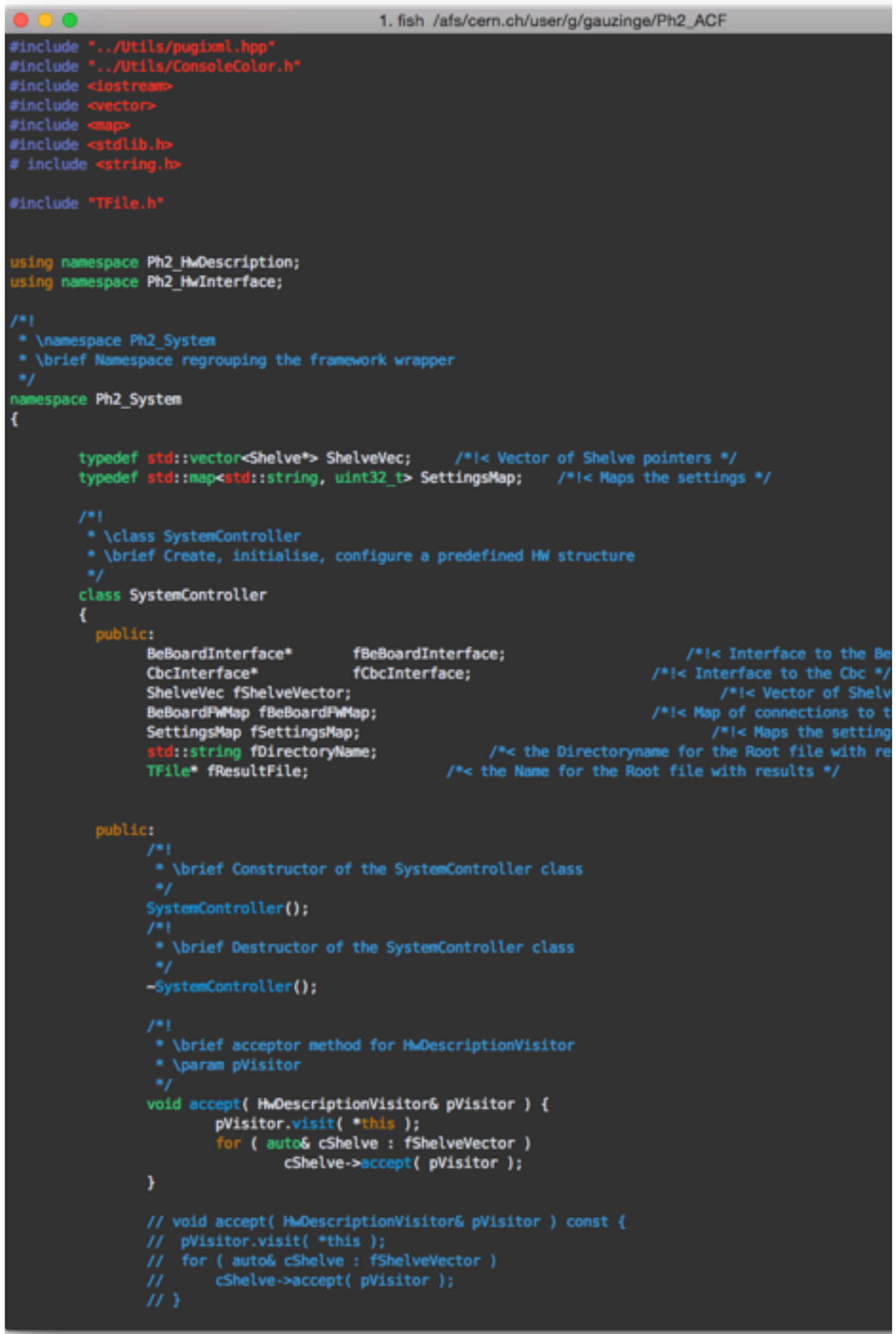
Ph2 DAQ Mid-/ High-Level SW Status & Tutorial

A. Kokabi, F. Ball, G. Auzinger, S. Mersi,
S. Dutta, S. Roy Chowdhury, S. Sarkar

georg.auzinger@cern.ch



Contents

- **Ph2 ACF**:
  - **Intro / Overview**
  - **Structure**
- **Higher-level tools**
- **GUI**
- **Tutorial**

Ph2 Acquisition & Control Framework Intro

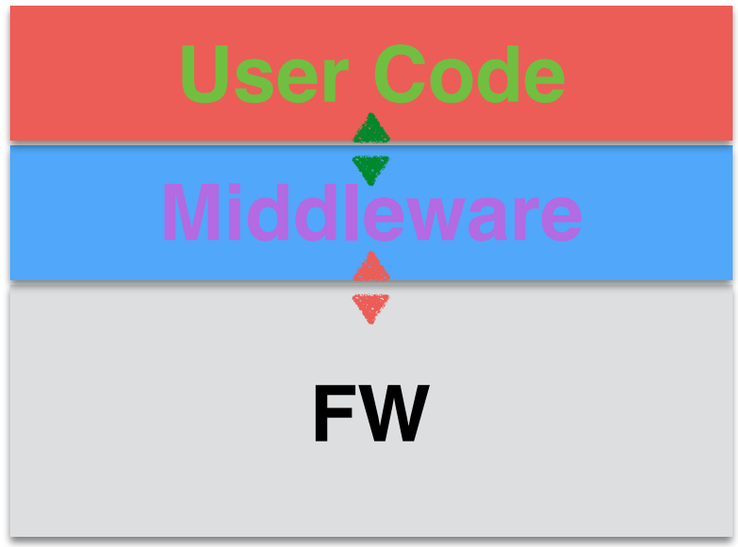
- release of “first beta” version last week - hopefully bug-free and stable
- now **circumventing the IPBus bug** reported by L. Gross (256th word always 0)
    - also affected CBC register read-back: some users experiencing trouble with FrontEndControl register [which is #256] when configuring - hopefully found a solution - needs validation!
    - will this be fixed in FW / uHAL in the future?
- code available on: https://github.com/gauzinge/Ph2_ACF
- 3 branches:
    - master: all you should be caring about - stable
    - Dev: only relevant if you are actively developing
    - DevGUI: Fionn’s GUI relevant code
- doxygen documentation: http://cms-project-tk-ph2daq-doxygen.web.cern.ch/
    - rudimentary but growing
- want to acknowledge the work of Kirika, Lorenzo Bidegain, Nicolas Pierre & others!

