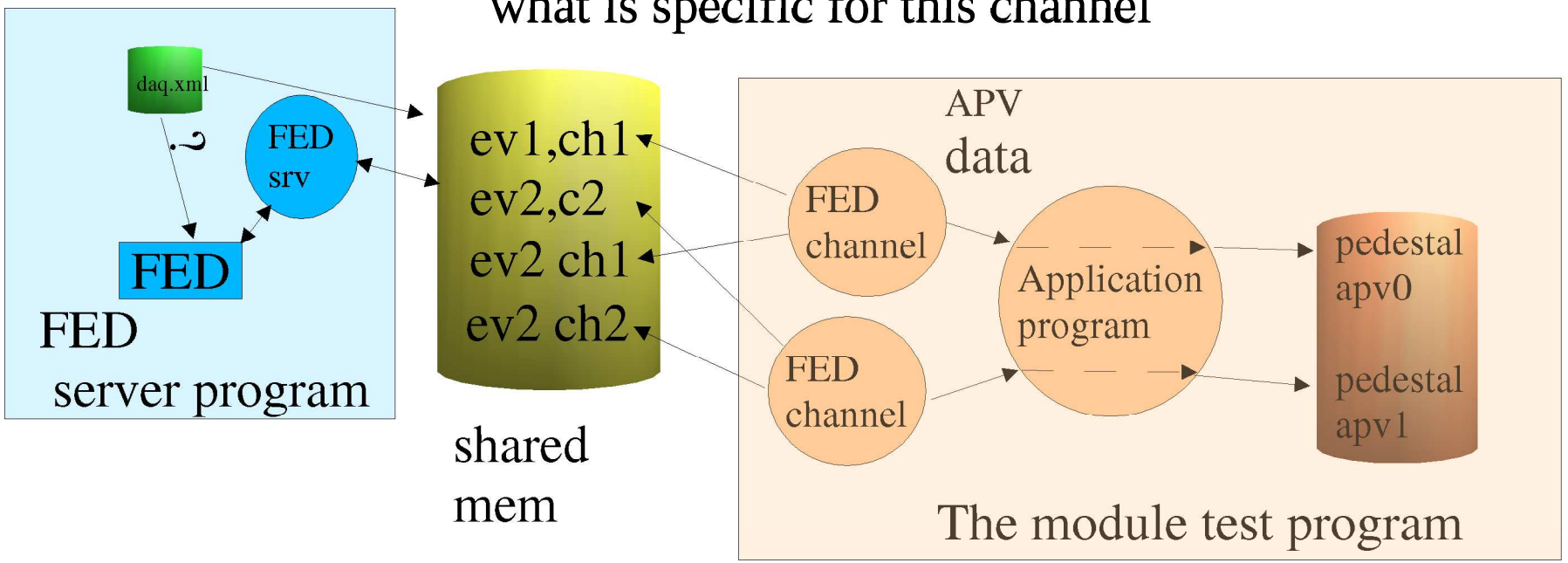
The daq.xml file tells the server where to "find" hardware in the computer, how to initialize the hardware and which data should be stored.

