



# **Design and Implementation of web interfaces for $\mu$ TCA technologies**

André Ståhl - University of Costa Rica  
Stuart Szvec - The University of Manchester

Supervised by:  
Ulf Behrens  
Alan Campbell

Summer 2013

## **Abstract**

The LHC collides proton bunches at a rate of 40MHz, with each individual collision producing approximately 1 MB of data in the CMS detector. Thus to reduce the event rate to a more manageable amount a triggering system is used. This reduces the event rate to 100 kHz by discarding events that are not significant. In ongoing upgrades, the Level-1 Global Trigger is moving from VMEbus and being redesigned and implemented in  $\mu$ TCA technology. The purpose of the summer project working alongside the  $\mu$ TCA group, was to design web interfaces that will be used for monitoring the  $\mu$ TCA systems that will be used in the coming upgrade.

## Contents

1. The Compact Muon Solenoid detector and Hadronic Calorimeter	3
2. The move from VME to $\mu$ TCA	4
3. The $\mu$ TCA set up at DESY	5
4. The Web Interface	5
4.1 NATview	6
4.2 System Manager	6
4.3 Designing the interface and accessing data from the $\mu$ TCA hardware	7
4.4 The final web interface	9
5. GLIB Gui	11
5.1 Cactus and IPbus	12
5.2 Web Toolkit framework	13
5.3 The Gui	13
6. Acknowledgements	15
7. References	15

## 1. The Compact Muon Solenoid detector and Hadronic Calorimeter

The CMS detector is one of the two general purpose particle physics detectors built at the large hadron collider (LHC) at CERN. Its purpose is to investigate a wide range of physics, including Higgs Physics, extra dimensions and much more. It will do this by measuring muons, electrons and photons with a large range of energy, determining the signatures of quarks and gluons from jet measurements. It can also measure missing transverse energy, which as a result enables the identification of non-interacting particles such as neutrinos and potentially any new particles to be identified<sup>1</sup>.

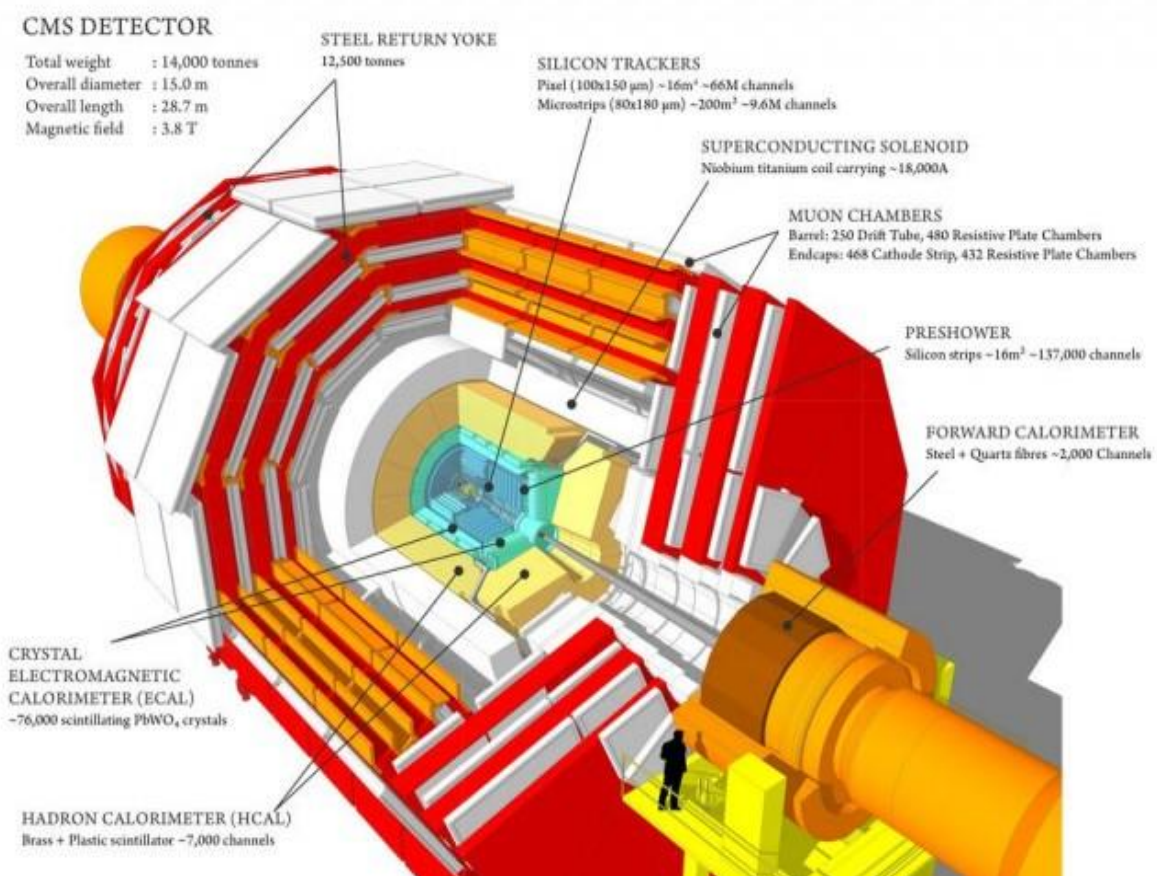


Figure 1- The CMS detector<sup>2</sup>

The  $\mu\text{TCA}$  group has been working on upgrading the electronics of the Hadronic Calorimeter (HCAL) of the CMS detector, specifically the slow control readout of the electronics. The calorimeter system inside CMS will measure quark, gluon and neutrino energies and directions of travel by determining the direction of different jets and the missing transverse energy. It is this determination of missing energy that will form part of the signature of new undiscovered particles.

