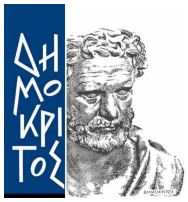
NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF APPLIED MATHEMATICAL & PHYSICAL SCIENCES
SCHOOL OF MECHANICAL ENGINEERING

NATIONAL CENTRE OF RESEARCH "DEMOKRITOS"
INSTITUTE OF NANOSCIENCE AND NANOTECHNOLOGY
INSTITUTE OF NUCLEAR AND PARTICLE PHYSICS



Postgraduate Program
Physics and Technological Applications

Department of Physics
Laboratory of Experimental High Energy Physics
& Related Technology-Instrumentation

Detector Control Systems for the Phase I upgrade of New Small Wheel in ATLAS Experiment

Master Thesis
of

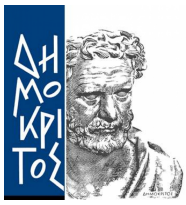
Stamatis Tzanos

Advisor:
Theodoros Alexopoulos
Professor, N.T.U.A

Athens, October 27, 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ
ΜΑΘΗΜΑΤΙΚΩΝ &
ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ

ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ
ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
"ΔΗΜΟΚΡΙΤΟΣ"
ΙΝΣΤΙΤΟΥΤΟ
ΝΑΝΟΕΠΙΣΤΗΜΗΣ &
ΝΑΝΟΤΕΧΝΟΛΟΓΙΑΣ
ΙΝΣΤΙΤΟΥΤΟ ΠΥΡΗΝΙΚΗΣ &
ΣΩΜΑΤΙΔΙΑΚΗΣ ΦΥΣΙΚΗΣ



Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
Φυσική και Τεχνολογικές Εφαρμογές

Τομέας Φυσικής
Εργαστήριο Πειραματικής Φυσικής Υψηλών Ενεργειών
& Συναφούς Οργανολογίας-Τεχνολογίας

Συστήματα ελέγχου ανιχνευτικών διατάξεων για την πρώτη φάση αναβάθμισης του New Small Wheel στο πείραμα ATLAS

Μεταπτυχιακή Εργασία
του
Σταμάτη Τζάνου

Επιβλέπων:
Θεόδωρος Αλεξόπουλος
Καθηγητής, Ε.Μ.Π.

Αθήνα, 27 Οκτωβρίου, 2022

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF APPLIED MATHEMATICS AND PHYSICAL SCIENCES
DEPARTMENT OF PHYSICS
LABORATORY OF EXPERIMENTAL HIGH ENERGY PHYSICS
& RELATED INSTRUMENTATION TECHNOLOGY

Detector Control Systems for the Phase I upgrade of New Small Wheel in ATLAS Experiment

Master Thesis

of

Stamatis Tzanos

Supervisor: Theodoros Alexopoulos
Professor, N.T.U.A.

Approved by the examination committee in October 2022.

.....	.....	.....
T. Alexopoulos	S. Maltezos	T. Geralis
Professor	Professor	Research Director
N.T.U.A.	N.T.U.A.	N.C.S.R. Demokritos

Athens, October 2022.

Εθνικο Μετσοβιο Πολυτεχνειο
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΠΕΙΡΑΜΑΤΙΚΗΣ ΦΥΣΙΚΗΣ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ
& ΣΧΕΤΙΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΟΡΓΑΝΩΝ

Συστήματα ελέγχου ανιχνευτικών διατάξεων για την πρώτη φάση αναβάθμισης του New Small Wheel στο πείραμα ATLAS

Μεταπτυχιακή Διπλωματική Εργασία

του

Σταμάτη Τζάνου

Επιβλέπων: Θεόδωρος Αλεξόπουλος
Καθηγητής, Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Οκτώβριο του 2022.

.....	.....	.....
Θ. Αλεξόπουλος	Σ. Μαλτέζος	Θ. Γέραλης
Καθηγητής	Καθηγητής	Διευθυντής Ερευνών
Ε.Μ.Π.	Ε.Μ.Π.	Ε.Κ.Ε.Φ.Ε. Δημόκριτος

Αθήνα, Οκτώβριος 2022.