The daq.xml file for the servers

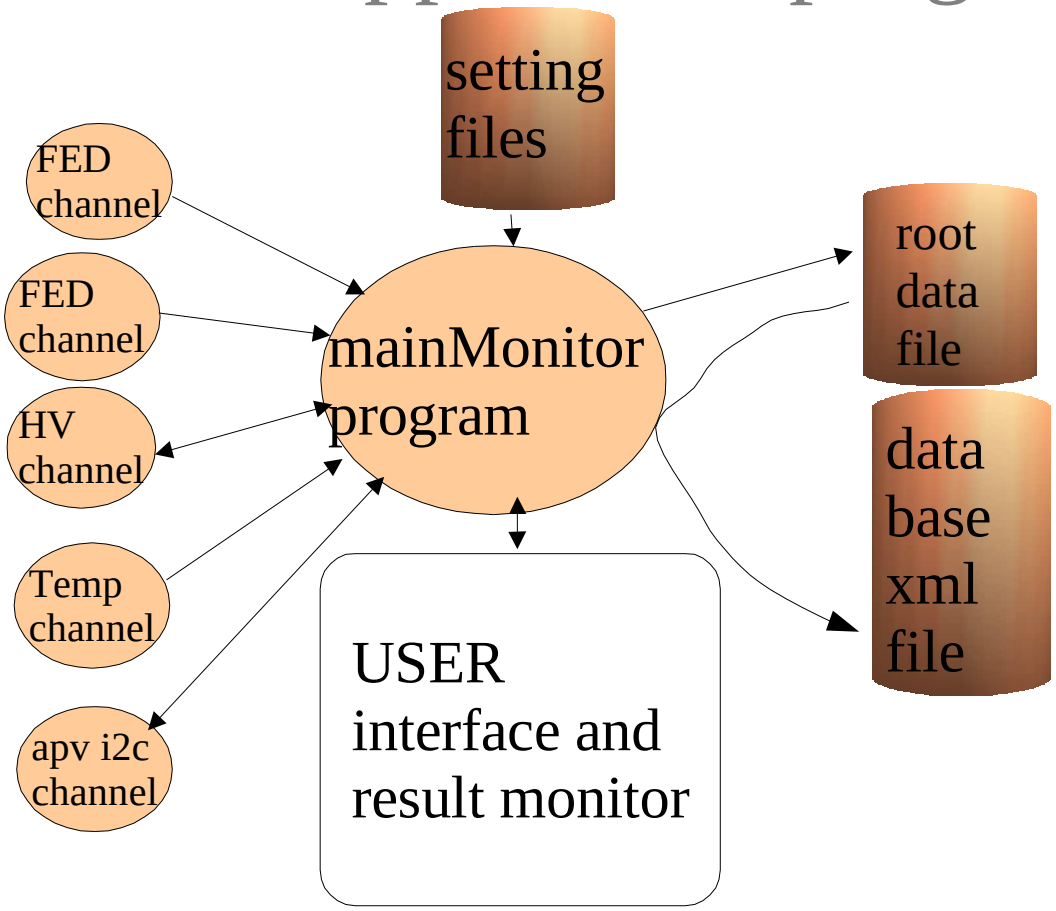
```
<?xml version="1.0" encoding="ISO−8859−1" ?>
<!DOCTYPE Daq SYSTEM "daq.dtd">
<Daq >
<Fed device="0" externalclock="true" externaltrigger="true" scopemode="false"
clockdelay="3" lowthreshold="200" highthreshold="300" sample="512"
   tscinhibittriggercontrol ="true"/>
<Fec device ="0">
<Ccu address="0x24">
<CcuChannel  number="5">
<I2cChannel address="0x68" type="PLL"/>
<I2cChannel address="0xCE" type="MUX"/>
<I2cChannel address="0x44" type="APV25"/>
<I2cChannel address="0x0" type="DCU"/>
</CcuChannel>
<CcuChannel  number="4">
<I2cChannel address="0x68" type="PLL"/>
<I2cChannel address="0xCE" type="MUX"/>
<I2cChannel address="0x44" type="APV25"/>
</CcuChannel>
</Ccu>
</Fec>

<Tsc />
<Adc device="0" frequency="1000" gain= "UP_5_00V" />
<Dac device="0"> <DacChannel number="0"  gain= "UP_5_00V" /> <DacChannel number="1"  gain= "UP_5_00V" /> </Dac>
<Nhqctrl  serialport= "0" />
<Trhx serialport ="1" />
<SY127 crate="0" address="0x340" />
<Ad4717Device  devicenumber="7">   <Ad4717  devchannel="7"   addresses="0 3" /></Ad4717Device>
</Daq>
```

The FED (client) channel

The application initiates the fed channel. This channel connects to the shard memory and takes from the data in the shared memory what is specific for this channel