Ph2 ACF Overview

- the package is **intended as middleware**, i.e. a set of libraries that wrap uHAL and FW calls in a **consistent and constant-over-time** package, providing convenient and abstract interfaces to the user

- it is **not a fully-featured DAQ SW** package!
  - although we also provide methods to read data that will offer more features in the future

- some application examples come with the package



Ph2 ACF Prerequisites & Installation

- find installation instructions on github (README.md):
- pre-requesites:
  - requires gcc > 4.8 (C++0x features)
  - uHAL 2.3.0
  - no more dependence on boost (other than uHAL)
- installation:
  - git clone https://github.com/gauzinge/Ph2_ACF
  - make in top-level directory
  - add /bin and /lib to your path
- that's it!
- **soon to be included in new VM release**

Preliminary Setup

Firmware for the GLIB can be found in /firmware. To check you are using the correct firmware with the correct FMC, please read /latex/FirmwareHardware.pdf.

NOTE: If you are doing the install for the first time on the latest VM v1.1.0 then follow the preliminary setup, otherwise you can skip this.

1. Remove the current gcc and old boost libraries:

```
sudo yum remove devtoolset-1.1-gcc-debuginfo
sudo yum remove boost
```

2. Install the latest gcc compiler:

```
sudo yum install devtoolset-2
sudo ln -s /opt/rh/devtoolset-2/root/usr/bin* /usr/local/bin/
hash -r
```

This should give you gcc 4.8.1:

```
/usr/bin/gcc --version
```

3. Finally, update uHAL to version 2.3:

```
sudo yum groupremove uhal
wget http://svnweb.cern.ch/trac/cactus/export/28265/tags/ipbus_sw/uhal_2_3_0/scripts/rel
sudo cp cactus.slc5.x86_64.repo /etc/yum.repos.d/cactus.repo
sudo yum clean all
sudo yum groupinstall uhal
```

Ph2 ACF: what comes per default

- **libraries**:
  - Ph2_HWDescription: class-representations of all physical components of the DAQ chain with their properties
  - Ph2_HWInterface: interfaces to GLIB FW (transparent to the user) + 1 dedicated BeBoard(user)Interface + Cbc(user)Interface
  - Ph2_System / Ph2_Utils: auxillary classes like: top level manager, Event / Data classes, visitors, …
  - Ph2_Tools: classes derived from system controller for calibration, hybrid testing, …
- **test applications**: systemtest (config file validation), interfacetest (benchmarking, playground), datatest (read-out and data verification), calibration (implementation of the calibration algorithm introduced by K. Uchida), hybridtest (application for A. Honma's hybrid verification process)
- **config files**:
  - HWDescription_2(8)Cbc.xml: default config files for 2/8 CBC setups, examples
  - ApplicationName2CBC.xml: example config files for the various applications including the application-specific settings
  - Cbc_default_hole/electron.txt: starting point for calibration