## 2. The move from VME to $\mu$ TCA

The electronics of the hadronic calorimeter are in two separate stages. There are the frontend electronics, these are located on the detector itself in the experiment cavern (UXC) and have to be radiation hardened to ensure long lasting use (10+ years). The other part of the electronics is the backend, which is located in a nearby service cavern (USC). The digital signals from the frontend are transferred to the backend via 100 metres of optical cables.

It has been decided that the backend electronics will move from the old VME to the  $\mu$ TCA technology. The motivation for the upgrade is that it will minimise the potential of having maintainability and reliability problems due to the lack of redundant connections between different modules and the incapability of the old communication standard that has been used in the electronics infrastructure.

VME has been used for over 35 years, and while it has proven to be a very suitable platform for HEP experiments, in recent years it has struggled to cope with the data transfer rates.  $\mu$ TCA on the other hand is now the industry standard and offers high speed data transfer rates through the backplane and redundant connections with star topology. Star topology is a common network topology which in its simplest form consists of one central hub, to which all other nodes are connect, acts as a conduit to transmit messages. The benefit of using  $\mu$ TCA, with the crate control hubs, there are far better cooling options and power management configurations.

$\mu$ TCA is based on the AMC (Advanced Mezzanine Card) standard developed by the PCI Industrial Computer Manufacturers Group (PICMG) for ATCA cards. The  $\mu$ TCA backplane can hold up to 12 AMC cards with a MCH ( $\mu$ TCA Carrier Hub) providing connectivity between different slots on the backplane, although direct connections between slots are allowed also<sup>3</sup>.

### 3. The $\mu$ TCA set up at DESY

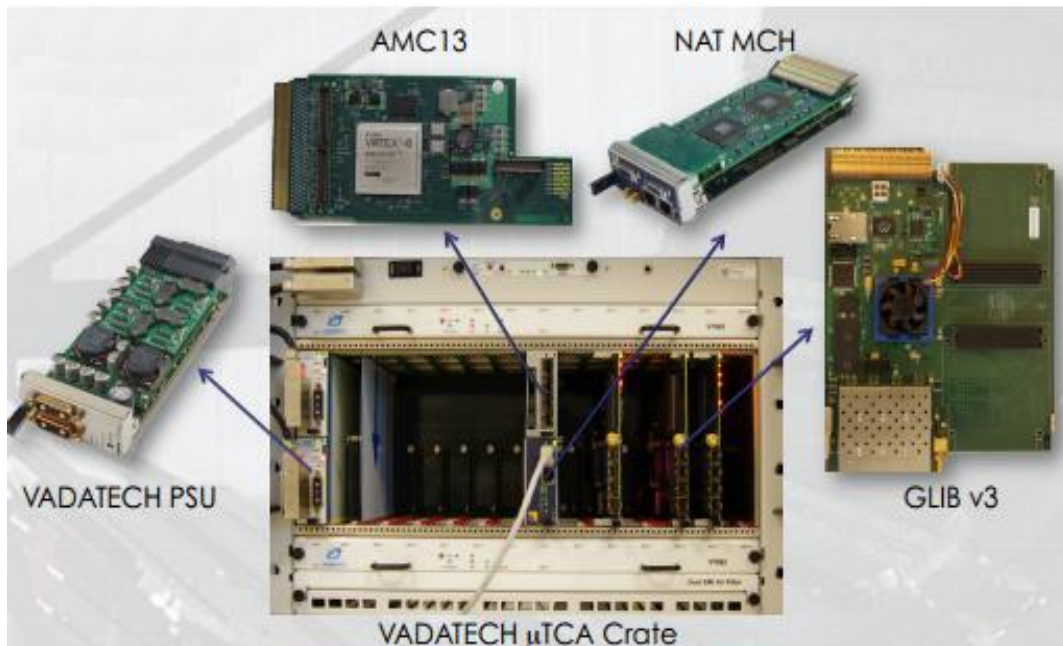


Figure 2 - The  $\mu$ TCA crate and cards at DESY

The  $\mu$ TCA system is encapsulated by the Vadatech crate. In the crate there are 2 power units (PSUs), 2 cooling units with one above and one below the central slots. There are 2 types of cards used in the crates. The first is the GLIB card. The GLIB (Gigabit Link Interface Board) is a prototype card developed by the CERN electronics group. It is an evolution platform and is used for high speed optical links in high energy particle physics experiments. The other card is the MCH, which is responsible for management of the crate and providing Ethernet link which enables access to modules in the crate through the backplane.

### 4. The web interface

The motivation for the design of the web interface is simple. Normally to access the crates and to retrieve desired information, the user must be connected to a specific terminal that is connected to the network. Whereas with a web based interface, the user can access the required crates on any device in any location providing they are connected to the network that the crates are connected to.