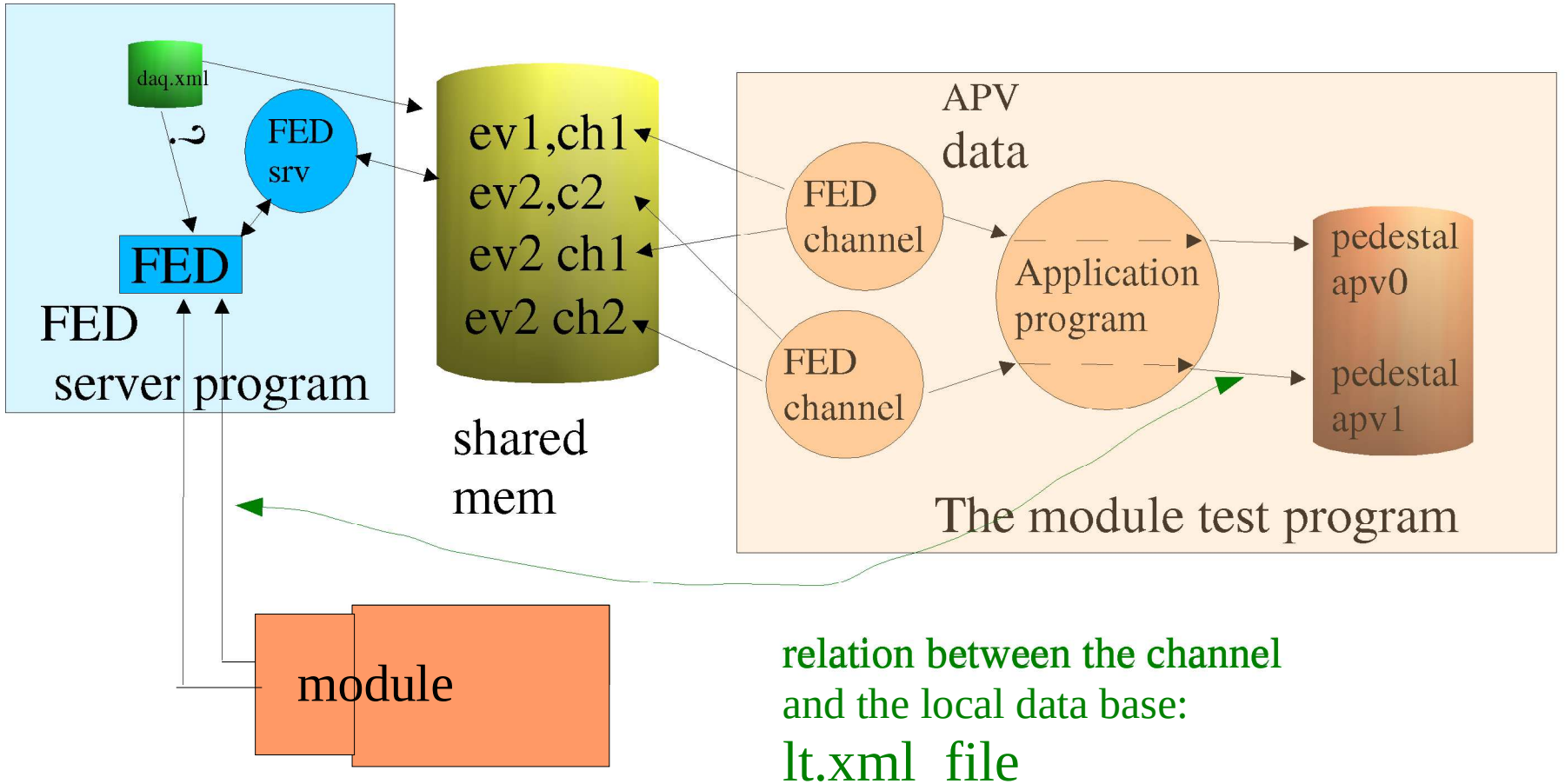
The application program



The application program

- Control of the data acquisition procedures for the different measurements, noise, calibration profile
- read environment and module conditions ("slow control")
- Store the result of data acquisition and slow control in root file and data base xml file.
- Controlled by user interface or scenario file with simple monitor representation of the results