Ph2 ACF: HW Description

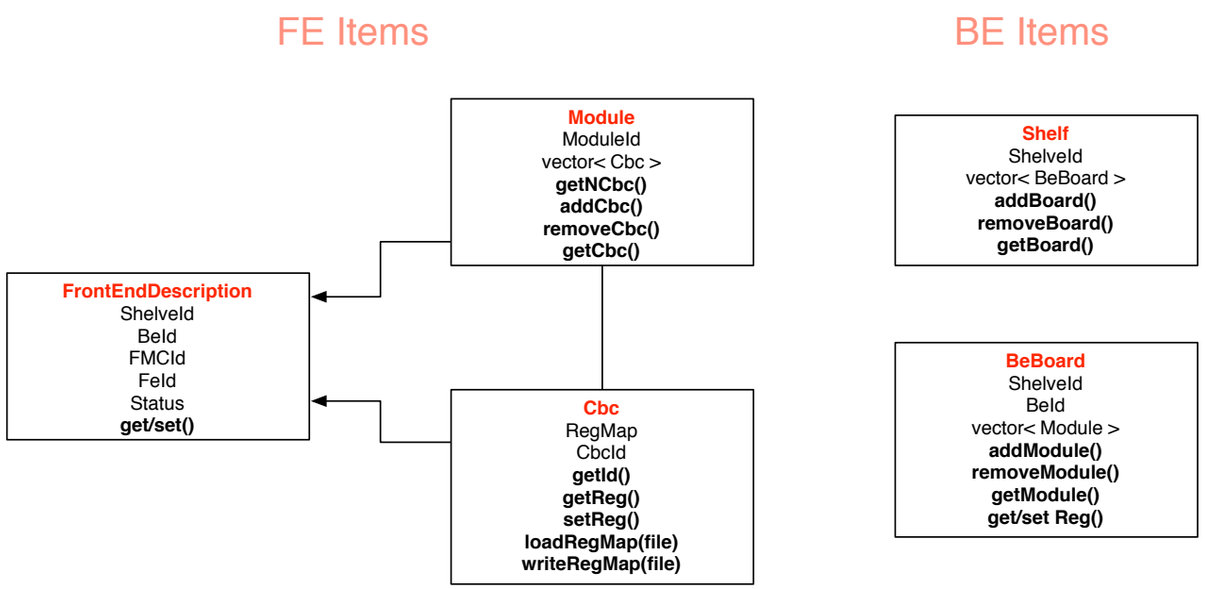
- SW structure explained here and here

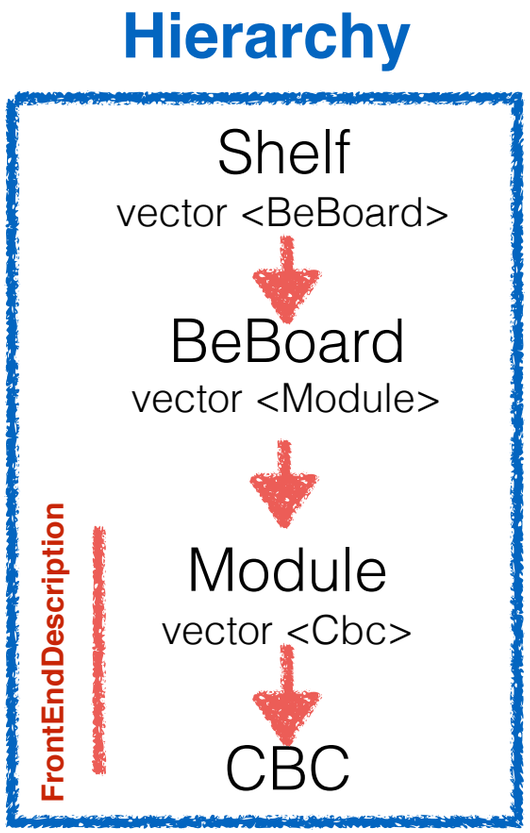
- **1 class for each physical object in the DAQ chain**: Cbc, Module, BeBoard, Shelf

- each with all relevant properties (maps for registers)

- **exact replica of HW object state is kept in memory**

- used to identify destinations for the interfaces





Ph2 ACF: HW Interfaces

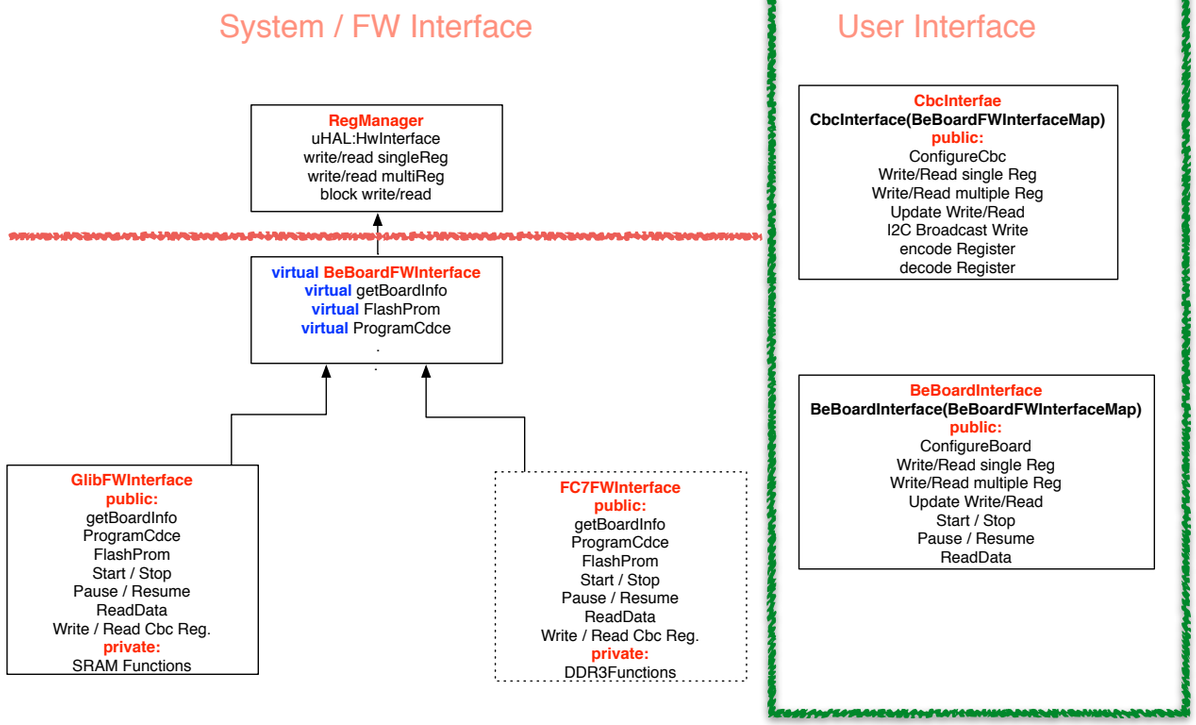
- 3 levels of Interfaces:

transparent to user

- RegManager: wraps uHAL
- (virtual) BeBoardFWInterface + GLIBFWInterface (implementation): HW specific sequences

user code

- BeBoard/CBC Interfaces for user: abstract methods (Configure, Start, Stop, Write, …)



- RegManager decouples us from changes to uHAL
- Board type transparent to user
- each new board / FW needs 1 new implementation of BeBoadFWInterface
- user interfaces should remain constant
- easy to add CBC3 or Concentrator interfaces later

Ph2 ACF System component

- SystemController class owns HWDescription hierarchy & Interfaces
- can parse config files and construct specified HW structure
- allows to override settings (GlobalCbcReg,…) and pass specific settings to applications
- is intended as base class for all other applications:
  - provides: InitializeHW(), InitializeSettings(), ConfigureHW(), Run(), InitializeResultFiles(),… methods
  -
```
class Calibration : public SystemController{

   whatever other methods();

};
```

Ph2 ACF Config Files

1. fish /afs/cern.ch/user/g/gauzinge/Ph2_ACF

```
<?xml version='1.0' encoding = 'UTF-8' ?>
<HwDescription>
    <Connections name="file://settings/connections_2CBC.xml"/>

    <Shelve Id="0" >
        <BeBoard Id="0" connectionId="board0" boardType="GLIB">

            <Register name="COMMISSIONNING_MODE_CBC_TEST_PULSE_VALID"> 1 </Register>
            <Register name="COMMISSIONNING_MODE_DELAY_AFTER_FAST_RESET"> 50 </Register>
            <Register name="COMMISSIONNING_MODE_DELAY_AFTER_L1A"> 400 </Register>
            <Register name="COMMISSIONNING_MODE_DELAY_AFTER_TEST_PULSE"> 201 </Register>
            <Register name="COMMISSIONNING_MODE_RQ"> 1 </Register>
            <Register name="cbc_stubdata_latency_adjust_fe1"> 1 </Register>
            <Register name="cbc_stubdata_latency_adjust_fe2"> 1 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands.ACQ_MODE"> 1 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands.CBC_DATA_GENE"> 1 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands.CBC_DATA_PACKET_NUMBER"> 10 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands.INT_TRIGGER_FREQ"> 8 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands.TRIGGER_SEL"> 0 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands2.clock_shift"> 0 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands2.negative_logic_CBC"> 1 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands2.negative_logic_sTTS"> 0 </Register>
            <Register name="user_wb_ttc_fmc_regs.pc_commands2.polarity_tlu"> 0 </Register>

            <Module FeId="0" FMCId="0" ModuleId="0" Status="1">
                <Global_CBC_Register name="VCth"> 0x78 </Global_CBC_Register>
                <Global_CBC_Register name="TriggerLatency"> 0x0C </Global_CBC_Register>
                <CBC_Files path="./settings/"/>

                <CBC Id="0" configfile="Cbc_default_hole.txt"/>
                <CBC Id="1" configfile="Cbc_default_hole.txt"/>
            </Module>
        </BeBoard>
    </Shelve>
</HwDescription>

<Settings>
    <Setting name="RunNumber"> 1 </Setting>
    <Setting name="HoleMode"> 1 </Setting>
</Settings>
```