## 4.1 NATview

In the beginning of the project, a program called NATview was used to access information about the crates. NATview is an easy to use visual tool for any  $\mu$ TCA system that includes a NAT-MCH. The software is operating system independent and runs on any host computer internal or external to the  $\mu$ TCA system. NATview allows the user to view the  $\mu$ TCA crates and process the information from the crates in a graphical way.

This is done via a connection with the MCH card. The protocol for the connection is RMCP or Remote Management Control Protocol which is requested by the  $\mu$ TCA specification.

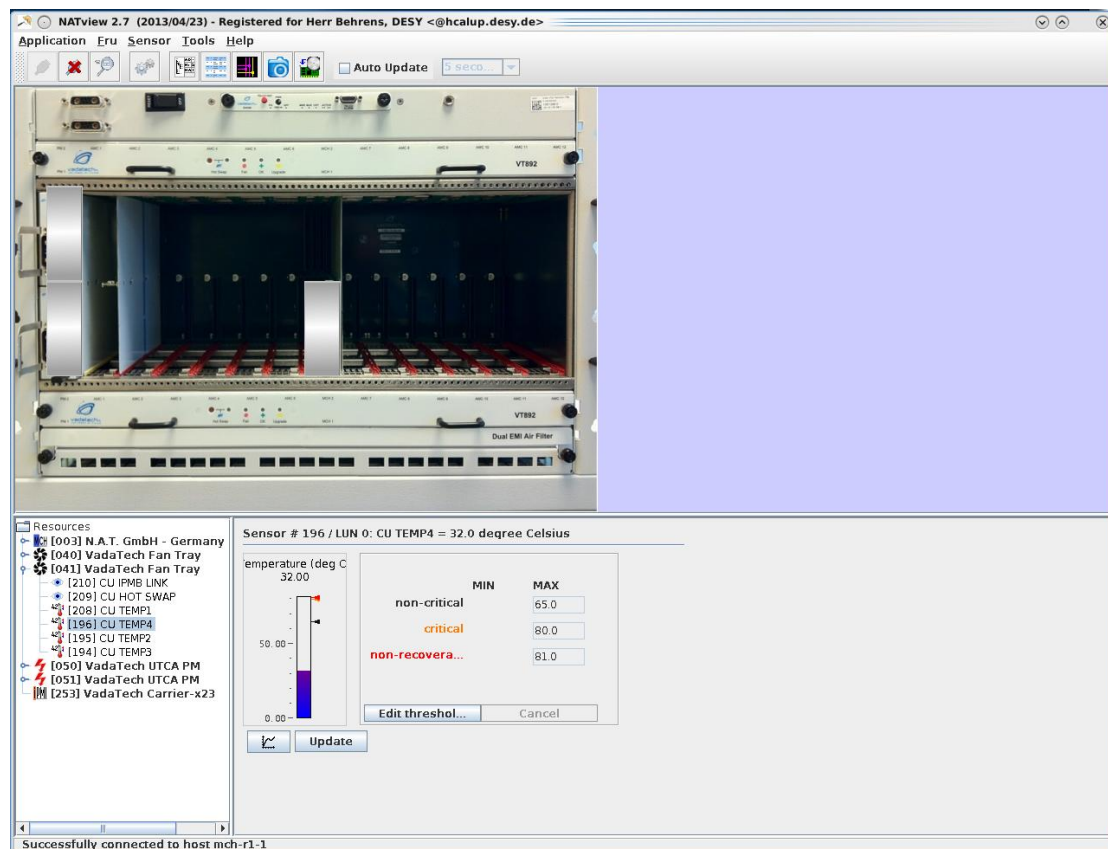


Figure 3 - The NATview interface<sup>4</sup>

## 4.2 System Manager

To begin with a program called system manager was used. It was this C++ library that accesses the crates and collects the desired information. It provides low-level hardware management services based on the Intelligent Platform Management

Interface (IPMI), which is a standardised computer system interface for easy management of servers.