Where to find the hardware

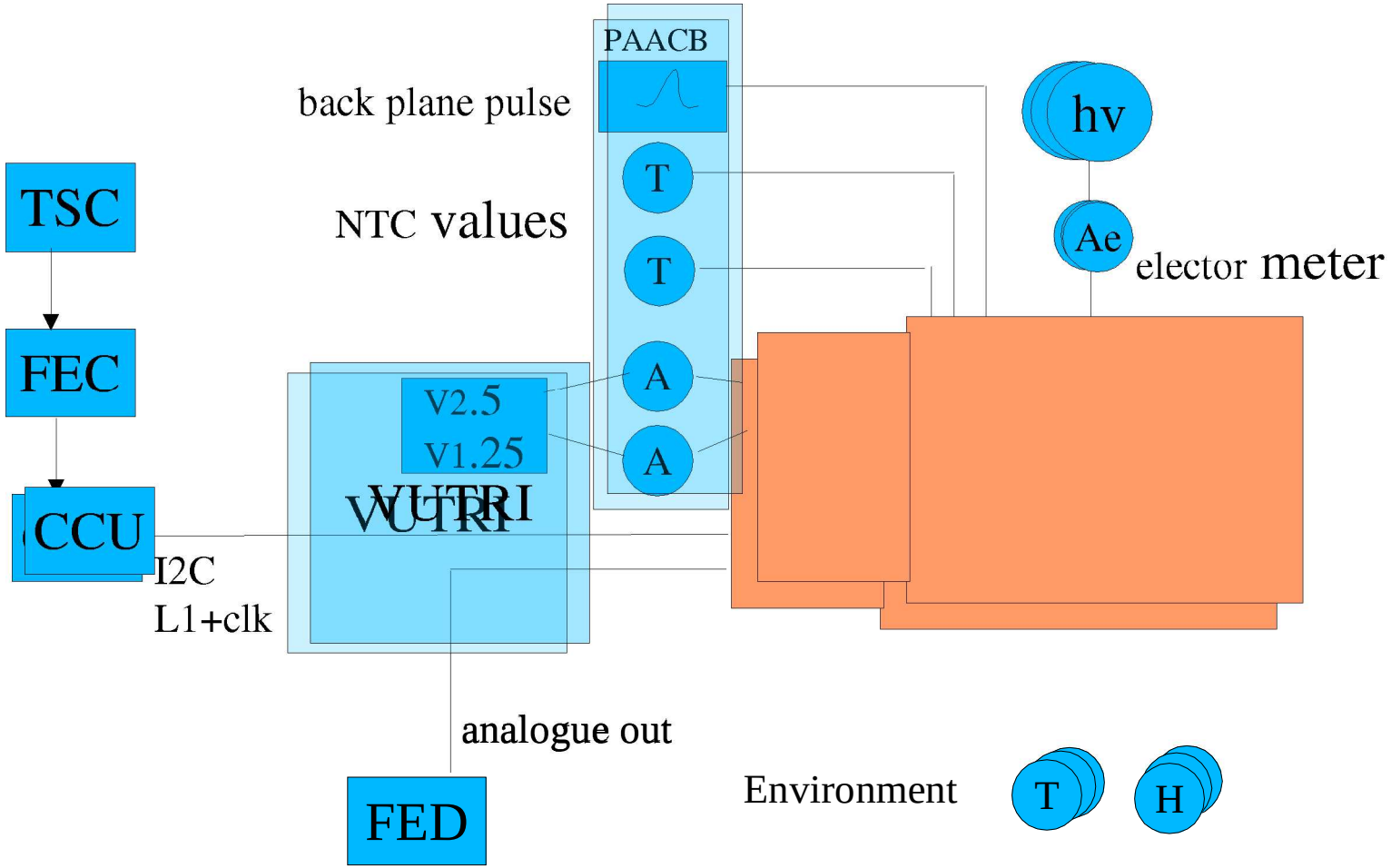


relation between the channel and the local data base:

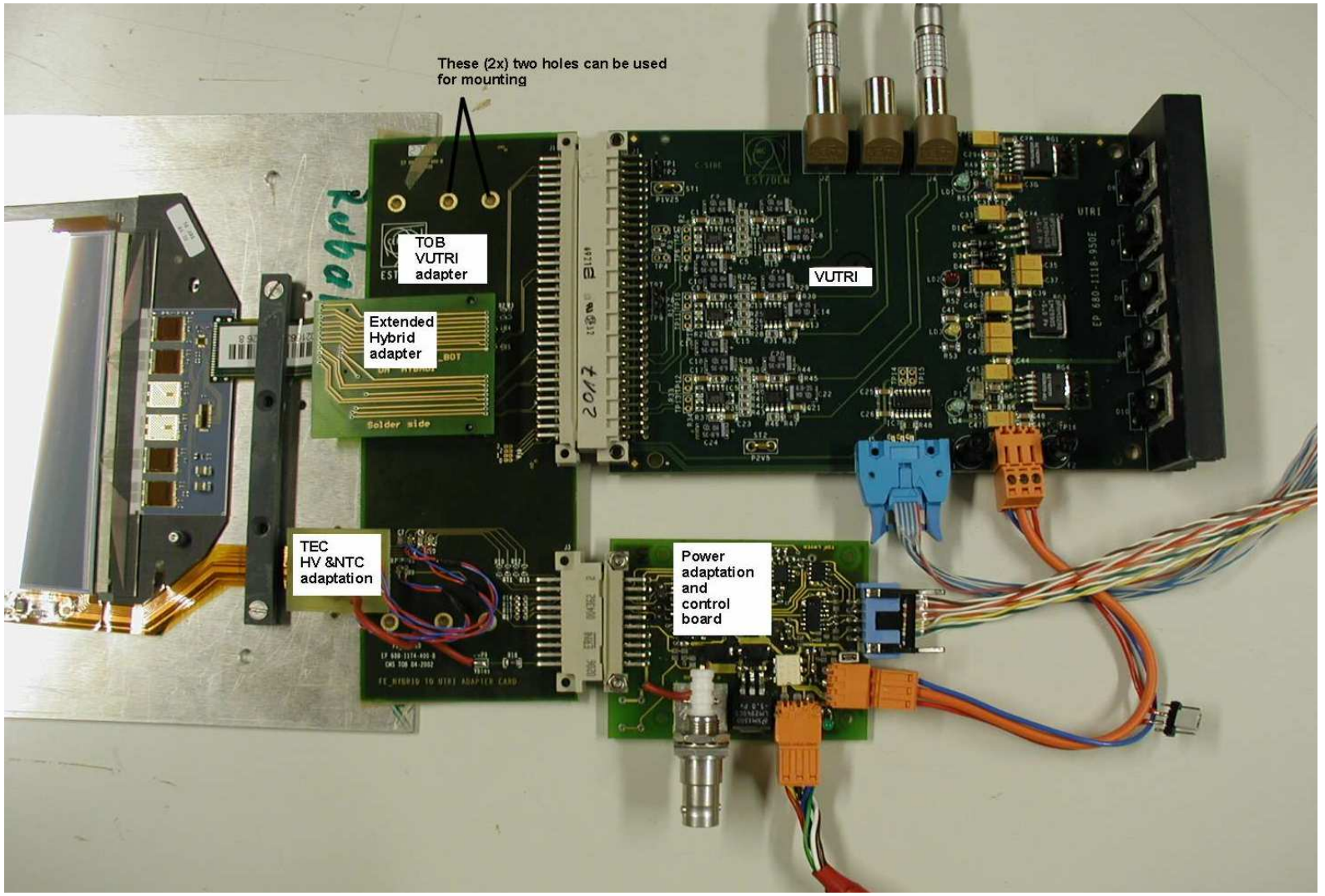
lt.xml file

The module connections

(for each bench position)



The module connections



The lt.xml file

- Definition connection of the environment sensors and module(s) to the hardware
- Calibration values for environment sensors (by example voltage to temperature conversion)
- Three sections
  - Data acquisition control (TSC)
  - Slow control (environment temperature coldbox)
  - Bench position connection (module connections)
- It DOESN’t define the number of modules !!!

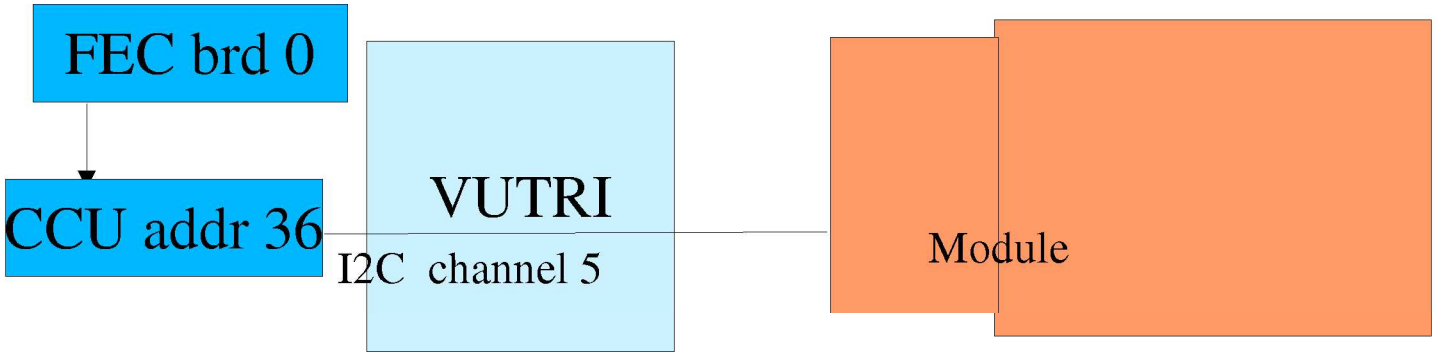
lt.xml bench position

```
<BenchPositionConnection device ="0">
  <I2cConnection    />
  <FedConnection   >
        <UtriChannel  utrichannel="0"  fedserialnr= "16" fedchannel="0"  />
        <UtriChannel  utrichannel="1"  fedserialnr= "16" fedchannel="2"  />
 </FedConnection>
  <TemperatureProbeNtc0  device ="1000"  name ="Module0a" simulation= "n">
        <TemperatureNTC    >       <AdcAd4717   devicenumber="7"  devicechannel ="7"  i2caddress="0"     channel="1"   /> </TemperatureNTC>
 </TemperatureProbeNtc0 >
 <TemperatureProbeNtc1  etc.   </TemperatureProbeNtc1 >
  <BiasHvConnection  device= "1000"   >
        <HvStruck gain="1"   >     <SerialChannel port="0"a  adress="1" />   </HvStruck>     </BiasHvConnection>
  <BiasCurrentConnection device= "1000"  >
        <BiasCurrentAdcChannel>
          <ElectroMeter   />
          <CalValues />
          <AdcChannel   boardnr="0"   channel="10"   />
        </BiasCurrentAdcChannel>
   </BiasCurrentConnection>
  <Current_1.25  device = "1001"  name="Lv1.25cur_mod0"  >
        <CurrentAdc>
          <CalValues    />
          <AdcAd4717 devicenumber="7" subdevicenumber="0"  devicetype ="tsci2c" devicechannel ="7"  i2caddress="0" channel="3"     />
        </CurrentAdc>
  </Current_1.25>
  <Current_2.50  etc              />   </CurrentAdc>         </Current_2.50>
  </BenchPositionConnection>
```

