# CMS Tracker DAQ Manual

| | |
|---|---|
| Author: | Christian Bonnin / L. Gross |
| Institute: | IPHC |
| Version: | 1.1.0 |
| Last revision date: | February 20, 2014 |

## Contributors

| Name | Affiliation |
|---|---|
| Christian BONNIN | IPHC |
| Laurent GROSS | IPHC |
| Laurent CHARLES | IPHC |
| Rémy BAUMANN | IPHC |
| Laurent MIRABITO | IPNL |
| Jérôme HOSSELET | IPHC |

# Table of Contents

# Overview

This project allows to run and configure a DAQ chain ranging from the front-end electronics (CBC chip) connected to a GLIB board up to data files written by the StorageManager module of CMSSW (via RU, BU, FU units). It takes place into the framework of the CMS Tracker upgrade developments.

Prior to using DAQ software modules, the appropriate firmware version must be programmed into the DAQ main board (GLIB). All firmware versions (sources as well as pre-compiled binaries) can be found on the project SVN repository.
SVN+SSH access:
svn+ssh://svn.cern.ch/reps/cmsptdaqup/CMS_UPGRADES_IPHC/FIRMWARE/PHASE_2/DAQ
WebSVN access:
https://svnweb.cern.ch/cern/wsvn/cmsptdaqup/CMS_UPGRADES_IPHC/FIRMWARE/PHASE_2/DAQ

The DAQ software modules source code is available on the project SVN repository.
SVN+SSH access:
svn+ssh://svn.cern.ch/reps/cmsptdaqup/CMS_UPGRADES_IPHC/SOFTWARE/PHASE_2/DAQ
WebSVN access:
https://svnweb.cern.ch/cern/wsvn/cmsptdaqup/CMS_UPGRADES_IPHC/SOFTWARE/PHASE_2/DAQ

The whole DAQ chain is pre-installed on a Virtual machine SLC 5.9 that can be downloaded at:
http://sbgcmstrackerupgrade.in2p3.fr/

It uses a middleware API layer called Ph2_ACF which wraps the firmware calls and handshakes into abstracted functions. It is hosted at the Gitlab repository:
https://gitlab.cern.ch/cmstkph2/Ph2_ACF.git

This DAQ chain is, from a software point of view, composed of different components

CBCDAQ, a XDAQ project written in C++. This project contains :

- GlibStreamer XDAQ package, to manage acquisition parameters and dialog with GLIB via IPBUS

- GlibSupervisor XDAQ package, to read/write GLIB parameters and visualize GLIB status flags

This project is the one checked out from SVN project repository into the directory of your choice. We will assume that this directory path is given by the environment variable $CBCDAQ that will be used in XML configuration files.

- An instance of the CMS run control, RCMS. This package includes RCMS tested configurations able to instanciate, configure and run all the modules of the complete DAQ chain:
  - GlibStreamer
  - GlibSupervisor
  - Event Manager (rubuilder::evm::Application)
  - TrackerManager
  - Read Unit (rubuilder::ru::Application)
  - Build Unit (rubuilder::bu::Application)
  - Event Processor (evf::FUEventProcessor)
  - Resource Broker (evf::FUResourceBroker)
  - StorageManager

Softwares installed in the virtual machine:

- CMSSW_6_2_0_SLHC2
  - ROOT 5.32
  - python 2.4.3
- XDAQ 11
- CACTUS 2.2.0 (https://svnweb.cern.ch/trac/cactus): manage IPBUS communications
- JobControl: visualize processes and their log

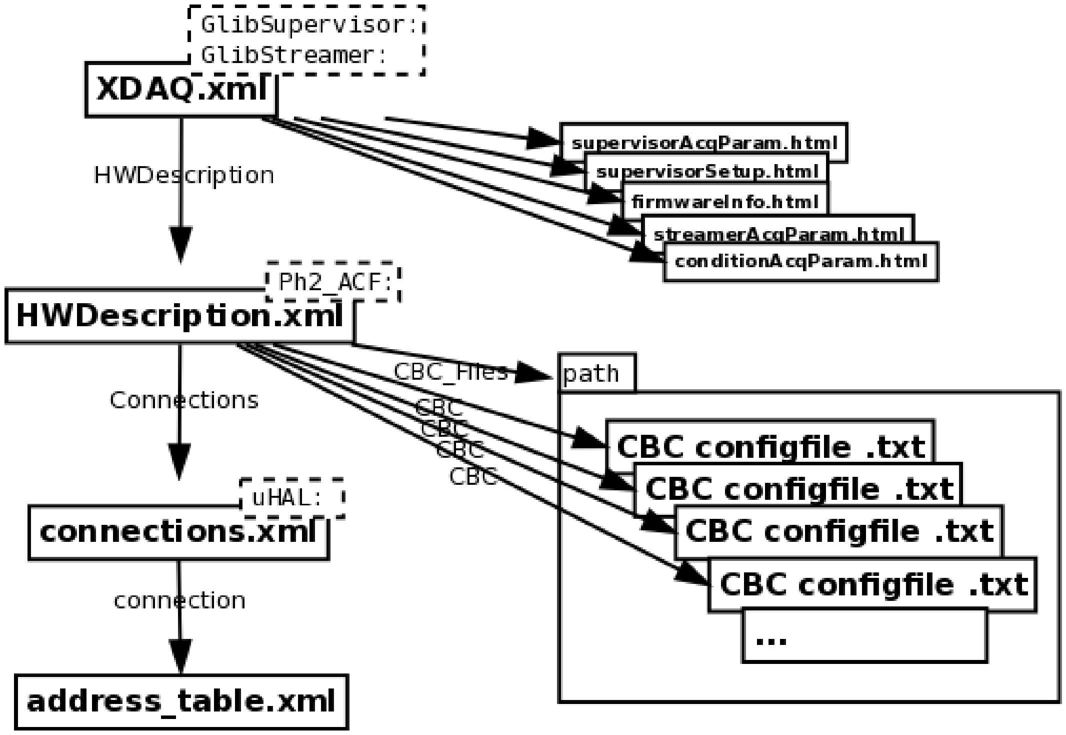
# uHAL and XDAQ configuration

The XDAQ packages use uHAL to connect to the boards. They are configured via a XDAQ XML file containing the following parameters:

- **HWDescription** contains the path to Ph2_ACF middleware hardware description XML file which contains amongst others the parameter
  - **Connections** which contains an URL to an uHAL XML connections file (It's usually an absolute path to the file starting with “file://”) which contains a
    - **connection** parameter for each board with its IP address and address file with all the parameter addresses.
  - **BeBoard** contains a board id to select one board number from the previous connections file.
- **ParamHtmlFile** is the path to the HTML form file used to read and write the acquisition parameter values that are stored in the board.
- **SetupHtmlFile** is the path to the HTML form file used to display GlibSupervisor global parameters
- **FpgaHtmlFile** is the path to the HTML form file used to show firmware information and upload a firmware into the FPGA

- **ConditionHtmlFile** is the path to the HTML form file used by the GlibStreamer to configure the condition data inserted into the phase-2 DAQ data files
- **RU** and **RUName** are used by the GlibStreamer to connect to the Reader Unit, first stage of the RUBUFU software chain.

Diagram of XML configuration files:



The connections file points to an uHAL XML address file which contains all parameter addresses and must match the firmware representation of these parameters. If not, IPBus exceptions may occur while accessing them.

The pycohal module allows to access the parameters with a python script. An example is given in :

```
$CBCDAQ/GlibSupervisor/script/uhalExample.py
```

## ***GLIB parameters set***

The addresses are in `$CBCDAQ/GlibSupervisor/xml/address_table.xml`

This file can be viewed using a navigator via the `param_table.xsl` XSL presentation.
The parameters set begins at the following address: **0x80000180**

# RCMS

RCMS is installed on the default VM instance.

Each time you will (re)boot the system, you will have to go through few steps before being able to use this runcontrol tool.

a) Check that the mysql daemon is up and running (ps ax |grep mysqld should issue a result line)

b) In a console, source the environment file needed to start apache. From home directory, type *source TomcatSourceBash/user.sh*

c) Start tomcat server. From home directory, type *StartTomcat*

d) Check that cmsuptracker003 is recognized as your machine name by pinging it. Type *ping cmsuptracker003*. If the host name is unknown, modify your /etc/hosts file to make sure it contains a line like “xxx.xxx.xxx.xxx        cmsuptracker003.cern.ch cmsuptracker003” where xxx.xxx.xxx.xxx is your IP address (to know your IP address, type */sbin/ifconfig* and read the “inet addr” line of the “eth0” interface). This has only to be done once.

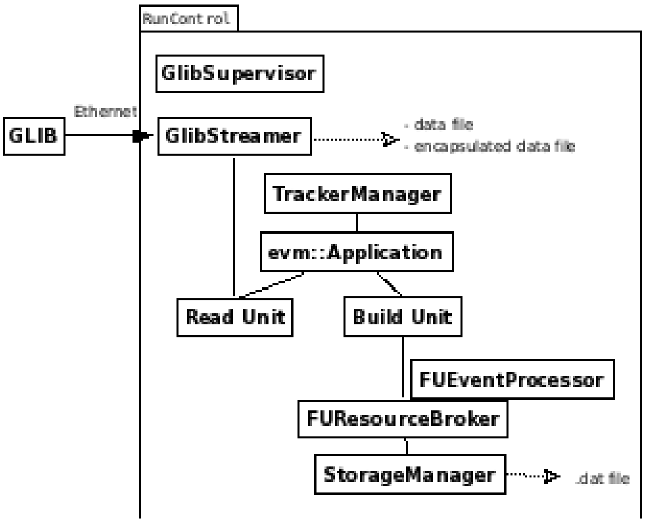
Open any web browser and go to URL : *http://cmsuptracker003:8080/rcms/*

This will bring you on the RCMS login page ; L/P is rcms/braze1

Then choose configuration GLIBDAQ/GLIB_DAQ_1MOD_1EP

At that point, you can run a full DAQ chain -up to the storage manager- controlled via RCMS.

Diagram of XDAQ packages:



The GlibStreamer can dump binary files that contain either only acquisition data or acquisition data encapsulated with Tracker, DAQ and FRL headers as it is sent to the Read Unit.