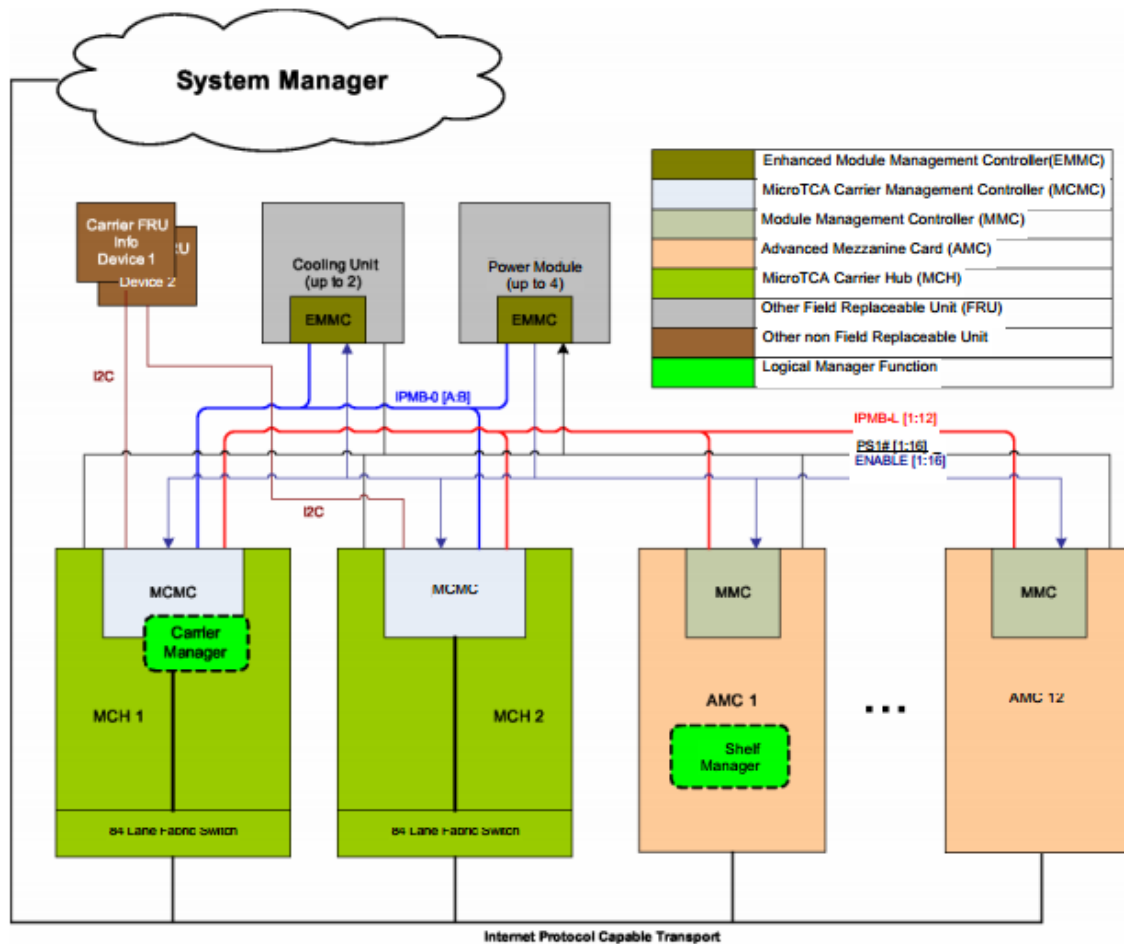


Figure 4 - System Manager access tree<sup>5</sup>

The  $\mu$ TCA shelf manager can manage up to 16  $\mu$ TCA carriers that comprise a  $\mu$ TCA shelf. Each individual carrier manager interfaces to the shelf manager using an IP-based interface. The shelf manager itself is a management function and can be implemented on any Field Replaceable Unit (FRU) and it watches over all hardware in the crate, such as AMC's, MCH's, power modules and cooling units.

#### 4.3 Designing the interface<sup>6</sup> and accessing data from the $\mu$ TCA hardware

The web interface was designed using a program called KompoZer. KompoZer is a WYSIWYG (What You See Is What You Get) web editor, very similar to Dreamweaver. It is designed to be extremely easy to use, and as such is ideal for non-technical users to create professional looking web interfaces without the need of extensive HTML knowledge.



After the design of the web interface, it had to be linked to the System Manager program. Firstly a web server was required, and eventually Apache was chosen and implement on the hcalup server. In the start the user sends a request through the webpage for information. This is handled by the Apache web server. Then with a CGI library, sysmgr.cgi is accessed (which is a C++ script). A CGI file (Common Gateway Interface) is a way of transferring information between a web server and another program. In our case the system manager program. Then depending on what information was requested, one of 7 different functions is called. The functions are as follows:

- LIST\_CARDS
- LIST\_CRATES
- LIST\_SENSORS
- GET\_CRATE\_INFO
- GET\_CARD\_INFO
- READ\_SENSOR
- GET\_SENSOR\_THRESHOLD

Then using the system manager program, the desired information is accessed from the MCH card in the crate. The information is then packeted as a JSON (JavaScript Object Notation - required use of a JSON C++ library) string and is then returned back to the Apache web server and turned back to HTML and displayed on the user's browser.