lt.xml bench position I2C connection

```
<BenchPositionConnection device ="0">
   <I2cConnection    />
```

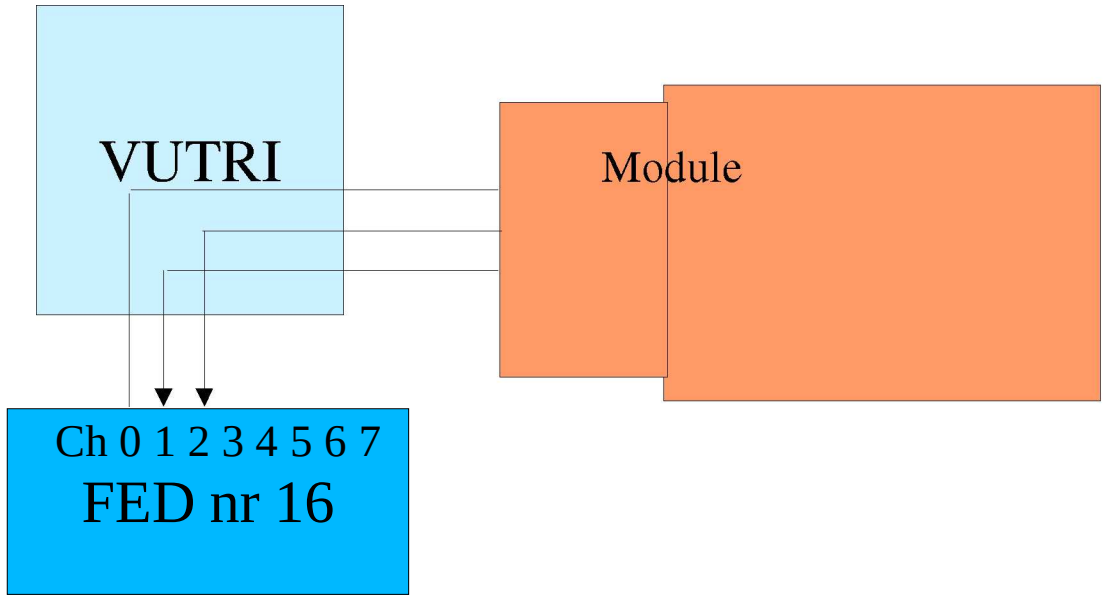
```
<I2cConnection     simulation="n"   fecboard="0"   ccuchannel ="5"
                ccuaddress ="36"     />
```




```
cenumber="0"  devicetype ="tsci2c" devicechannel ="7"  i2caddress="0" channel="3"     />
            </CurrentAdc>
      </Current_1.25>
      <Current_2.50  etc            />   </CurrentAdc>          </Current_2.50>

     </BenchPositionConnection>
```

lt.xml bench position FED connection

```
<BenchPositionConnection device ="0">
  <I2cConnection    />
    <FedConnection   >
      <UtriChannel  utrichannel="0"  fedserialnr= "16" fedchannel="0"  />
      <UtriChannel  utrichannel="2"  fedserialnr= "16" fedchannel="1"  />
      <UtriChannel  utrichannel="1"  fedserialnr= "16" fedchannel="2"  />
    </FedConnection>
```