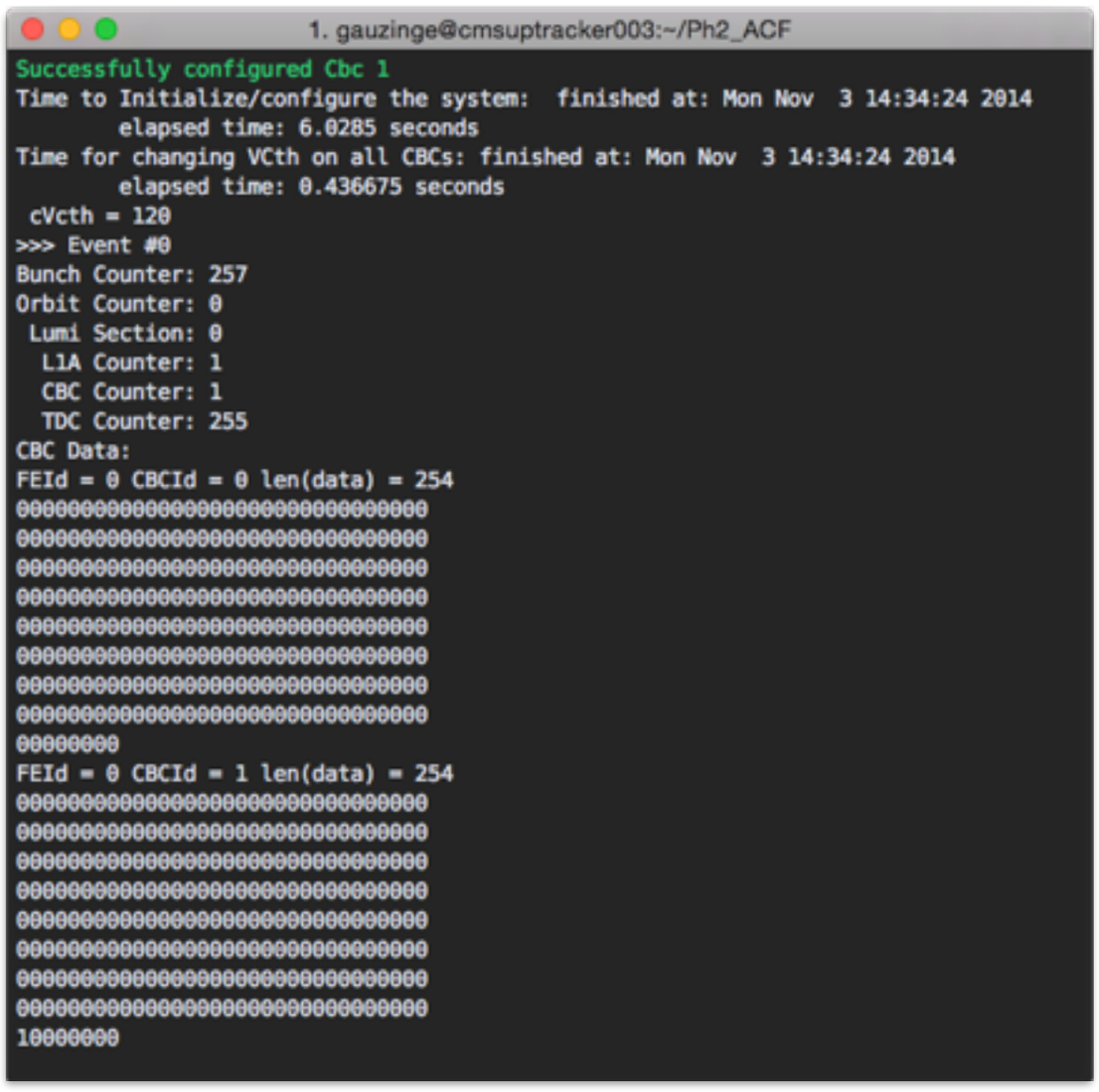
- are structured as .xml nodes
- contain the path to connections.xml for uHAL
- represent the hierarchy
- specify the path to the CBC config files
- can contain application specific settings
- validate file using systemtest application