# GlibSupervisor

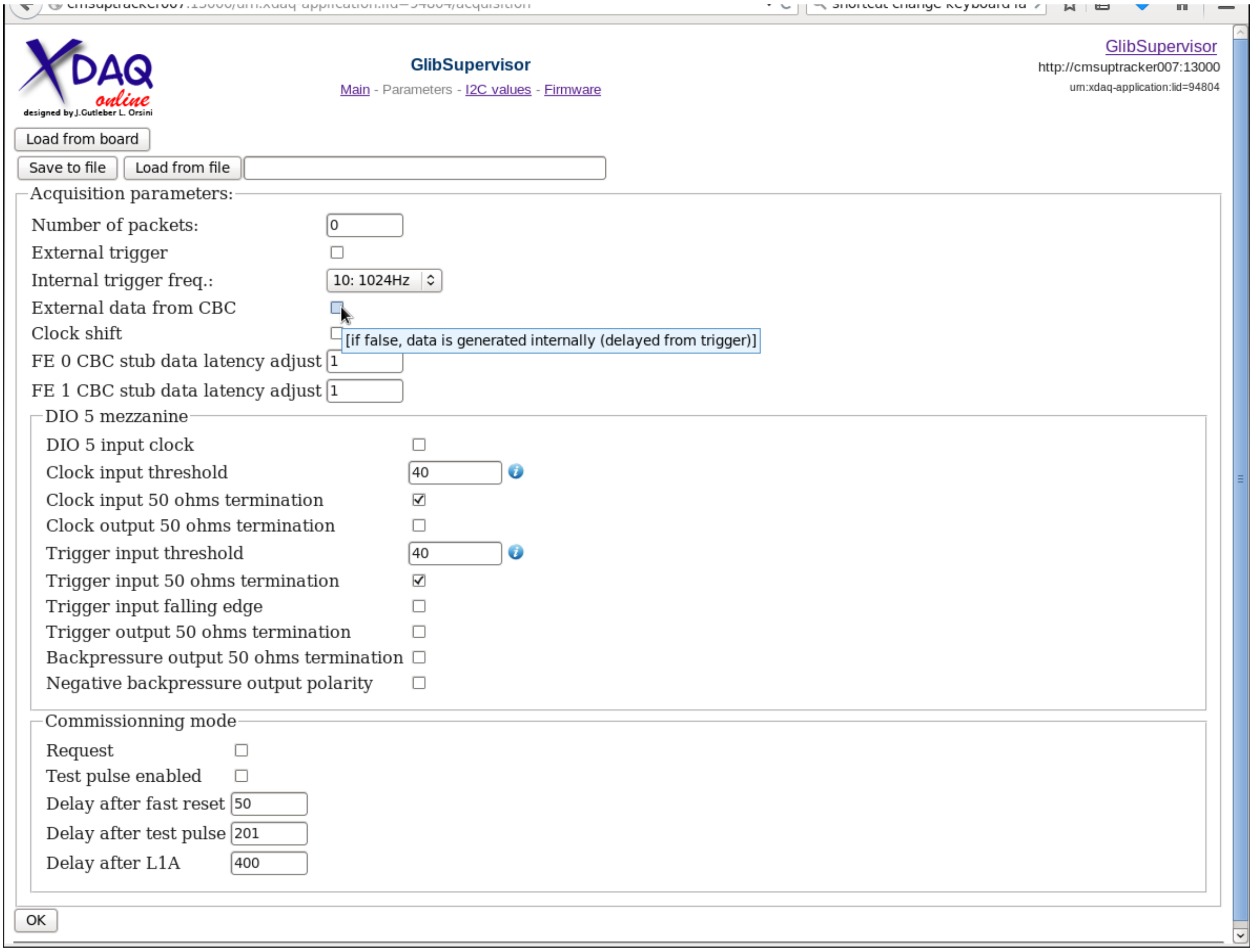
This XDAQ package

- displays the GLIB board status flags (main page),
- manages the acquisition parameters that are written in the board (parameters page),
- manages the CBCs parameter values (I2C values page),
- displays the firmware information and allows to upload a firmware file (firmware page).

## Main

A picture shows the position of FMC mezzanines depending on the number of CBCs configured in the middleware XML file

## Parameters



XDAQ online
designed by J.Gutleber L. Orsini
GlibSupervisor
Main - Parameters - I2C values - Firmware
GlibSupervisor
http://cmsuptracker007:13000
urn:xdaq-application:lid=94804
Load from board
Save to file
Load from file
OK

Acquisition parameters:

Number of packets: 0
External trigger ☐
Internal trigger freq.: 10: 1024Hz
External data from CBC ☐
Clock shift ☐
[if false, data is generated internally (delayed from trigger)]
FE 0 CBC stub data latency adjust 1
FE 1 CBC stub data latency adjust 1

DIO 5 mezzanine

DIO 5 input clock ☐
Clock input threshold 40
Clock input 50 ohms termination ☑
Clock output 50 ohms termination ☐
Trigger input threshold 40
Trigger input 50 ohms termination ☑
Trigger input falling edge ☐
Trigger output 50 ohms termination ☐
Backpressure output 50 ohms termination ☐
Negative backpressure output polarity ☐

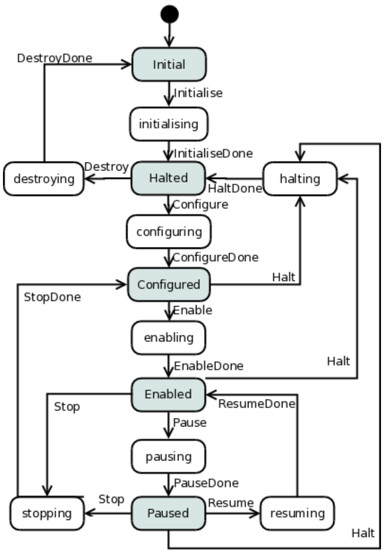
Commissionning mode

Request ☐
Test pulse enabled ☐
Delay after fast reset 50
Delay after test pulse 201
Delay after L1A 400

Explanations for parameters are given in tooltips.

Don't forget to click OK. The values will be written into the board when clicking the **Configure** button on the main page.

Is has the same FSM (Finite State Machine) as the Fed9USupervisor:



## *I2C parameters*

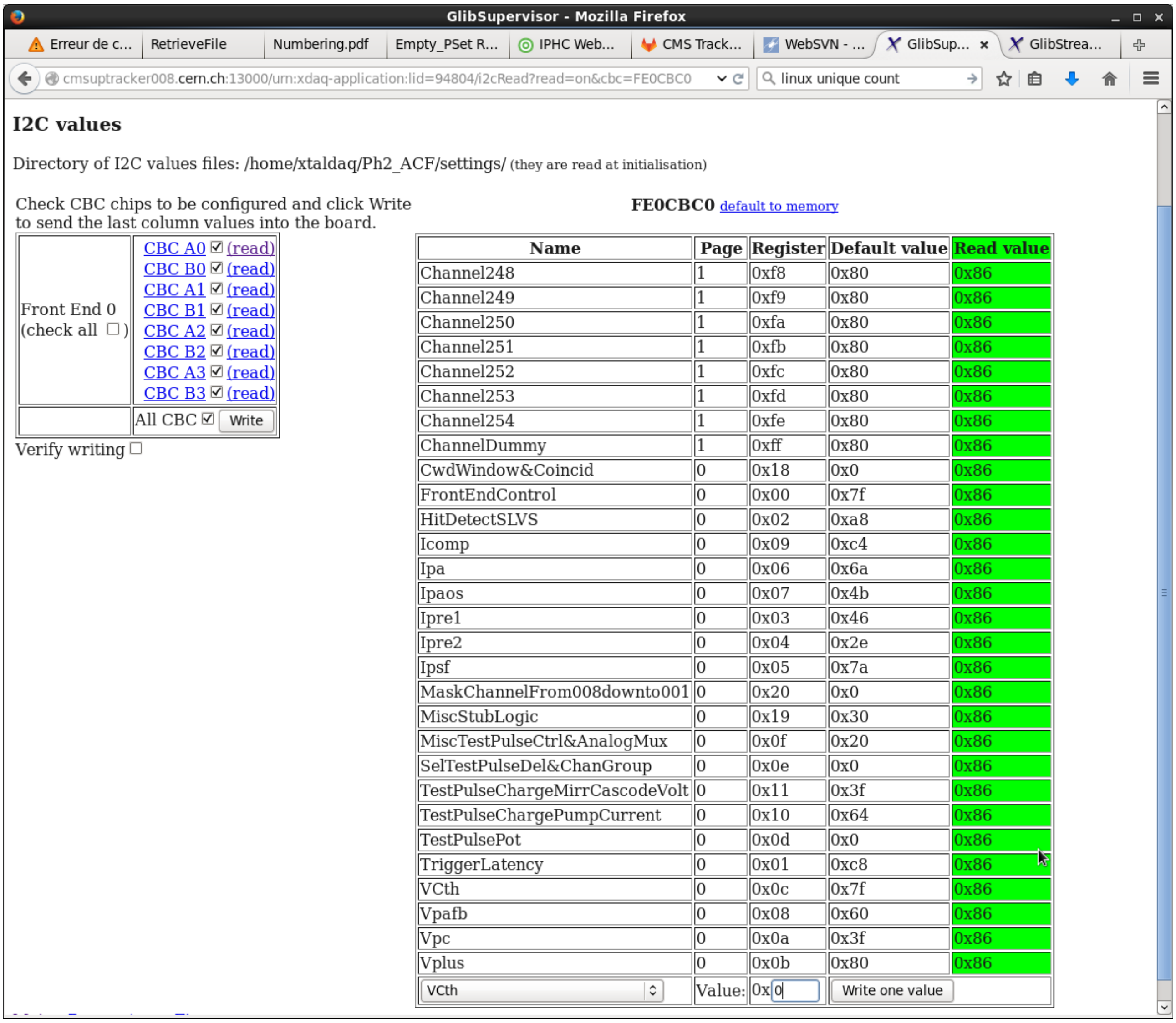
Two I2C buses are used:

- One for the core GLIB, its values are fixed
- One for CBC registers, its values and addresses can be found in a directory whose path is in the CBC_Files parameter of the middleware XML file (one txt file per CBC). For each CBC,
  - a link displays the values stored in memory by the middleware
  - a read link gets the values from CBC into memory and displays them
  - a checkbox selects it so that
    - all its value are written in it with the **Write** button
    - one value is written in it with the **Write one value** button (this button appears at the bottom of the page after the values are displayed)

To write only one value, display them all either by reading them from CBC or by displaying from memory, select its name, enter the value. The **Write one value** button writes it into the current displayed CBC or into those selected if there are any.

Remember that no values are written into CBCs until a **Write** button is pushed (there is no automatic I2C configuration).

The CBC values files are read by the GlibSupervisor at initialization, so if you want it to read them again, you can go to the main page and click **Destroy** then **Initialise**.



## I2C values

Directory of I2C values files: /home/xtaldaq/Ph2_ACF/settings/ (they are read at initialisation)

Check CBC chips to be configured and click Write to send the last column values into the board.

| | |
|---|---|
| Front End 0 (check all ☐) | CBC A0 ☑ (read)<br>CBC B0 ☑ (read)<br>CBC A1 ☑ (read)<br>CBC B1 ☑ (read)<br>CBC A2 ☑ (read)<br>CBC B2 ☑ (read)<br>CBC A3 ☑ (read)<br>CBC B3 ☑ (read) |
| | All CBC ☑ Write |

Verify writing ☐

**FE0CBC0** default to memory

| Name | Page | Register | Default value | Read value |
|---|---|---|---|---|
| Channel248 | 1 | 0xf8 | 0x80 | 0x86 |
| Channel249 | 1 | 0xf9 | 0x80 | 0x86 |
| Channel250 | 1 | 0xfa | 0x80 | 0x86 |
| Channel251 | 1 | 0xfb | 0x80 | 0x86 |
| Channel252 | 1 | 0xfc | 0x80 | 0x86 |
| Channel253 | 1 | 0xfd | 0x80 | 0x86 |
| Channel254 | 1 | 0xfe | 0x80 | 0x86 |
| ChannelDummy | 1 | 0xff | 0x80 | 0x86 |
| CwdWindow&Coincid | 0 | 0x18 | 0x0 | 0x86 |
| FrontEndControl | 0 | 0x00 | 0x7f | 0x86 |
| HitDetectSLVS | 0 | 0x02 | 0xa8 | 0x86 |
| Icomp | 0 | 0x09 | 0xc4 | 0x86 |
| Ipa | 0 | 0x06 | 0x6a | 0x86 |
| Ipaos | 0 | 0x07 | 0x4b | 0x86 |
| Ipre1 | 0 | 0x03 | 0x46 | 0x86 |
| Ipre2 | 0 | 0x04 | 0x2e | 0x86 |
| Ipsf | 0 | 0x05 | 0x7a | 0x86 |
| MaskChannelFrom008downto001 | 0 | 0x20 | 0x0 | 0x86 |
| MiscStubLogic | 0 | 0x19 | 0x30 | 0x86 |
| MiscTestPulseCtrl&AnalogMux | 0 | 0x0f | 0x20 | 0x86 |
| SelTestPulseDel&ChanGroup | 0 | 0x0e | 0x0 | 0x86 |
| TestPulseChargeMirrCascodeVolt | 0 | 0x11 | 0x3f | 0x86 |
| TestPulseChargePumpCurrent | 0 | 0x10 | 0x64 | 0x86 |
| TestPulsePot | 0 | 0x0d | 0x0 | 0x86 |
| TriggerLatency | 0 | 0x01 | 0xc8 | 0x86 |
| VCth | 0 | 0x0c | 0x7f | 0x86 |
| Vpafb | 0 | 0x08 | 0x60 | 0x86 |
| Vpc | 0 | 0x0a | 0x3f | 0x86 |
| Vplus | 0 | 0x0b | 0x80 | 0x86 |
| VCth | Value: | 0x 0 | Write one value | |