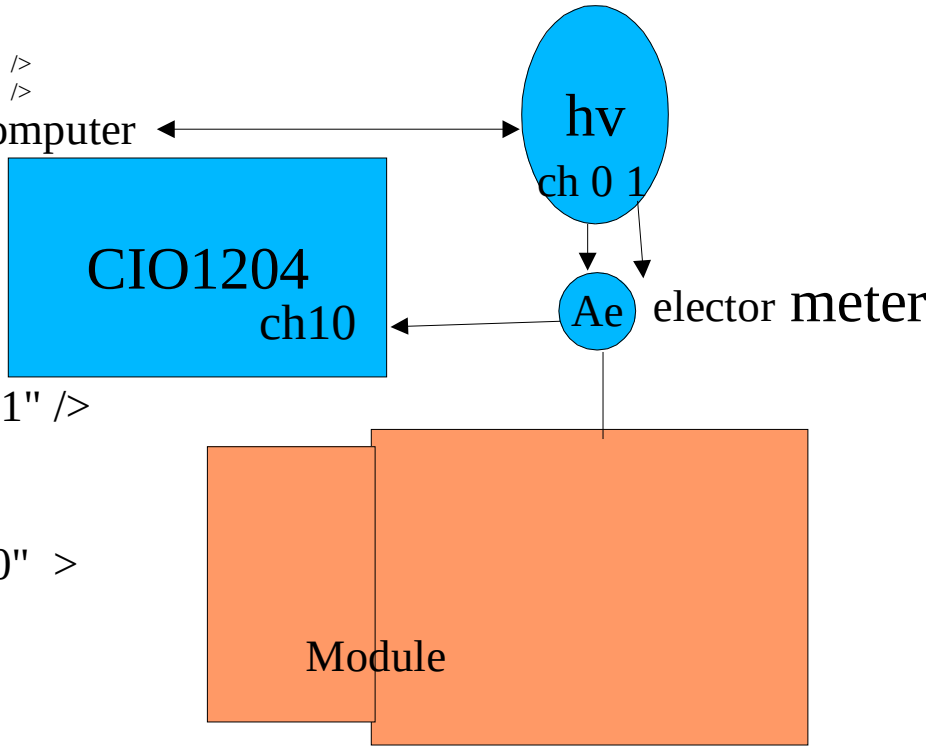



```
</BenchPositionConnection>
```

lt.xml bench position
the hv source and current

```
<BenchPositionConnection device ="0">
  <I2cConnection    />
    <FedConnection   >
        <UtriChannel  utrichannel="0"  fedserialnr= "16" fedchannel="0"  />
        <UtriChannel  utrichannel="1"  fedserialnr= "16" fedchannel="2"  />
  </Fed
```

to com0 of the computer




```
  <TemperatureProbeNtc1  etc.   </TemperatureProbeNtc1 >
   <BiasHvConnection  device= "1000"   >
       <HvStruck gain="1"   >
        <SerialChannel port="0"a  adress="1" />
       </HvStruck>
     </BiasHvConnection>
    <BiasCurrentConnection device= "1000"  >
       <BiasCurrentAdcChannel>
        <ElectroMeter       />
        <CalValues />
        <AdcChannel   boardnr="0"   channel="10"   />
       </BiasCurrentAdcChannel>
     </BiasCurrentConnection>

  </BenchPositionConnection>
```

Ltsetup file / bar code reader interface

- bar code reader interface is not yet implemented
- The LtSetup file defines the modules with their ID that are in the test. This has to be defined for each new test. Inside the Ltsetup file you find for each module
  - the bench position
  - the module ID
  - the numer of AVP's

The user interface

- slow control data is only updated if you click the slow control button
- reaction is not always fast
- blank graphs if no measurements are done
- only a limited graphs are available
- an "on line" monitor will be available to inspect all graphs