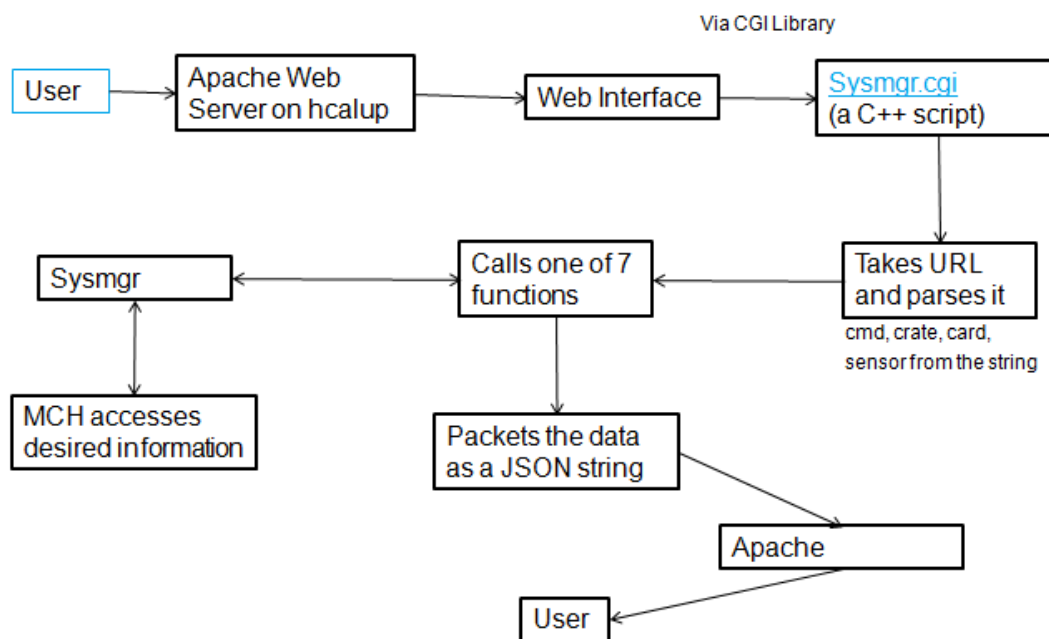


Figure 5 -Flow Diagram of data acquisition



#### 4.4 The final web interface

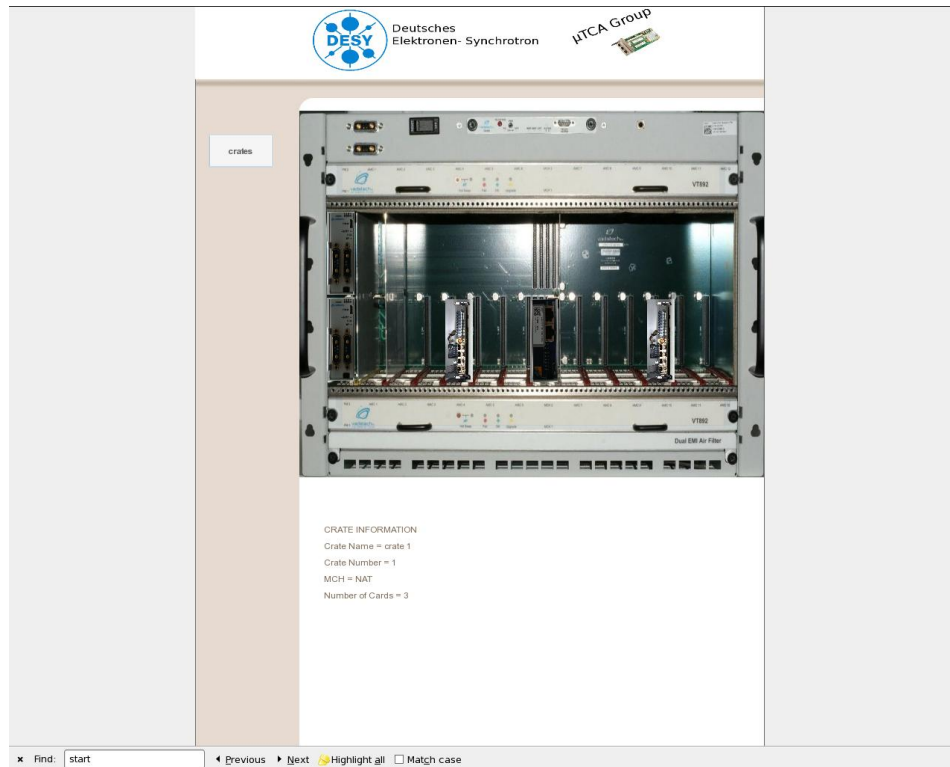
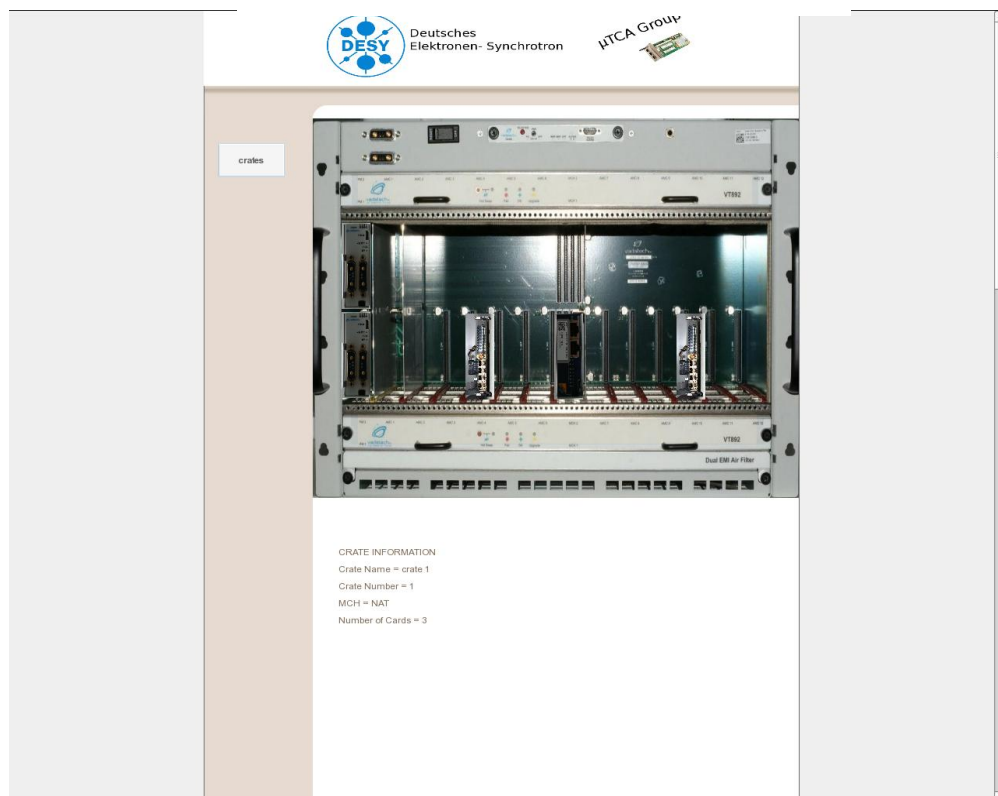