.....
Stamatis Tzanos
Applied Physicist

.....
Σταμάτης Τζάνος
Φυσικός Εφαρμογών

Abstract

In conjunction with the High Luminosity upgrade of the Large Hadron Collider, the ATLAS detector is also undergoing an upgrade to handle the significantly higher data rates. The muon end-cap system upgrade in ATLAS, lies with the replacement of the Small Wheel. The New Small Wheel (NSW) is expected to combine high tracking precision with upgraded information for the Level-1 trigger. To accomplish this, small Thin Gap Chamber (sTGC) and MicroMesh Gaseous Structure (Micromegas) detector technologies are being deployed.

In order to successfully integrate, commission, monitor and control both detector technologies on NSW, various Detector Control System (DCS) applications need to be developed. The standard industrial control software used at CERN is WinCC OA. Through WinCC OA, a Supervisory Control and Data Acquisition framework has been established to maximize synergy between all experiments. ATLAS is hierarchically organized in a tree-like structure of detectors, followed by sub-detectors, sub-systems for each detector, etc. Thus, the ATLAS DCS must conform to this structure. Moreover, it is tasked to be a coherent and safe for operation interface for all sub-detectors of the experiment. It must also be able to bring the detector into any desired operation state, allow automatic or manual actions, report abnormal system behavior to operators, monitor and archive all parameters necessary for the experiment.

Due to the installation location of NSW in ATLAS, the effects of Barrel Toroid and End-Cap Toroid magnets on the sub-detectors must be measured. For the final experiment at ATLAS, each NSW large sector will be equipped with Hall sensors to monitor the magnetic field near the NSW. This readout is accomplished with an Embedded Local Monitor Board called MDT DCS Module, all of which is monitored by the corresponding control system.

Since the Micromegas is a gaseous type particle detector, all parameters of the gas system have to be monitored. The ATLAS Gas Group has implemented the monitoring according to the design of the gas system and thus, the DCS has to also confronm to the same scheme. Data from each sensor both in the gas mixer and distribution modules is published and monitored by the DCS. At the same time, alarms for all important parameters will bring operational statuses and states to every component of the gas system.

This thesis will describe both the commissioning and development of the two above mentioned control systems.

Περίληψη

Σε συνδυασμό με την αναβάθμιση Υψηλής Φωτεινότητας του Μεγάλου Αδρονικού Επιταχυντή, το πείραμα ATLAS θα πρέπει επίσης να αναβαθμιστεί για να μπορεί να διαχειριστεί την σημαντικά υψηλότερη ροή δεδομένων. Η αναβάθμιση του μιονικού φασματόμετρου στο ATLAS, έγκειται στην αντικατάσταση του Small Wheel. Το New Small Wheel (NSW) αναμένεται να συνδυάζει υψηλή ακρίβεια στην ανακατασκευή τροχιάς με αναβαθμισμένη πληροφορία για τον πρωτογενή σκανδαλισμό. Για την επίτευξη αυτού του αποτελέσματος, έχουν αναπτυχθεί δύο τύποι ανιχνευτικών διατάξεων: οι small Thin Gap Chamber (sTGC) και οι ανιχνευτές MicroMesh Gaseous Structure (Micromegas).

Για την επιτυχή ενοποίηση, θέση σε λειτουργία, παρακολούθηση και έλεγχο και των δύο ανιχνευτικών διατάξεων στο NSW, πρέπει να αναπτυχθούν εφαρμογές Συστημάτων Ελέγχου Ανιχνευτών (DCS). Το κύριο λογισμικό βιομηχανικού ελέγχου που χρησιμοποιείται στο CERN είναι το WinCC OA. Μέσω του WinCC OA, έχει δημιουργηθεί ένα πλαίσιο SCADA για τη μεγιστοποίηση της συνέργειας μεταξύ όλων των πειραμάτων. Το ATLAS είναι ιεραρχικά οργανωμένο σε δενδροειδή δομή ανιχνευτών, αποτελούμενη από υπο-ανιχνευτές, υποσυστήματα για κάθε ανιχνευτή κ.λπ. Επομένως, το σύστημα ελέγχου τουATLAS πρέπει να συμμορφώνεται με αυτή τη δομή. Επιπλέον, έχει ως αποστολή να είναι μια συνεκτική και ασφαλής διεπαφή λειτουργίας για όλους τους υπο-ανιχνευτές του πειράματος. Πρέπει επίσης να μπορεί να φέρει τον ανιχνευτή σε οποιαδήποτε επιθυμητή κατάσταση λειτουργίας, να επιτρέπει αυτόματες ή μη ενέργειες, να αναφέρει μη φυσιολογική συμπεριφορά του συστήματος στους χειριστές, να παρακολουθεί και να αποθηκεύει όλες τις απαραίτητες παραμέτρους για το πείραμα.