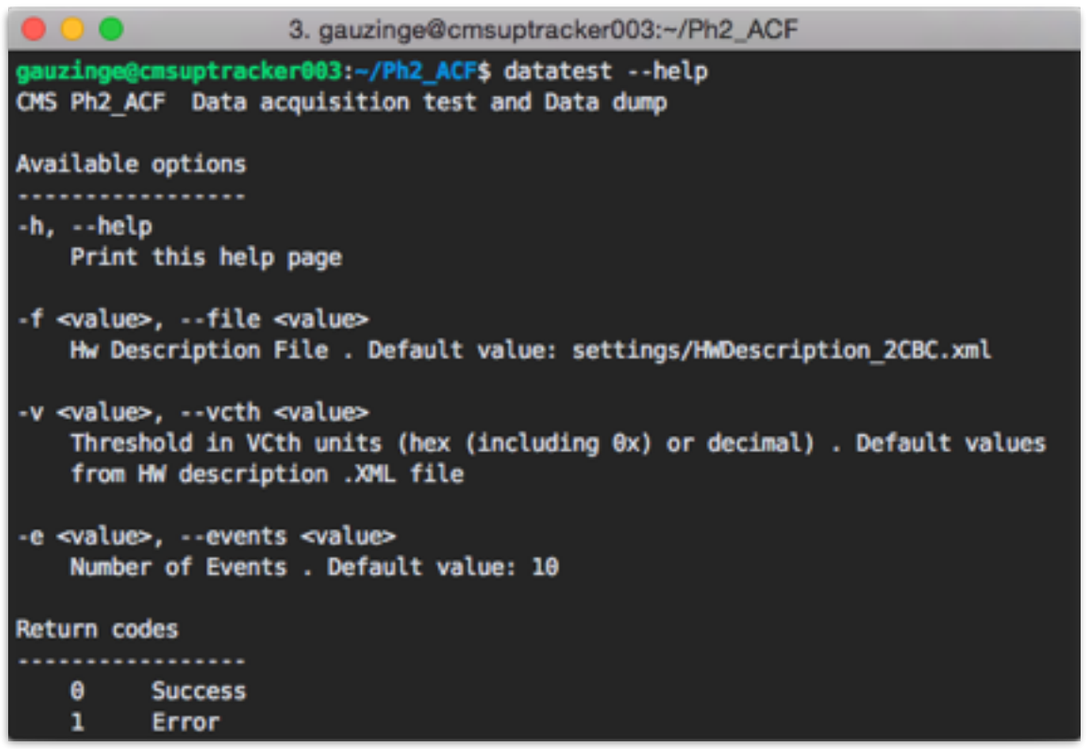
Higher level tools I

- all applications have command line parsing:

  applicationname -h (- - help)

  that shows the available options (see backup for summary)



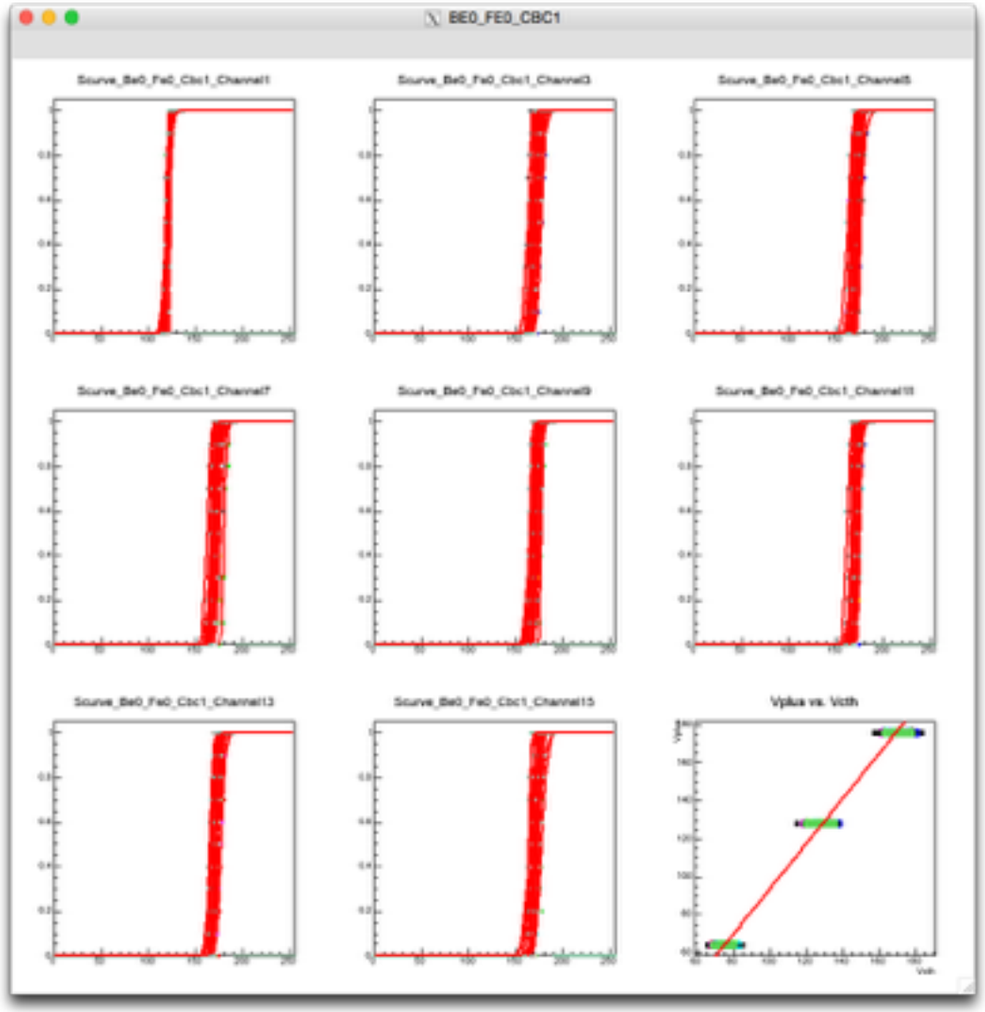


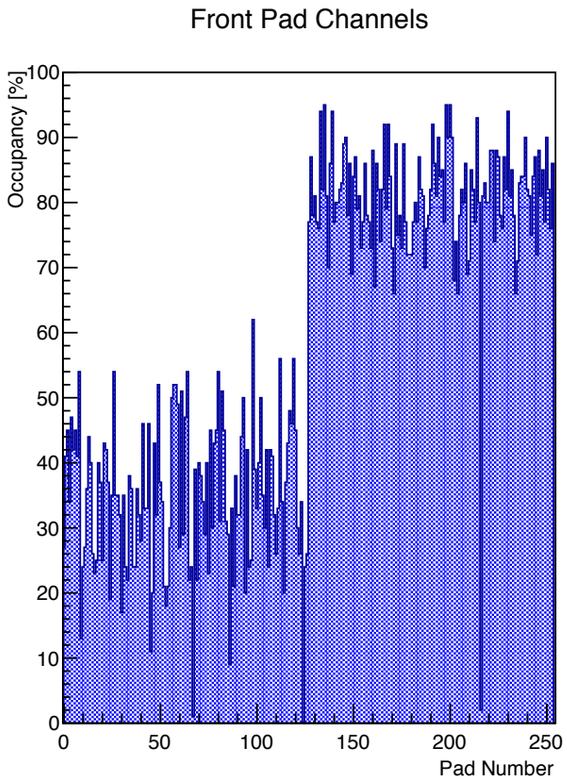
- systemtest / interfacetest: small apps that allow you to verify the format of your HWDescription.xml file, the functionality of the communication with the HW and the transaction speed