Figure 6 - The Interface



9  
Figure 7 - Selecting a crate

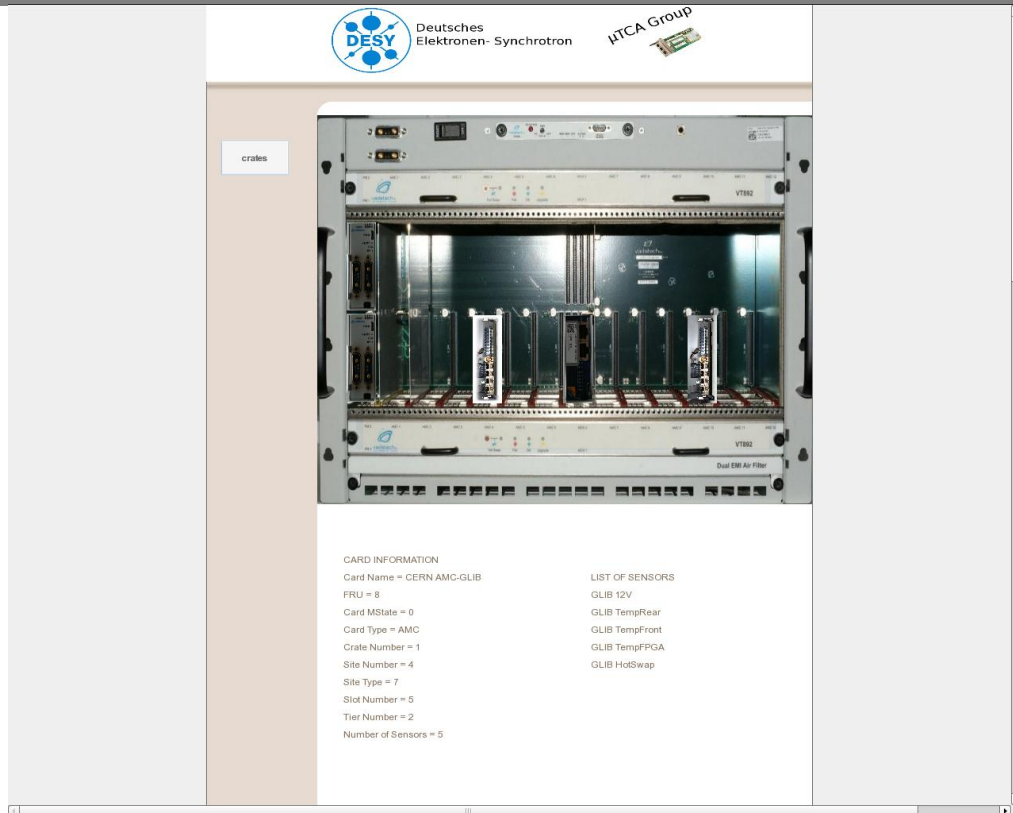
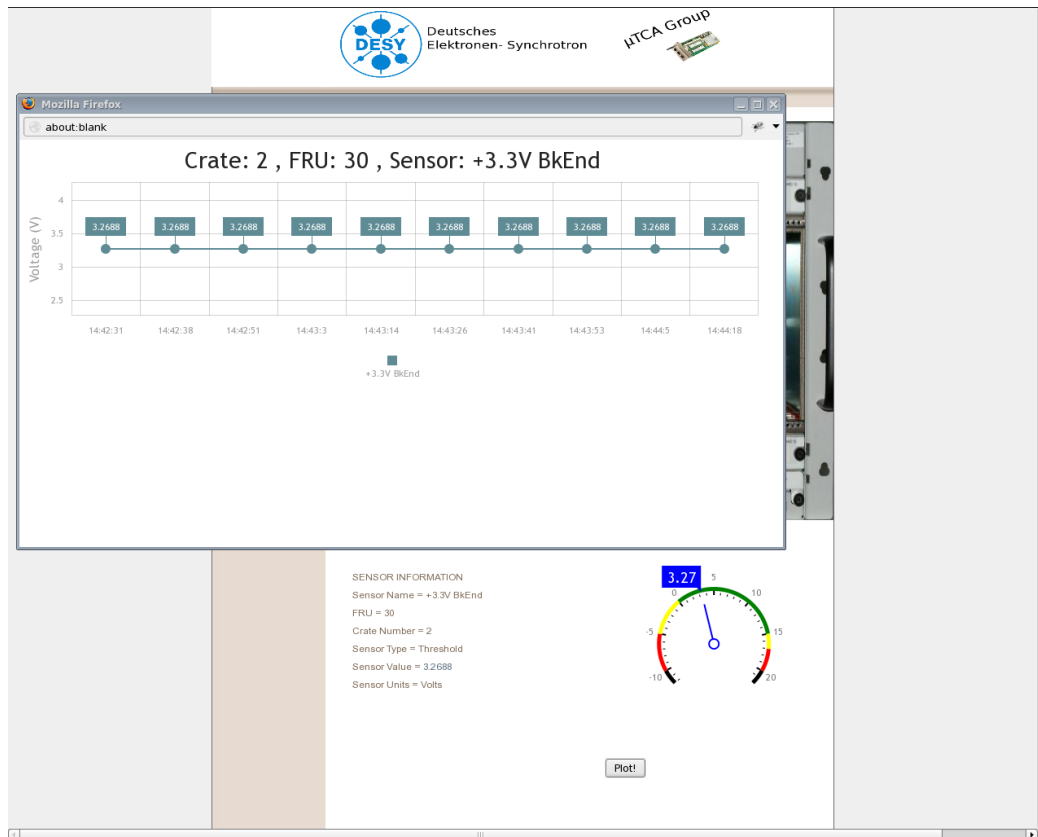