## *Firmware page*

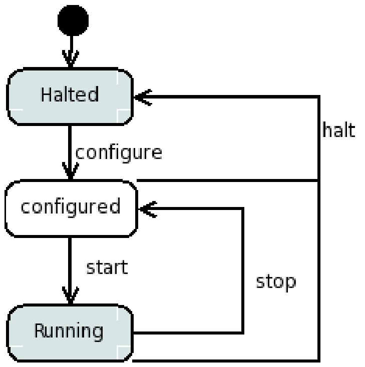
This page allows to upload an FPGA configuration (or firmware) into the GLIB board.

To configurations (golden and user) can be uploaded and it is possible to jump from one to another. The golden configuration is automatically loaded at startup.

# GlibStreamer

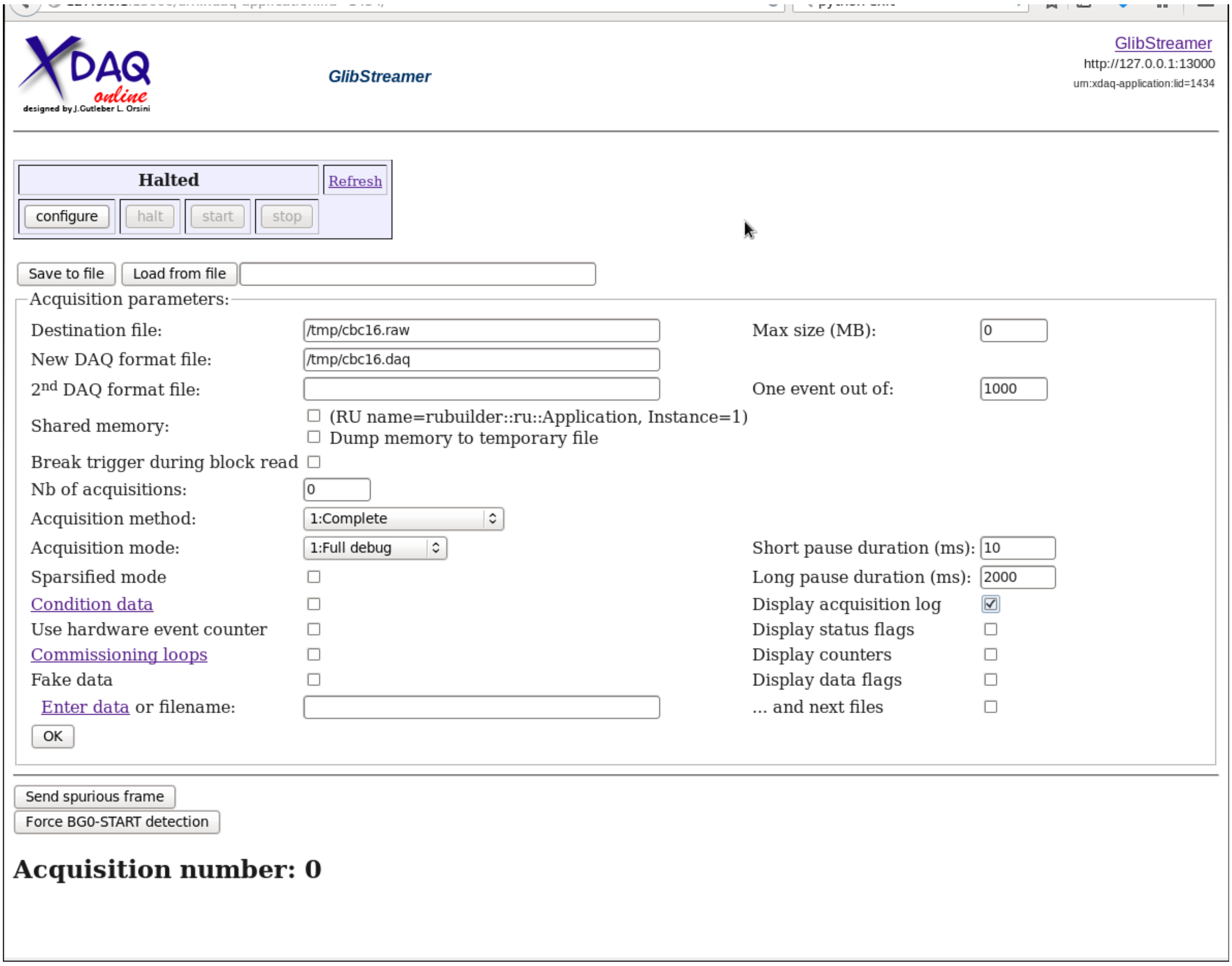
This XDAQ package drives the GLIB board to start and stop acquisitions and retrieve data that can be written to disk and/or passed to a Read Unit.

Its state machine is as follows:



The acquisition data can be generated to simulate the GLIB board.

It manages the acquisition parameters that are not written in the board.