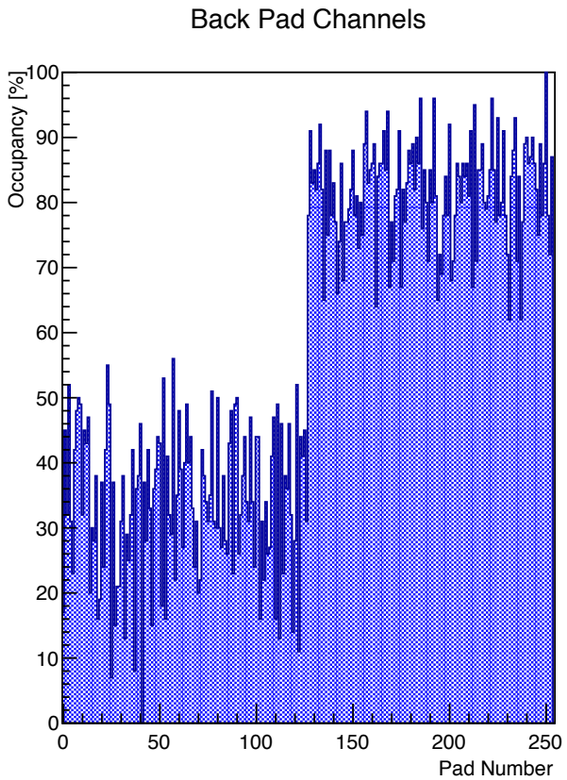
- datatest: allows you to read data and dump events to shell

Higher level tools II

- calibration: implementation of the calibration algorithm proposed by M. Raymond & K. Uchida
  - scan SCurves for various Vplus values at constant offset
  - extract Vplus setting for desired VCth setting
  - do a bit-wise tuning of per-channel offsets
- faster, more efficient version is being worked on (scan SCurves sweeping offsets as second step, …)



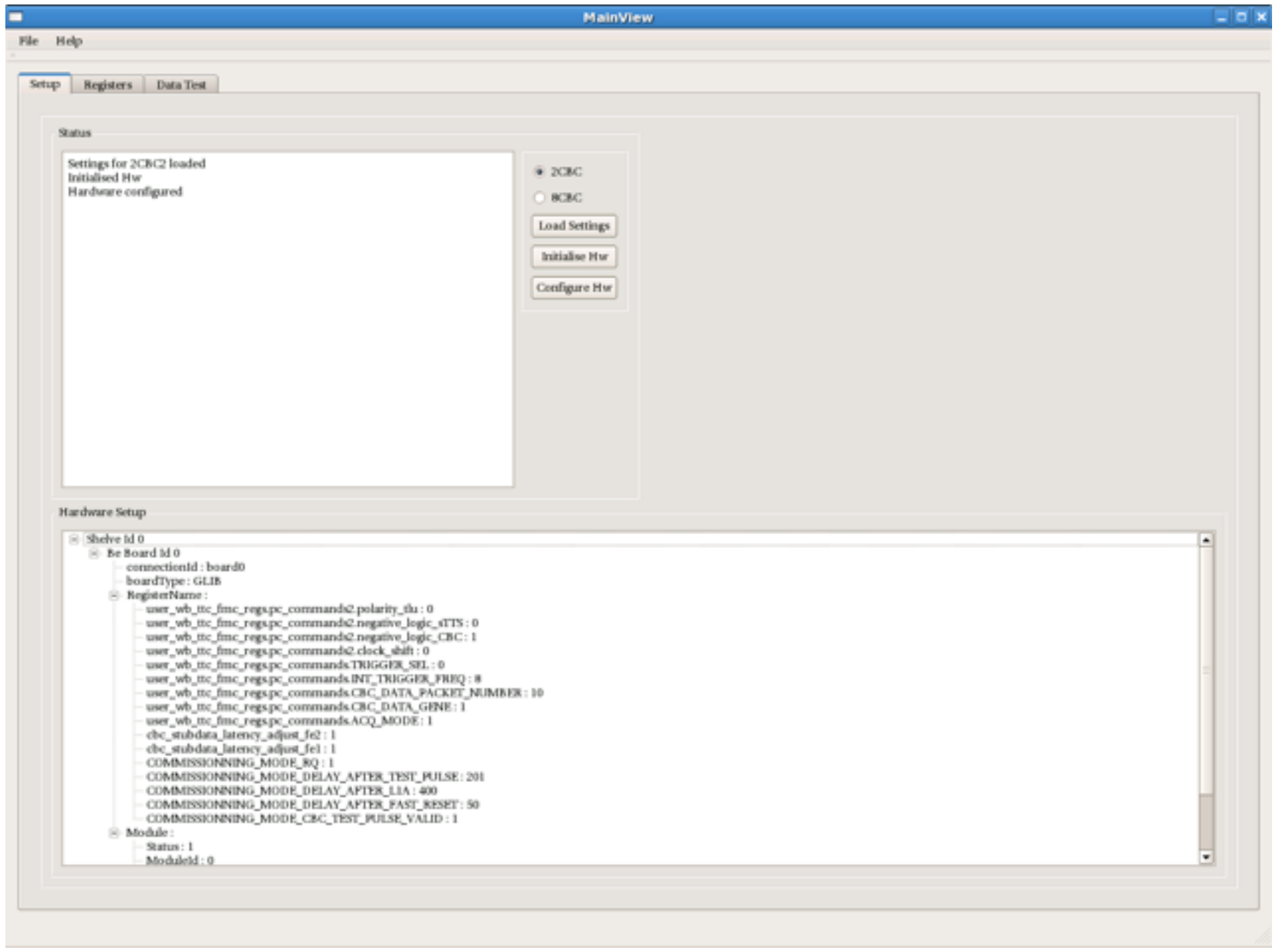




- hybridtest: specific for the hybrid validation proposed by A. Honma
  - validate CBC I2C registers by writing complimentary bit patterns 1 by 1
  - scan global efficiency to identify threshold with ~0 noise occupancy
  - measure the single-strip efficiency under the influence of an external signal induced on the bond-pads