Figure 8 - Displaying Card information



Figure 9 - Displaying Sensor Information in a linear gauge



**Figure 10 - Plotting the last 10 sensor values**

Initially the webpage was designed to be static, but eventually it was made to refresh every 10 seconds. As a result it would update what cards were connected to crates and change the sensor values accordingly.

## 5. GLIB Gui<sup>7</sup>

The second stage of the summer project revolves around a program called GLIB Gui. The GLIB Gui is a graphical user interface for the control of the GLIB cards in the  $\mu$ TCA crates and it has been implemented almost entirely in Java. At the most basic level, it exchanges user datagram protocol (UDP) messages with the hardware via a Java class. The format of these UDP packets is defined by the IPbus standard.

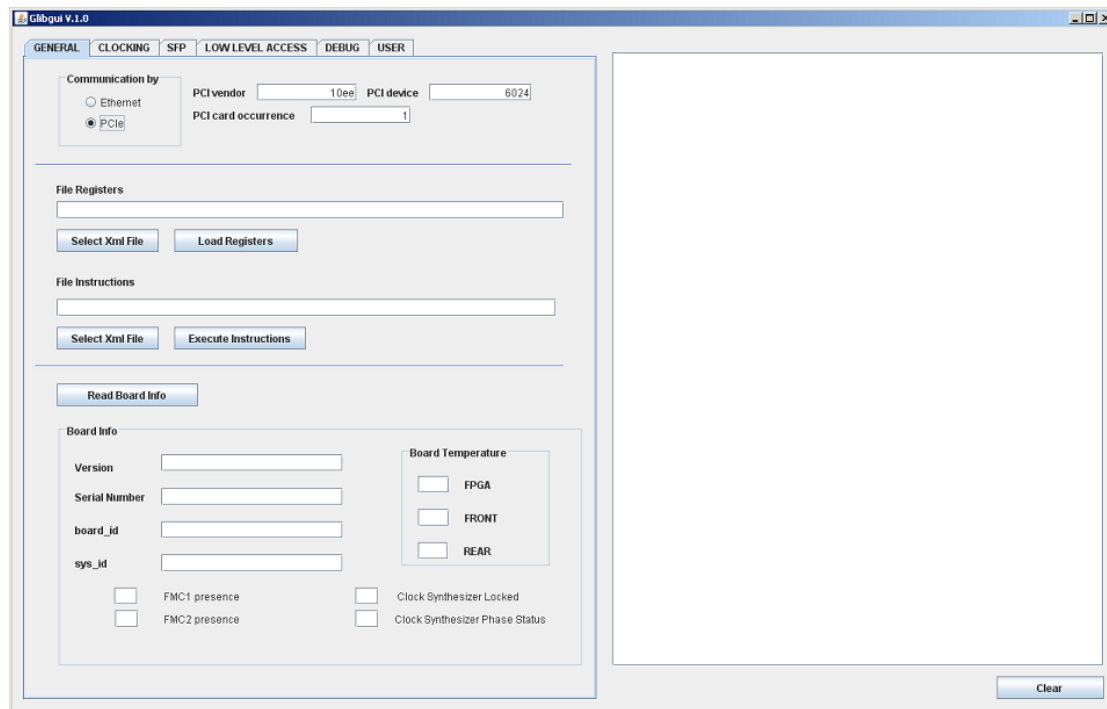


Figure 11 - The initial workspace

The Gui itself has several functions. In the general tab, the user can specify the ip and port of the GLIB card needed to establish the communication. The board information is also displayed, this covers:

- Serial number
- Version
- Board Temperature
- Indicates the presence of FMCs
- Board ID and Sys ID

Under the low level access tab, the user can exchange data with the GLIB card in an interactive way.

## 5.1 Cactus and IPbus

Like before with the first web interface, we are accessing the crates. Yet this time it is not done via the MCH and the System Manager program, instead individual cards in the crate are accessed. This is done using the IPbus protocol and Cactus (Code Archive for CERN Trigger UpgradeS), which is the C++ library for ipbus. IPbus is a protocol designed to establish a management connection between hardware and server computers via IP over Ethernet. It uses UDP as the transport protocol. It describes the basic transactions: read, write, non-incrementing read/write and atomic masked write. Each transaction request/response is self contained. Transactions can be concatenated (joined together) together into the same packet. Thus we can queue requests and dispatch them when needed. This improves network transport efficiency. Yet issues

were encountered in a sense that IPbus could not collect all the necessary information (specifically the temperatures) so we had to use system manager again for those values.