GlibStreamer

GlibStreamer
http://127.0.0.1:13000
urn:xdaq-application:lid=1434

**Halted** Refresh

configure halt start stop

Save to file Load from file

Acquisition parameters:

| | | | |
|---|---|---|---|
| Destination file: | /tmp/cbc16.raw | Max size (MB): | 0 |
| New DAQ format file: | /tmp/cbc16.daq | | |
| 2nd DAQ format file: | | One event out of: | 1000 |
| Shared memory: | □ (RU name=rubuilder::ru::Application, Instance=1)<br>□ Dump memory to temporary file | | |
| Break trigger during block read | □ | | |
| Nb of acquisitions: | 0 | | |
| Acquisition method: | 1:Complete | | |
| Acquisition mode: | 1:Full debug | Short pause duration (ms): | 10 |
| Sparsified mode | □ | Long pause duration (ms): | 2000 |
| Condition data | □ | Display acquisition log | ☑ |
| Use hardware event counter | □ | Display status flags | □ |
| Commissioning loops | □ | Display counters | □ |
| Fake data | □ | Display data flags | □ |
| Enter data or filename: | | ... and next files | □ |

OK

Send spurious frame

Force BG0-START detection

**Acquisition number: 0**

| Parameters | Meaning |
|---|---|
| Destination file | Path to write acquisition data read from the board<br>Can be of text format (with .txt extension) or raw (format follows) |
| New DAQ format file | Read data formatted using the Phase-2 tracker data format |
| Acquisition mode | The Phase-2 tracker data format allows 3 modes:<br>• Full debug<br>• CBC Error<br>• Summary error |
| Zero Suppressed mode | If false, then the Virgin Raw operating mode will be used |
| Condition data | Condition data can be added to complete sent data |
| Shared memory | • Data is sent to the RU whose name is given by the XML configuration file<br>• Sent data can be dumped into a temporary file |
| Short pause duration (ms) | Wait for cbc_data_rdy |
| Long pause duration (ms) | Pause after initialization and configuration |
| Nb of acquisitions | Number of acquisition cycles to be performed (0 means endless acquisition). One cycles contains the number of events defined in GlibSupervisor Number of packets parameter |
| Use hardware event counter | Event number generated by the board (or else incremented by soft) |
| Fake data (or playback mode) | Acquisition data is generated by software or read from a file<br>Enter a “file name” with a .txt extension to read the read from an ASCII file, each line is an event, each strip is 0 or 1, or .raw to read a raw data file or .daq to a phase-2 format. |
| … and next files | If checked, files with the same name of the fake data file but with an extension .1, .2, … will be searched and read if found |
| Display<br>• acquisition log<br>• status flags<br>• counters<br>• data flags | Check these boxes to display additional information during acquisition. |
| Condition data | Each enabled condition data will be added into the New DAQ format file |
| Commissioning loop | Performs acquisition loops with varying I2C parameters |

Don't forget to click OK.

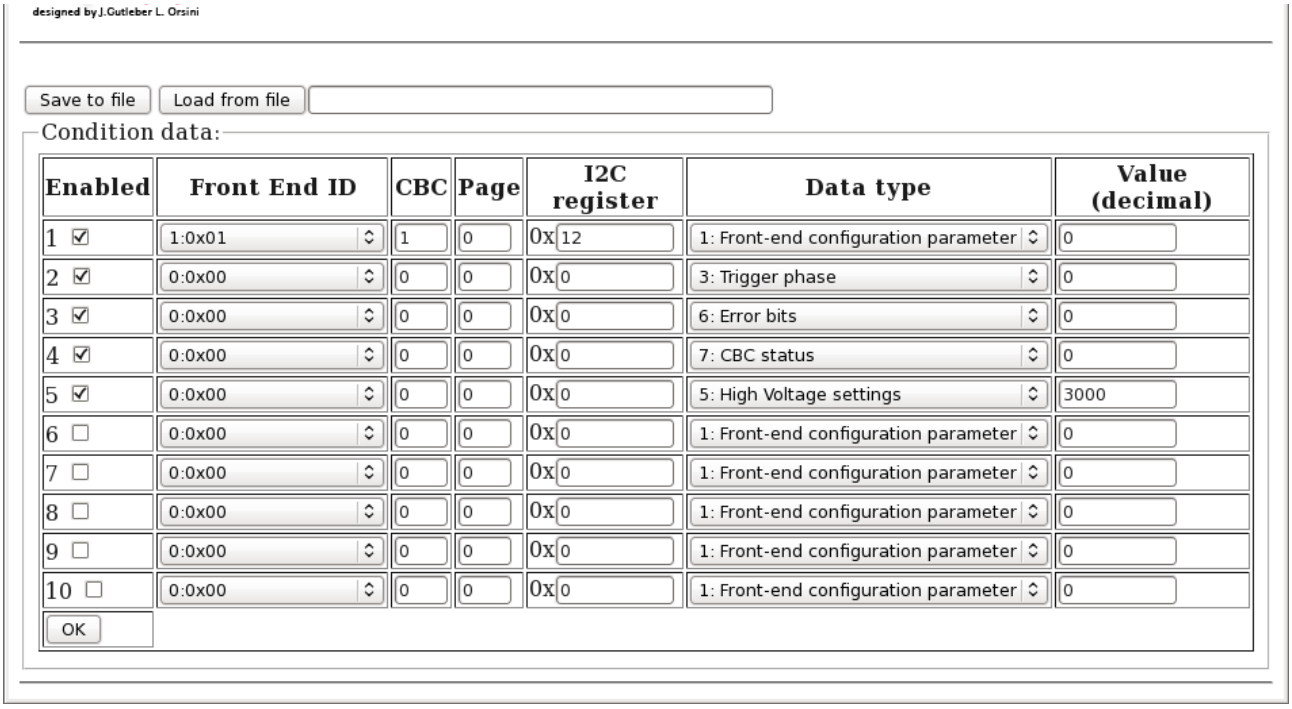
## Condition data

The condition data lines will be inserted into the Tracker-2 format .daq files generated from acquired data or raw data playback or simulated data but not from .daq file playback.

Each condition data line is an information that will be inserted into each event to the RUBUFU chain.

- The line has to be enabled
- The FE Id, CBC Id, Page and I2c register (address) are for the FE cofniguration parameter type. This I2C value is read from CBC before the acquisition (or each commissioning loop).