Λόγω της θέσης του NSW στο ATLAS, πρέπει να μπορούν να μετρηθούν οι τιμές των τοροϊδών μαγνητών Barrel Toroid και End-Cap Toroid στους υπο-ανιχνευτές. Για το τελικό πείραμα, κάθε μεγάλος τομέας του NSW θα είναι εξοπλισμένος με αισθητήρες φαινομένου Hall για την μέτρηση του μαγνητικού πεδίου κοντά στο NSW. Η λήψη των δεδομένων από τους αισθητήρες πραγματοποιείται με μία ενσωματωμένη πλακέτα επιτήρησης που ονομάζεται MDT DCS Module. Η όλη αυτή διαδικασία παρακολουθείται και από το αντίστοιχο σύστημα ελέγχου.

Δεδομένου ότι ο ανιχνευτής Micromegas είναι ένας ανιχνευτής αερίου, όλες οι παράμετροι του συστήματος αερίου πρέπει να παρακολουθούνται. Το ATLAS Gas group έχει σχεδιάσει την λήψη των δεδομένων σύμφωνα με το σχεδιασμό του συστήματος αερίου και ως εκ τούτου, το σύστημα ελέγχου πρέπει επίσης να συμμορφωθεί με τον ίδιο τρόπο. Τα δεδομένα από κάθε αισθητήρα τόσο στον μίκτη αερίου όσο και στις μονάδες διανομής αερίου εκδίδονται και έπειτα παρακολουθούνται από το σύστημα ελέγχου. Ταυτόχρονα, τα διαφορετικά εύρη τιμών στις σημαντικές φυσικές παραμέτρους πρέπει να φέρνουν το σύστημα σε κατάλληλες καταστάσεις λειτουργίας.

Σε αυτή τη μεταπτυχιακή διατριβή θα περιγραφεί τόσο η ανάπτυξη όσο και η δοκιμή των δύο προαναφερθέντων συστημάτων ελέγχου.

Acknowledgements

I will start by saying that this thesis would not be possible if it was not for Theo Alexopoulos who trusted me and my abilities with this extraordinary opportunity to work for New Small Wheel. Theo has been a wise mentor and a hardworking colleague at the same time. I also want to thank him for giving me other control system tasks during my undergraduate thesis that luckily made me realize my career path and passion. My school and National Technical University of Athens will always be a part of my journey as a physicist and as a person.

In the lab, I had the chance to spend time with colleagues (that later became good friends) that inspired and guided me in similar areas: Anthony, Polyneikis and Christos. I would like to thank them as well for their contribution in shaping my character as a scientist, as a colleague and especially as a winter sports athlete.

Working at CERN for this thesis also gave me the opportunity to meet some people that I still consider important in my life: Anthony, George, Giannis, Olga, Foteini, Emanuele, Tirsi and many more. It was a privilege to work and hang out alongside them in this common setting of late night work shifts, lunch breaks or Switzerland and Super Market expeditions.

I finally want to thank my parents, Dimitris and Anna but together my childhood friends (who have been promoted to family as well) for staying at my side and helping me on the way: Kostas, Christos, Dimitris, Markos, Froso, Magda, Iria and Eryk, Kimon, Ester, Myrto, Lydia, Stavros, Giannis, Eva, Nikolina, Sia and Maria.

Contents

Chapter 1

CERN and the Large Hadron Collider

1.1 CERN as the European Organization for Nuclear research

CERN was born from the French abbreviation of *Conseil Européen pour la Recherche Nucleaire* which stands for European Organization for Nuclear Research. It is the world's leading laboratory for particle physics and hosts 11175 users from institutions in 77 countries [1].

The convention establishing CERN as an organization was ratified on 29 September 1954 by 12 countries in Western Europe: Belgium, Denmark, France, the Federal Republic of Germany, Greece, Italy, the Netherlands, Norway, Sweden, Switzerland, the United Kingdom and Yugoslavia [2]. During these early years, the council worked at the University of Copenhagen under the direction of Niels Bohr before moving to its present site in Geneva.