Ltsetup file / bar code reader interface

Example for 8 modules

0 11 4

1 12 4

2 13 4

3 14 4

4 15 4

5 16 4

6 17 4

7 18 4

the setting.xml file

- Tells the mainMonitor program which settings it has to use for the measurements
  - APV register settings
  - number of events , cuts
- Default values are stored in the settings.dtd file. Only the differences are put in the settings.xml file.
- example settings for the cal profile run:
  <CalProfRunDescriptor name="default" nevpoint="500"
  latstart="100" latend="90"
  calamp="80" tbad="3" />

the setting.xml file

- Different I2C descriptor are used in scenario files
- examples:
  - <I2CDescriptor name="i2cpedpeak" Mode="15" Latency="100" />
  - <I2CDescriptor name="default" Mode="15" Latency="100"/>
  - <I2CDescriptor name="i2cpedpeakinv" Mode="47" Latency="100"/>
  - <I2CDescriptor name="i2cpeddecinv" Mode="37" Latency="100"/>

The scenario file

- The scenario file is the instruction sequence for the main monitor program for by example the qualification procedure
- located in $DAQHOME/ModuleTest/Test/Data
- example
  - time action parameter
  - 1 Start 0
    5 ChangeHV 100
    50 PedRun i2cpedpeak
    100 SaveRec 0
    110 SaveRec 1
    120 SaveFile testpeak
    130 Stop 0

The scenario file

- The time is the starting time. Has to be corrected for each test station. Do first a test via the control panel and check in the logfile the duration of a test
- slow control is checked automatically
- save record , save file see explanation local data base files
- Same type of test overwrites the previous result if not saved before

storing the data in the production data base direct

- After a scenario file with the save instructions is finished click on the save to xml
- choose the correct template (*.res) file. by example modvalidation.res . The file 132_modvalidation.res will be produced (132 is an example module ID)
- copy the file to 132.res in the result directory of the big browser.
- In the workstation/calibration tab of the big browser give the module ID and save the results

storing the data in the production data base after the run (user interface to be implemented)

- After a scenario file with the save instructions is finished a file res_M132.root is created (132 is the module ID)
- inspect the results with the root browser

easy to use ?!

- Servers have to start manually (script)
- initialization ("connect module to setup") has to be done every time. Requirement during production
- Almost all parameters can be altered by the user
- Main Goal:
  Automatic test procedure
  still manual
  - starting of the scenario file
  - saving of the file that has to be stored in the databas
  - storing of this file into the data base

software location structure

- Lt
  - Config directory:
    - settings.xml lt.xml and script examples
  - structure of the program directories
    - Makefile
    - include , src : the classes definitions and source fils
    - obj library and object files after compiling
    - doc documentation
    - test/src code for exectuables
    - test/bin the executables after linking
    - test/data : result files input files for the exectuables

software structure (cont)

- Lt
  - ClientTools
    - include , src : the classes for the temperature, fed ..... channels
    - test/src code to test individual channels
    - test/bin the executables after linking like TestLtTemperature 12 shows the readout of the temperature probe with device ID 12

software location structure (cont)

- Lt
  - ModuleTest
    - include , src : the classes for the application mainMonitor and onmon : userinterface , testprocedures, runs (pedrun etc) , data analyzing
    - test/src main application program mainMonitor
    - test/bin the executables after linking
    - test/data the directory to start mainMonitor where (examples) for scenario and Ltsetup files are located and the templates for the production data base, root macro's

software location structure (cont)

- Lt
  - DbLocal
    - include , src : the classes to store data into and extract data from the local data base (root) file .
    - test/src not documented test programs
    - test/bin
    - test/data (obsolete) root macro's

software location structure (cont)

- Lt
  - DbBase
    - include , src : the classes to read data from the local data base (root) file , the template *.res file and to store it into the production database format (xml) .
    - test/src source to read a local data base (root) file , read a template and stores it into the production database format (xml) .
    - test/bin the excecutabe
    - test/data

status

- most necessary procedures for module qualification are implemented
- not implemented I2C tests (address tests etc.)
- CalProfRun, IV are slow
- no bar code reader interface
- Results of the tests can be stored in the production data base
- local root files for diagnostics with root browser or onmon program (soon).

to do

- Testing
- connectivity tests
- implement LED tests
- implement Multiplexer
- finalize online monitor
- control of the cooling plant via dim (LLN)

support

- Documentation :
  http://hep.uia.ac.be/cms/testing/
- e−mail support:
  ltmodsup@uia.ua.ac.be
- mailing list for announcements
  cms−support−tk−sw−moduletest@cern.ch
  with web archive