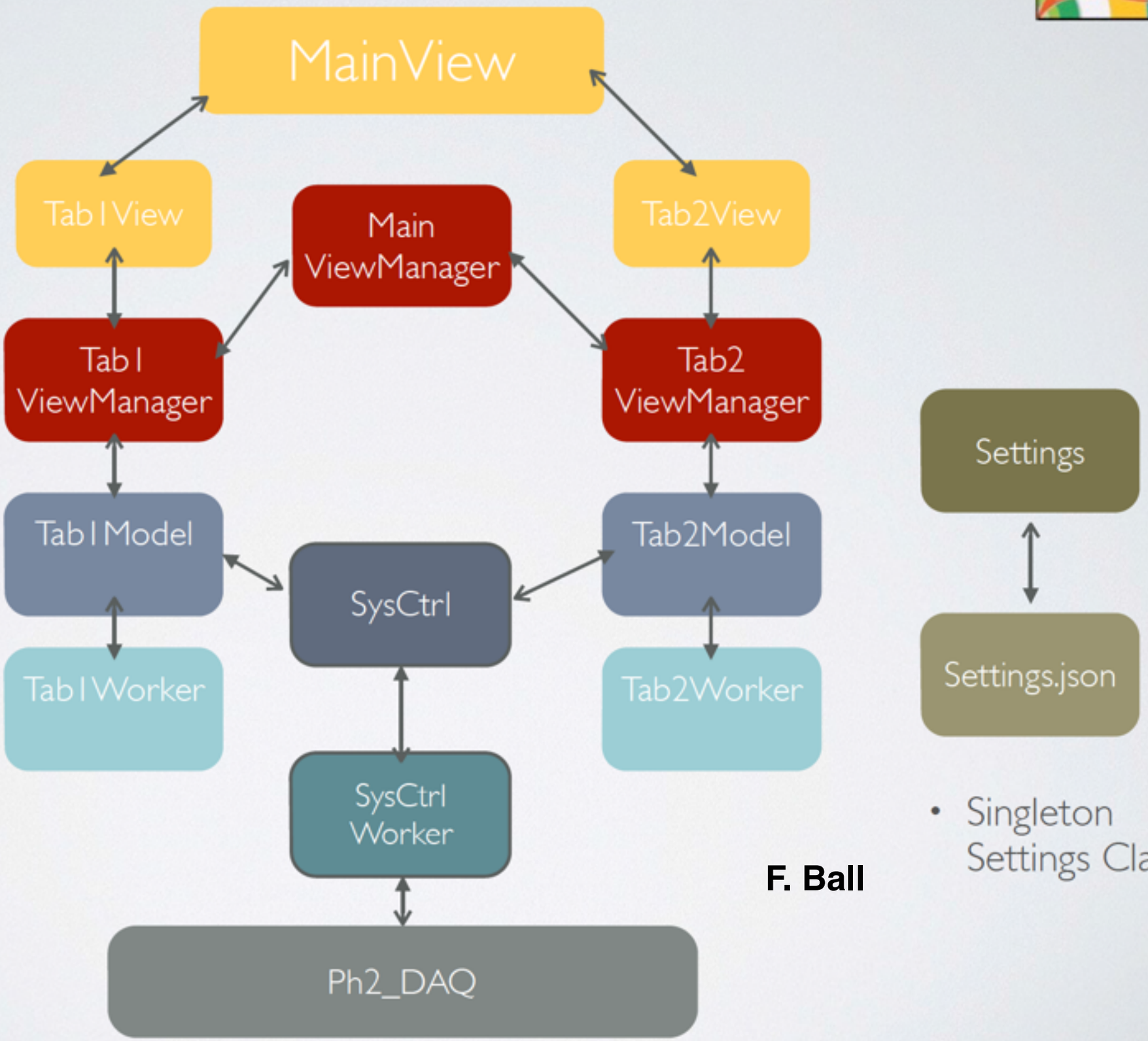
GUI

- Uses C++11 with the **QT-4** framework
- Interface to ROOT via QT-ROOT
- Parses JSON instead of .xml for easy hardware tree navigation
- Uses the Model/View design model
- currently porting command line scripts - hybridtest almost done!