CERN's large scale experimental area is currently hosted in the Swiss-French border and can be seen in Figure 1.1. It hosts the biggest accelerator complex in the world, led by the Large Hadron Collider (LHC) and it's four particle detector experiments: ATLAS, LHCb, ALICE and CMS. CERN also utilizes hundreds of buildings that serve as offices, administration buildings, control and operation buildings, storages and experimental sites.

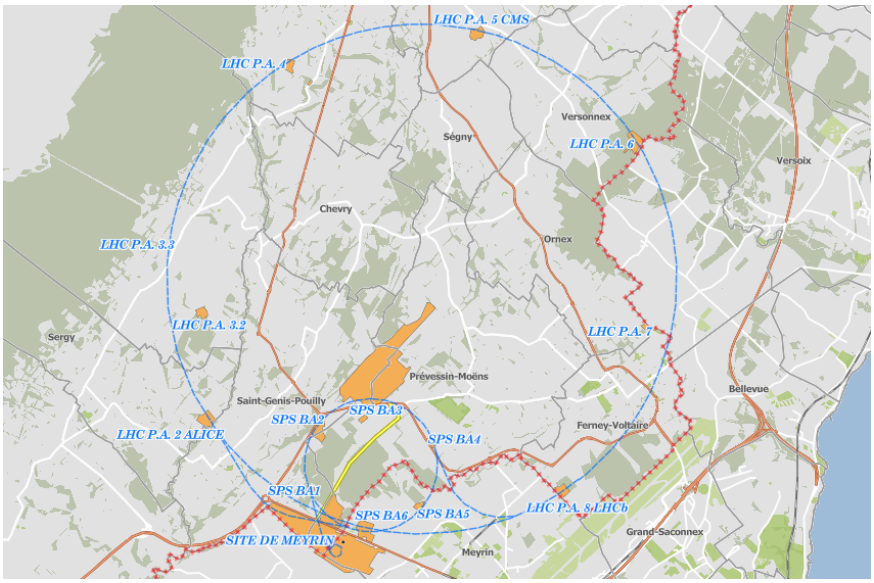


Figure 1.1: Most of CERN's accelerator complex (blue) and service buildings (orange).

Many novel technologies emerge from the knowledge and expertise developed at CERN for particle physics. By identifying key technologies with potential societal impact, CERN seeks out opportunities to work with industry on a wide range of applications. The most famous example is the World Wide Web in 1989. Tim Berners-Lee, a British scientist, invented the World Wide Web (WWW) while working at CERN. The web was originally conceived and developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world [3].

CERN is currently held responsible for several important achievements in particle physics, some of the most notable being:

- 1973: The discovery of neutral currents in the Gargamelle bubble chamber [4] [5].
- 1983: The discovery of W and Z bosons in the UA1 and UA2 experiments [5].
- 1989: The determination of the number of light neutrino families at the Large Electron–Positron Collider (LEP) operating on the Z boson peak [6].
- 1995: The first creation of antihydrogen atoms in the PS210 experiment [7] [8].
- 1999: The discovery of direct CP violation in the NA48 experiment [9].
- 2000: The Heavy Ion Programme discovered new state of matter, the Quark Gluon Plasma [10].
- 2010: The isolation of 38 atoms of antihydrogen [11].
- 2011: Maintaining antihydrogen for over 15 minutes [12].
- 2012: A boson with mass around 125 GeV c^2 consistent with the long-sought Higgs boson [13] [14].

1.2 The Large Hadron Collider at CERN

CERN operates a large network of seven accelerators (with some additional small accelerators) and two decelerators. Each machine in the chain increases the energy of particle beams before delivering them to experiments or to the next more powerful accelerator. Before injection into the LHC, the beams are produced and gradually accelerated up to 450 GeV by this series of smaller accelerators. This is known as the LHC injector complex and can be seen in Figure 1.2 [15].

The Large Hadron Collider (LHC) is the world's largest and most powerful particle accelerator. It first started up on 10 September 2008 and remains the latest addition to CERN's accelerator complex. The LHC consists of a 27-kilometre ring of superconducting magnets [16] (to bend) and a number of radiofrequency cavities [17] (to accelerate) the beam of particles along the way. Due the costs of acquiring such large areas of land, it was much cheaper to excavate a tunnel and host the experiment underground. The tunnel was built at a mean depth of 100 m where the Earth's crust provides good shielding for the radiation.