## 5.2 Web Toolkit framework

Similar to before, the data had to be displayed neatly and efficiently in a web interface. Rather than using a WYSIWYG approach it was decided to use Web Toolkit framework. Wt framework is a C++ library that is used for developing web applications. The application programming interface (API) is widget centric. It offers simple means of creating user interfaces for a wide variety of projects. The benefit of using this method to design the web interface is that it is entirely coded in C++ by implementing the widgets. As such, all the HTML, CSS and scripts that are needed are handled by the C++ code, so a detailed knowledge of web design is not required. Unlike before where a web server was required to host the webpage, using Wt, everything is created locally on the user's computer so it is much more simplistic.

## 5.3 The Gui

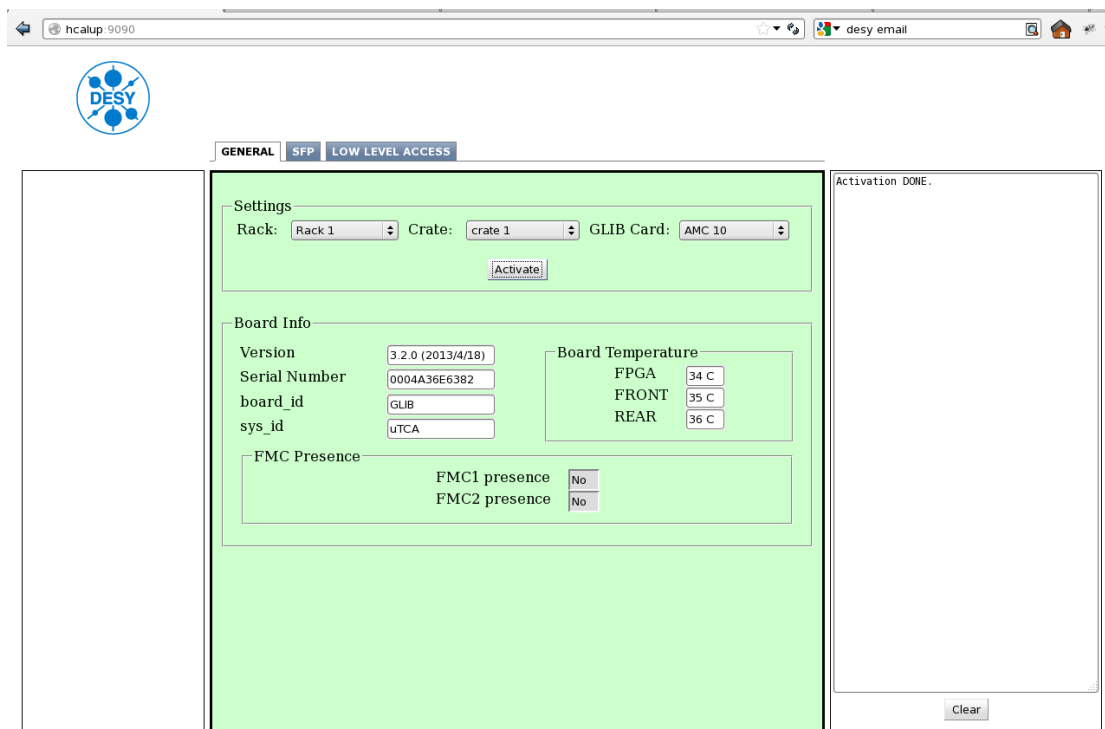


Figure 12 - The GLIB Gui

hcalup:9090 desy email

DES

GENERAL SFP LOW LEVEL ACCESS

On Board

	PRESENCE	RXLOS	TXFAULT
SFP1	No		
SFP2	Yes	Yes	No
SFP3	No		
SFP4	No		

Activation DONE.

Clear

Figure 13 - The SFP tab of the Gui

## 6. Acknowledgements

We would like to sincerely thank our supervisors Ulf Behrens and Alan Campbell for their time and effort during the summer program. We have learned a great deal and are thankful for them showing us an aspect of the LHC, we would have never known about otherwise. It is also a pleasure for us to thank DESY and the CMS  $\mu$ TCA group who made this project possible.

I would also like to thank my co-worker, André Ståhl for the work he has put in to help make the interfaces and his invaluable input.

Stuart

I would like to show my gratitude to my co-worker Stuart Szwec for his support and excellent work during the summer student program.

André

## 7. References

1. The HCAL Technical Design Report (CERN/LHCC 97-31, CMS TDR 2, 20 June 1997)
2. <http://cms.desy.de/>
3. CMS MicroTCA crate concepts & AMC card requirements. Gregory Iles, Magnus Hansen and Eric Hazen
4. <http://www.nateurope.com/products/NATview.html>
5. Micro Telecommunications Computing Architecture Short Form Specification, September 21st, 2006. Stuart Jamieson.
6. <http://www.kompozer.net/index.php>
7. <https://espace.cern.ch/project-GBLIB/public/default.aspx>