- The Trigger phase (TDC), Error bits, CBC status data types correspond to an information extracted from raw data.
- The other data types are constant values that have to be entered in the Value field.



designed by J.Gutleber L. Orsini

Condition data:

| Enabled | Front End ID | CBC | Page | I2C register | Data type | Value (decimal) |
|---|---|---|---|---|---|---|
| 1 ☑ | 1:0x01 | 1 | 0 | 0x 12 | 1: Front-end configuration parameter | 0 |
| 2 ☑ | 0:0x00 | 0 | 0 | 0x 0 | 3: Trigger phase | 0 |
| 3 ☑ | 0:0x00 | 0 | 0 | 0x 0 | 6: Error bits | 0 |
| 4 ☑ | 0:0x00 | 0 | 0 | 0x 0 | 7: CBC status | 0 |
| 5 ☑ | 0:0x00 | 0 | 0 | 0x 0 | 5: High Voltage settings | 3000 |
| 6 □ | 0:0x00 | 0 | 0 | 0x 0 | 1: Front-end configuration parameter | 0 |
| 7 □ | 0:0x00 | 0 | 0 | 0x 0 | 1: Front-end configuration parameter | 0 |
| 8 □ | 0:0x00 | 0 | 0 | 0x 0 | 1: Front-end configuration parameter | 0 |
| 9 □ | 0:0x00 | 0 | 0 | 0x 0 | 1: Front-end configuration parameter | 0 |
| 10 □ | 0:0x00 | 0 | 0 | 0x 0 | 1: Front-end configuration parameter | 0 |
| OK | | | | | | |

## Commissioning loops

The commissioning loop link shows an HTML form to enter the I2C parameter that will vary:

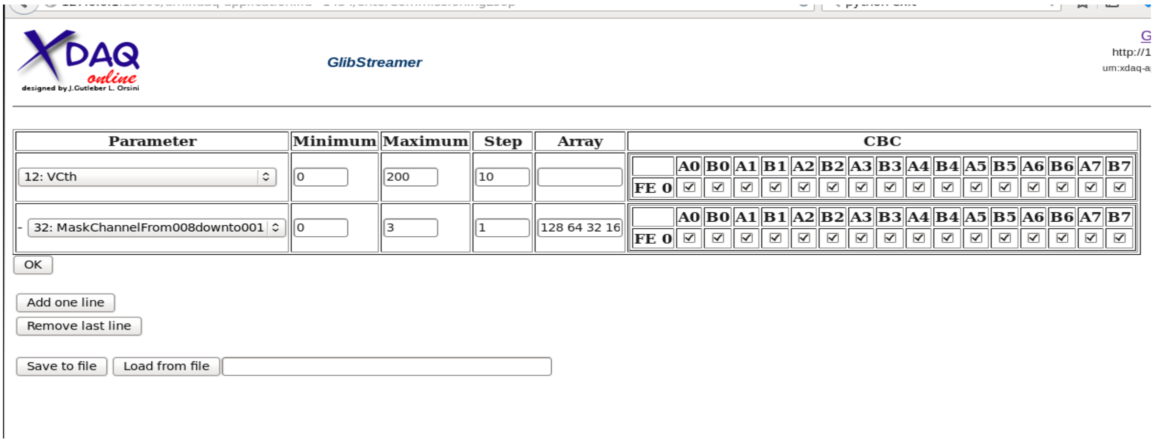
Attention: each line will be a nested loop or varying parameter.

If an array of values is specified (separated with spaces), the calculated parameter will be taken as an index in that array.

For each parameter, you can choose the CBCs that will be applied the I2C values.

Only enabled CBCs from the GlibSupervisor configuration can be checked.

For this reason the GlibStreamer has to be configured and it has to be after the GlibSupervisor is configured.



| Parameter | Minimum | Maximum | Step | Array | CBC | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | A0 | B0 | A1 | B1 | A2 | B2 | A3 | B3 | A4 | B4 | A5 | B5 | A6 | B6 | A7 | B7 |
| 12: VCth | 0 | 200 | 10 | | FE 0 | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| | | | | | | A0 | B0 | A1 | B1 | A2 | B2 | A3 | B3 | A4 | B4 | A5 | B5 | A6 | B6 | A7 | B7 |
| - 32: MaskChannelFrom008downto001 | 0 | 3 | 1 | 128 64 32 16 | FE 0 | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |

## ***Raw data format***

The raw data format from the board depends on the number of CBC chips.
The descriptions of this format for 4, 8, 16 CBCs can be found in the following files:

- $CBCDAQ/GlibSupervisor/doc/rawDataFormat4cbc.html
- $CBCDAQ/GlibSupervisor/doc/rawDataFormat8cbc.html
- $CBCDAQ/GlibSupervisor/doc/rawDataFormat16cbc.html

Note: 4 is the minimum, even if there are only 2 CBCs. In this case, two empty CBC data are added.

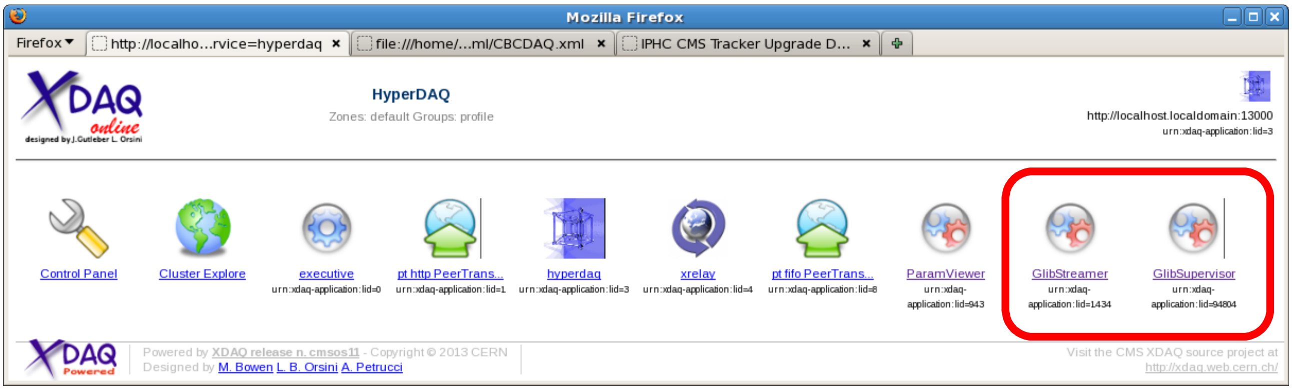
Data format from a GLIB acquiring from 4 CBC chips:

| | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Channel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | | | | | | | | | Bunch counter | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | Orbit counter | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | Lumisection | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | L1A counter | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | CBC counter | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Error bits | | CBC Status | | | | | | | | Channel data | | | | | | | | | | | | | | | | | | | | | |
| 6 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Stub data |
| 14 | Error bits | | CBC Status | | | | | | | | Channel data | | | | | | | | | | | | | | | | | | | | | |
| 15 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Stub data |
| 23 | Error bits | | CBC Status | | | | | | | | Channel data | | | | | | | | | | | | | | | | | | | | | |
| 24 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Stub data |
| 32 | Error bits | | CBC Status | | | | | | | | Channel data | | | | | | | | | | | | | | | | | | | | | |
| 33 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 34 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 35 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 36 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 37 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 38 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 39 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 40 | Channel data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Stub data |
| 41 | | | | | | | | | | | | | | | | | | | | | | | | | TDC | | | | | | | |
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |

# Simple acquisition

The XDAQ packages GlibSupervisor and GlibStreamer can be used independently of the RunControl to drive an acquisition.

Command line:


Mozilla Firefox
Firefox
http://localho...rvice=hyperdaq
file:///home/...ml/CBCDAQ.xml
IPHC CMS Tracker Upgrade D...
XDAQ online
designed by J.Gutleber L. Orsini
HyperDAQ
Zones: default Groups: profile
http://localhost.localdomain:13000
urn:xdaq-application:lid=3
Control Panel
Cluster Explore
executive
urn:xdaq-application:lid=0
pt http PeerTrans...
urn:xdaq-application:lid=1
hyperdaq
urn:xdaq-application:lid=3
xrelay
urn:xdaq-application:lid=4
pt fifo PeerTrans...
urn:xdaq-application:lid=8
ParamViewer
urn:xdaq-application:lid=943
GlibStreamer
urn:xdaq-application:lid=1434
GlibSupervisor
urn:xdaq-application:lid=94804
XDAQ Powered
Powered by XDAQ release n. cmsos11 - Copyright © 2013 CERN
Designed by M. Bowen L. B. Orsini A. Petrucci
Visit the CMS XDAQ source project at
http://xdaq.web.cern.ch/

```
 /opt/xdaq/bin/xdaq.exe -p 13000 -h localhost
      -c $CBCDAQ/GlibSupervisor/xml/GlibSuper.xml
```

The board name is given by `BoardName` tags.

In this case, you have to:

- initialize the GlibSupervisor
- If you want to modify the acquisition parameters written on the board,
  - go to the parameters page
  - enter the desired values
  - click OK
  - return to main page and click Configure
- On the GlibStreamer page,
  - Modify the desired GlibStreamer acquisition parameters
  - click OK
  - click the Configure then the Start button.

It is possible to automatize with the tag <**StartupSequence**> in the XML XDAQ configuration file (in the **properties** of the GlibSupervisor node, at the same level as **HWDescription**).

If this parameter contains the word **Initialisation**, the GlibSupervisor will initialize itself (connection to uHAL board), if it contains **Configuration**, the GlibSupervisor will configure (the acquisition parameter will be read from the HWDescription XML file), and if it contains **I2C** the CBC values will be written from CBC text files (and checked if the **Check** word is present).

This allows to Initialize the GlibSupervisor, configure and write I2C values without having to click on several buttons in an precise order.

## Step by Step manual to set up, from scratch, a VM+GLIB base system

Download VM instance from page: http://sbgcmstrackerupgrade.in2p3.fr/
Create new VM in VmWare according to instructions on http://sbgcmstrackerupgrade.in2p3.fr/
Run VM
Log in as xtaldaq/xtaldaq

**In VM**

```
cd /home/xtaldaq
mkdir firmware
cd firmware
```

In Firefox, in VM
Go to url https://svnweb.cern.ch/cern/wsvn/cmsptdaqup/Firmware/tags/tracker1.2.glibv3.dualcbc2/
Download “cdce62005_config” repository as */home/xtaldaq/firmware/ cmsptdaqupcdce62005_config.r123.tar.gz*

```
gunzip cmsptdaqup-cdce62005_config.r123.tar.gz
tar xvf cmsptdaqup-cdce62005_config.r123.tar
```

Now going OUT OF VM, because very likely your ISE is installed on your Windows system.

**Out of VM**

Create *c:\temp\firmware*
Open firefox
Go to
https://svnweb.cern.ch/cern/wsvn/cmsptdaqup/Firmware/tags/tracker1.2.glibv3.dualcbc2/virtex6_prog_files/BE/
Download glib_be.bit as c:\temp\firmware\glib_be.bit
Download glib_be.mcs as c:\temp\firmware\glib_be.mcs
Check that computer RJ45 plug is configured as 1Gbit/s full duplex with IP host 192.168.0.100
Take a brand new GLIB out of his box
Set jumpers to allow on desk work
Connect computer to the GLIB via an RJ45 Gbit/s compatible wire
Connect Xilinx programmateur to PC and GLIB
Power on GLIB
Via ISE/Impact, Flash the bin file c:\temp\firmware\glib_be.bit in GLIB EPROM (permanent) using constraints file c:\temp\firmware\glib_be.mcs
Power off the glib
Power on the glib
Check from native OS (Windows here) that one can ping the GLIB from a console : ping 192.168.0.175 -> ok
Back to the VM

**In VM**

Check from a console that one can ping the GLIB from the VM : ping 192.168.0.175 -> ok
Go to /home/xtaldaq/firmware/cdce62005_config.r123/BE/pychips_scripts and run

```
python glib_cdce_write.py
python glib_cdce_copy_to_eeprom.py
python glib_cdce_read.py
```

Power off/on the GLIB
Check that GLIB can be pinged from VM : ping 192.168.0.175 ▭ ok
At that point, DAQ should be working (refer to RCMS section of the current document for first steps) and the board can be accessed. All other/future problems should be, for 99% of them, configuration problems.