The LHC accelerates two particle beams in opposite directions which collide in each of the four detection experiments. A linear accelerator produces much less energy to form new particles. Specifically, the energy available in both cases is the center-of-mass energy. For a collider, the energy produced is the sum of the energies of the two colliding particles

$$E = E_{beam1} + E_{beam2}$$

whereas in a linear accelerator, it proportional to the square root of the energy of the particle hitting the target and thus

$$E \approx \sqrt{E_{beam}}$$

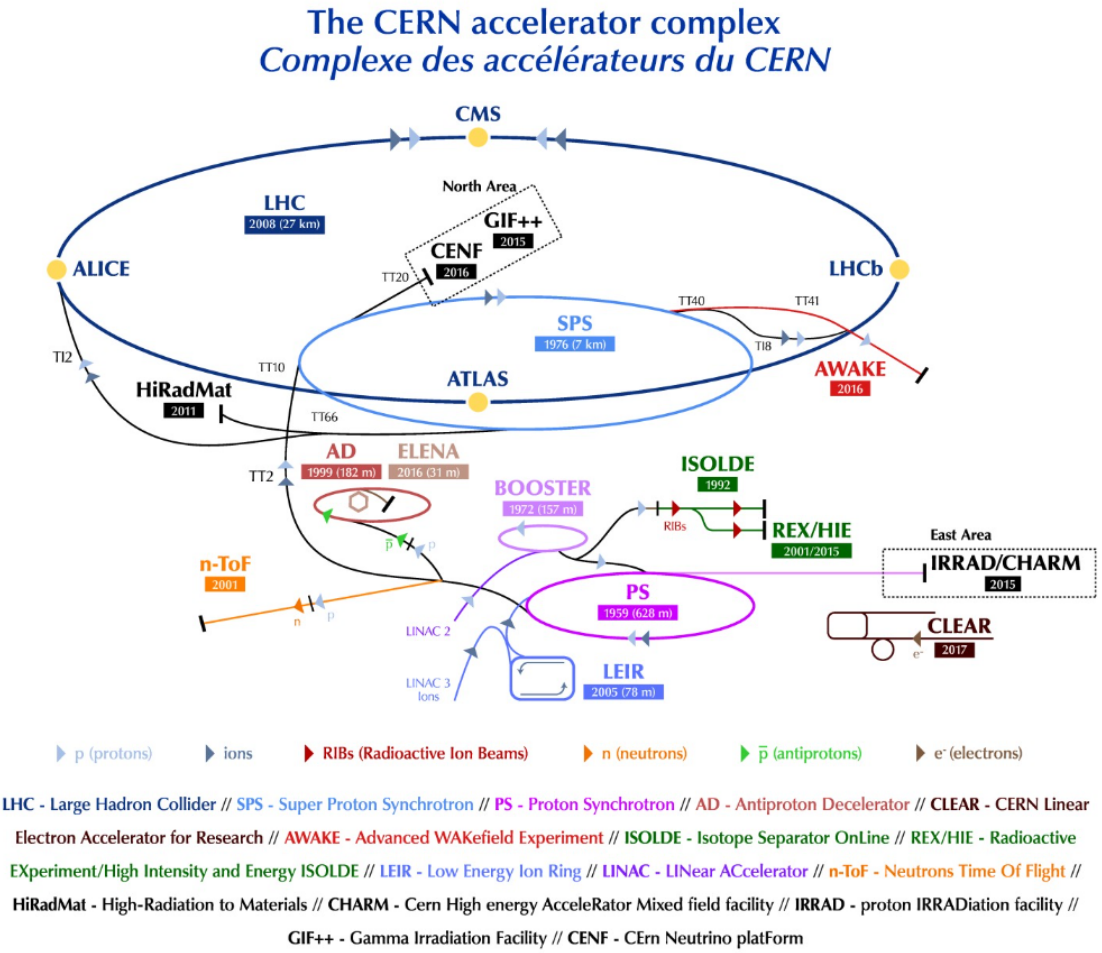


Figure 1.2: The CERN accelarator complex.