F. Ball

PH2DAQ_GUI



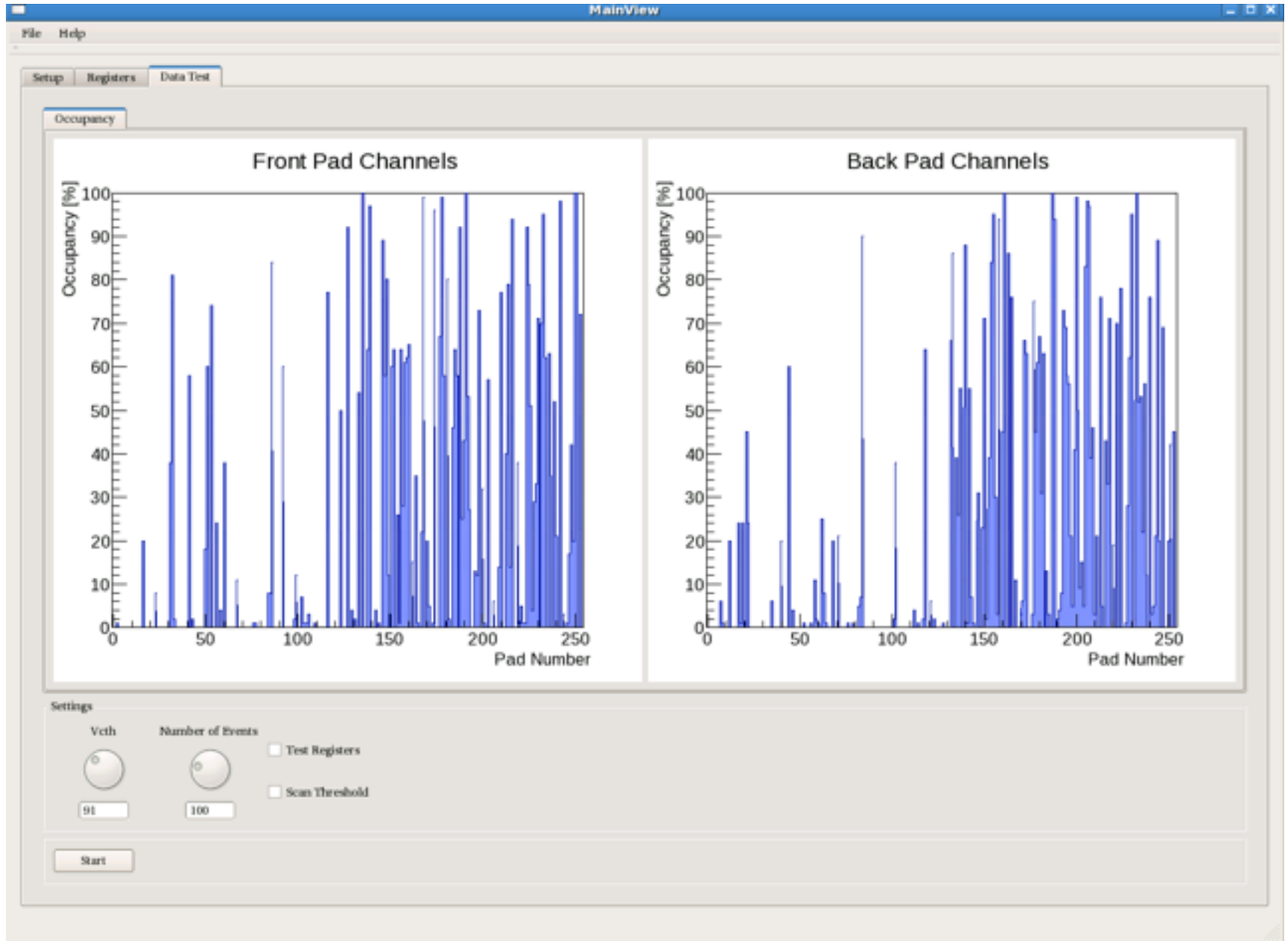
- TabWorkers are non persistent whereas the SysCtrlWorker is a persistent thread
- MainViewManager ensure no thread conflicts

- Singleton Settings Class

F. Ball

GUI Development Plans

- identify the best way to integrate CMD line scripts - without duplicating code in the threaded environment
- porting the calibration procedure
- Adding user save/ load methods
- Full 8CBC support



F. Ball

Tutorial: Looping the HW Structure

- many operations require identical operations on multiple BeBoards / Chips, …
- I2C broadcast not yet implemented in FW
- 2 ways of doing it:
  - **loops** (we use C++11 so auto keyword is your friend)
  - **vistors** (design principle): each HW object has an accept method that visits itself and all it's children
- visitors inherit from virtual void HWDescriptionVisitor::visit(HWObject&){}
- some default visitors provided (write/ read single registers,…)

1. fish /afs/cern.ch/user/g/gauzinge/Ph2_ACF

```
myloop()
{
        for ( auto& cShelve : fShelveVector )
        {
                for ( auto& cBoard : cShelve->fBoardVector )
                {
                        for ( auto& cFe : cBoard.fModuleVector )
                        {
                                for ( auto& cCbc : cFe.fCbcVector )
                                {
                                        // do whatever it is you do!
                                }
                        }
                }
        }
```

1. fish /afs/cern.ch/user/g/gauzinge/Ph2_ACF

```
myvisitor()
{
        // ad-hoc visitor
        struct SomeVisitor : public HWDescriptionVisitor
        {
                // re-implement visit() methods for the specific layers
                // I want to perform actions on
                void visit( BeBoard& pBeBoard ) {
                        somecode();
                }
                void visit( Cbc& pCbc ) {
                        somemorecode();
                }
        };

        // instantiate
        SomeVisitor myVisitor;
        // I inherit from System Controller which has accept()
        this->accept( myVisitor );
        // propagates through HW Hierarchy
        //  and performs action on BeBoards & Cbcs
}
```

Tutorial: Workflow

1. calibrate the hybrid:

    - modify your HWDescription.xml file to your likings: global settings, electron or hole mode, …

    - ./calibrate -f settings/Calibration2CBC.xml (- -fast in the future)— will take default files and settings from .xml file and calibrate the chips, dumping the SCurves and CBC config files in ./Results/Calibration_<Mode>_<Date&Time>/

2. modify your .xml file to point to the calibrated CBC config files

3. use these files for your own tests, data-acquisition with RU-BU-FU chain, …

4. write your own applications!!

Summary & Activities

- **Ph2 ACF available**, comes with libraries and some examples
- not many high-level applications available yet:
  - work on fast calibration, common mode test ongoing
  - can easily be adapted to do all kinds of commissioning procedures (latency scans, …)
- at the same time work on improving the middleware functionality:
  - continuous DAQ (now we start triggers, read data and stop triggers in a loop)
  - improving speed, efficiency and reliability
- obviously bug fixing
- advancing the GUI
- eventually port XDAQ application to use the MW

Backup - CMD line Options

systemtest:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -c - -configure: actually configure the HW

interfacetest:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -c - -configure: benchmark the configuration step
- -s - -single: benchmark single register transactions
- -m - -multi: benchmark multi-register transactions

datatest:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -v - -vcth: set the VCth for all chips (0x or decimal)
- -e - -events: number of events to read

calibrate:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -o - -output: directory where to store the results
- -s - -skip: skip the VCth vs Vplus scan
- -f - -fast: use fast calibration algorithm
- -bm - -bitmode: use bit-wise offset tuning
- -achan - -AllChan: calibrate all channels in parallel

hybridtest:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -o - -output: directory where to store the results
- -s - -scan: scan the noise occupancy
- -r - -registers: test all registers 1 by 1

upcoming

cmtest:
- -h - -help: show the list of options
- -f - -file: the HWDescription.xml file
- -o - -output: directory where to store the results
- -s - -scan: scan for noisy strips