The LHC specifically uses two beams of particles of the same kind, either protons or lead ions which are hadrons. These are very suitable candidates as an accelerator has to accelerate particles that are charged and stable. This way, the electromagnetic machinery used in the tunnel can accelerate and turn the beam as desired.

LHC was successfully commissioned in 2010 for proton–proton collisions with a 7 TeV centre-of-mass energy. It delivered 8 TeV centre-of-mass proton collisions from April 2012 until the end of Run 1 in 2013. Following a long technical stop in 2013-2014, it operated with 13 TeV centre-of-mass proton collisions during Run 2 from 2015 onwards [18]. A yearly timeline of LHC and it's upgrades is shown in Figure 1.3 which also displays the upgrade of LHC into HL-LHC [19].

After Run 3 the statistical gain in running the accelerator without a significant luminosity increase beyond its design and ultimate values will become marginal. The running time necessary to halve the statistical error of a given measurement after 2020 will be more than ten years. Therefore, to maintain scientific progress and to exploit its full capacity, the LHC will need to have a decisive increase of its luminosity after 2020.

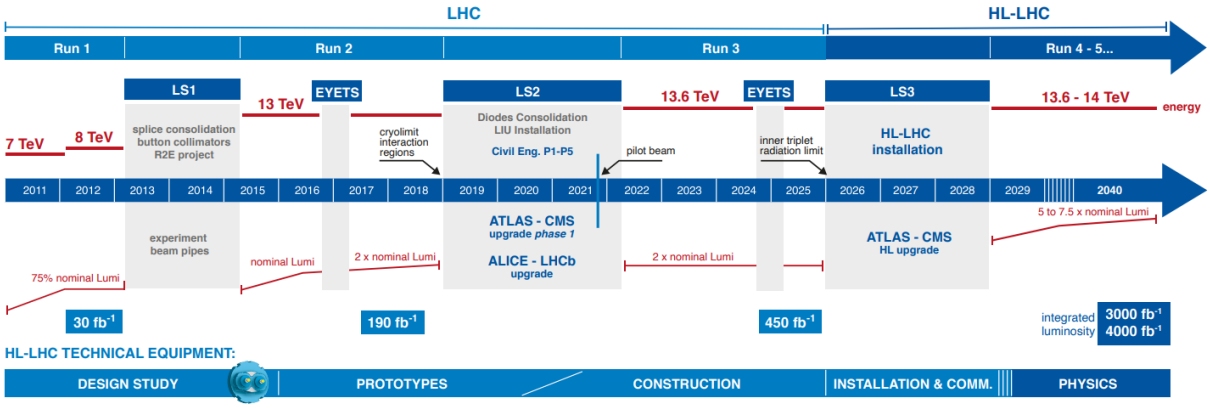


Figure 1.3: The LHC and HL-LHC tiimeline.

1.3 In search for new Physics

CERN's main mission is the study of particle physics - the fundamental constituents of matter. The focus lays mostly on the Standard Model of particle physics, which describes three of the four known fundamental forces in the universe and classifies all known elementary particles [20].

In most proton collisions, a large variety of particles are produced by the interactions of the quarks [21] and gluons [22] inside them. The interactions produce mostly low-energy ordinary elementary particles but occasionally, heavier particles are produced or other that are very energetic together with their antiparticles. Depending on the energy of these products, these new particles may decay to other or even collide with one another and produce even more particles. The higher the energy of the beams colliding, the more 'room' for the protons to create more interactions. Consequently, more collisions lead to better probability of an interaction to occur, better statistics of the interactions and thus better understanding of the results. A typical approach of high-energy physics research is to theoretically predict a particle's behavior and attempt to measure it through the interactions seen in the experiments. The Higgs boson is seen through some very delicate interactions that occur very rarely and thus, very high energy beam collisions were required to distinguish them from the sea of particles produced.

The measurement responsibility after the beam collision lies with the detector experiments that attempt to measure the critical parameters of the particles produced from the quadrilion collisions. The main goal is to reconstruct the track from the initial interaction and calculate which other particles took part in it and how. To achieve such results, particle detectors are constituted of a large area of layered technologies that are used to calculate all these different parameters. More on this matter will be explained in the next chapter.

After the discovery of the Higgs boson at $125\,\mathrm{GeV}/c^2$ and therefore the confirmation of the Higgs mechanism [23] [24] [25], physics is thrown in a new era of exploration in the subatomic universe and it's properties. To observe interactions that occur with much higher rarity or calculate results with much better precision the collider would need more collisions per second and with higher energy.

In scattering theory and accelerator physics, luminosity [26] is the ratio of the number of events detected dN in a certain period of time dt to the cross section σ [27]:

$$L = \frac{1}{\sigma}\frac{dN}{dt}$$

It has the dimensions of events per time per area [cm^2/s]. In practice, L is dependent on the particle beam parameters, such as beam width, density and particle flow rate. Taking these paremeters into consideration, the integrated luminosity can be calculated as a related quantity:

$$L_{int} = \int L dt$$

The luminosity and integrated luminosity are useful values to characterize the performance of a particle accelerator. In particular, all collider experiments aim to maximize their integrated luminosities, as it produces more data available for analysis.

The next upgrade of the LHC, High-Luminosity LHC, will attempt to answer very difficult questions in particle physics by providing even higher energies and more concentrated beams. At the same time, the enormous statistics of interactions produced will provide more precision for already calculated parameters in theory. Some of the most popular problems that still remain unsolved in particle physics are:

- Hierarchy problem: Why is gravity such a weak force? It becomes strong for particles only at the Planck scale [28] around 10^{19} GeV.
- Proton decay and spin crisis: Is the proton fundamentally stable? Or does it decay with a finite lifetime as predicted by some extensions to the standard model [29]? How do the quarks and gluons carry the spin of protons?
- Supersymmetry: Is spacetime supersymmetry [30] realized at TeV scale? If so, what is the mechanism of supersymmetry breaking? Does supersymmetry stabilize the electroweak scale [31], preventing high quantum corrections?
- Strong CP problem and axions: Why is the strong nuclear interaction invariant to parity [32] and charge conjugation [33]? Could axions [34] be the main component of dark matter [35]?
- Matter Antimatter Asymmetry: Why is everything almost entirely made up of matter and not antimatter? Why do the laws of physics not apply equally to matter and antimatter?[36]

These and many more questions that are posed for physics to solve over the decades are still mysteries which can be slowly uncovered with better understanding of the subatomic universe.

Chapter 2

Particle detection at CERN

As it has already been mentioned, particles are produced from LHC beam collisions at very high energies. Particles produced in collisions normally travel in straight lines but in the presence of a magnetic field their paths become curved. Electromagnets around particle detectors generate magnetic fields to exploit this effect. The momentum of a particle – a clue to its identity – can be calculated from the curvature of its path: particles with high momentum travel in almost straight lines, whereas those with very low momentum move forward in tight spirals inside the detector.

Modern particle detectors consist of layers of subdetectors, each designed to look for particular properties, or specific types of particles. Tracking devices reveal the path of a particle; calorimeters stop, absorb and measure a particle's energy; and particle-identification detectors use a range of techniques to pin down a particle's identity.

Tracking Devices Tracking devices reveal the paths of charged particles as they pass through and interact with suitable substances. Most tracking devices do not make particle tracks directly visible, but record tiny electrical signals that particles trigger as they move through the device. Software is then used to reconstruct the recorded patterns of tracks.

One type of particle, the muon [37], interacts very little with matter – it can travel through meters of dense material before being stopped. Muons therefore pass easily through the inner layers of a detector, which is why muon chambers – tracking devices specialised in detecting muons – usually make up the outermost layer of a detector.

Calorimeters A calorimeter measures the energy a particle loses as it passes through. It is usually designed to stop entirely or "absorb" most of the particles coming from the collisions, forcing them to deposit all of their energy within the detector and thus measuring their full energy. Calorimeters have to perform two different tasks at the same time: stopping particles and measuring energy loss. They usually consist of layers of different materials: a "passive" or "absorbing" high-density material (i.e. lead) interleaved with an 'active' medium such as plastic scintillators or liquid argon.

Electromagnetic calorimeters measure the energy of electrons and photons as they interact with the electrically charged particles in matter. Hadronic calorimeters sample the energy of hadrons (particles containing quarks, such as protons and neutrons) as they interact with atomic nuclei. Calorimeters can stop most known particles except muons and neutrinos.

The LHC hosts four main particle detection experiments: The ATLAS Experiment [38], the ALICE experiment [39], the LHCb experiment [40] and the CMS experiment [43]. The focus on this thesis will be on ATLAS experiment and it's phase I upgrade.

2.1 The ATLAS Detector

ATLAS detector is hosted in the CERN experimental cavern UX15. It is a 44 m long cylinder with a 25 m diameter and sits at 100 m below ground. ATLAS detector together with the subdetectors and their subsystems weights around the total of 7000 t [42].

The detector is constituted of six different detecting subsystems wrapped concentrically in layers around the collision point to record the trajectory, momentum, and energy of particles, allowing them to be individually identified and measured. A huge magnet system bends the paths of the charged particles so that their momenta can be measured as precisely as possible. ATLAS detector and it's subsystems can be seen in Figure 2.1 below.

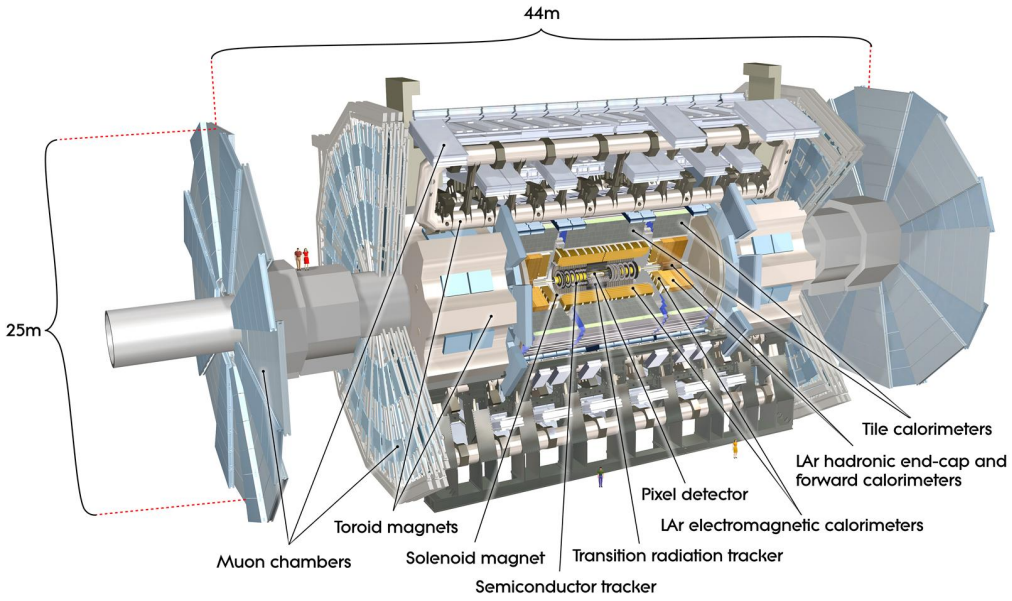


Figure 2.1: Most of ATLAS detector main components and subsystems. ATLAS is divided in the "Barrel" region which is identified as the cylindrical area and the "End-Cap" regions which are identified as the caps of the barrel geometry.

ATLAS is one of two general-purpose detectors at the LHC. It investigates a wide range of physics, from the search for the Higgs boson to extra dimensions and particles that could make up dark matter. Although it has the same scientific goals as the CMS experiment, it uses both different technical solutions and a different magnet-system design.

The interactions in the ATLAS detectors create an enormous flow of data. To digest the data, ATLAS uses an advanced "trigger" system [44] to filter out undesired events. The events recorded are processed by complex algorithms for reconstruction and analysis. The ATLAS trigger system carries out the selection process of events in two stages:

1. Level-1 trigger: a hardware trigger that selects the events to keep in less than 2.5 µs. Buffers keep the data before it is either discarded or selected and then pass the selected data to the second-level trigger.

2. Level-2 trigger: a software trigger that operates with around 40000 CPU cores. This trigger conducts detailed analyses of each collision event and passes the data to a storage system for offline analysis.

This operation is carried out by the Small Wheels (Level-1 trigger) and the Big Wheels (Level-2 triggers).

2.1.1 ATLAS Muon spectrometer

The muon spectrometer is designed to identify and measure the momentum of muons in three regions: the Barrel, extending in the pseudorapidity [45] region $|\eta| \leq 1.2$ and the two End-Caps covering the pseudorapidity regions $1 < |\eta| < 2.7$ [47]. These areas can be seen in Figure 2.2.

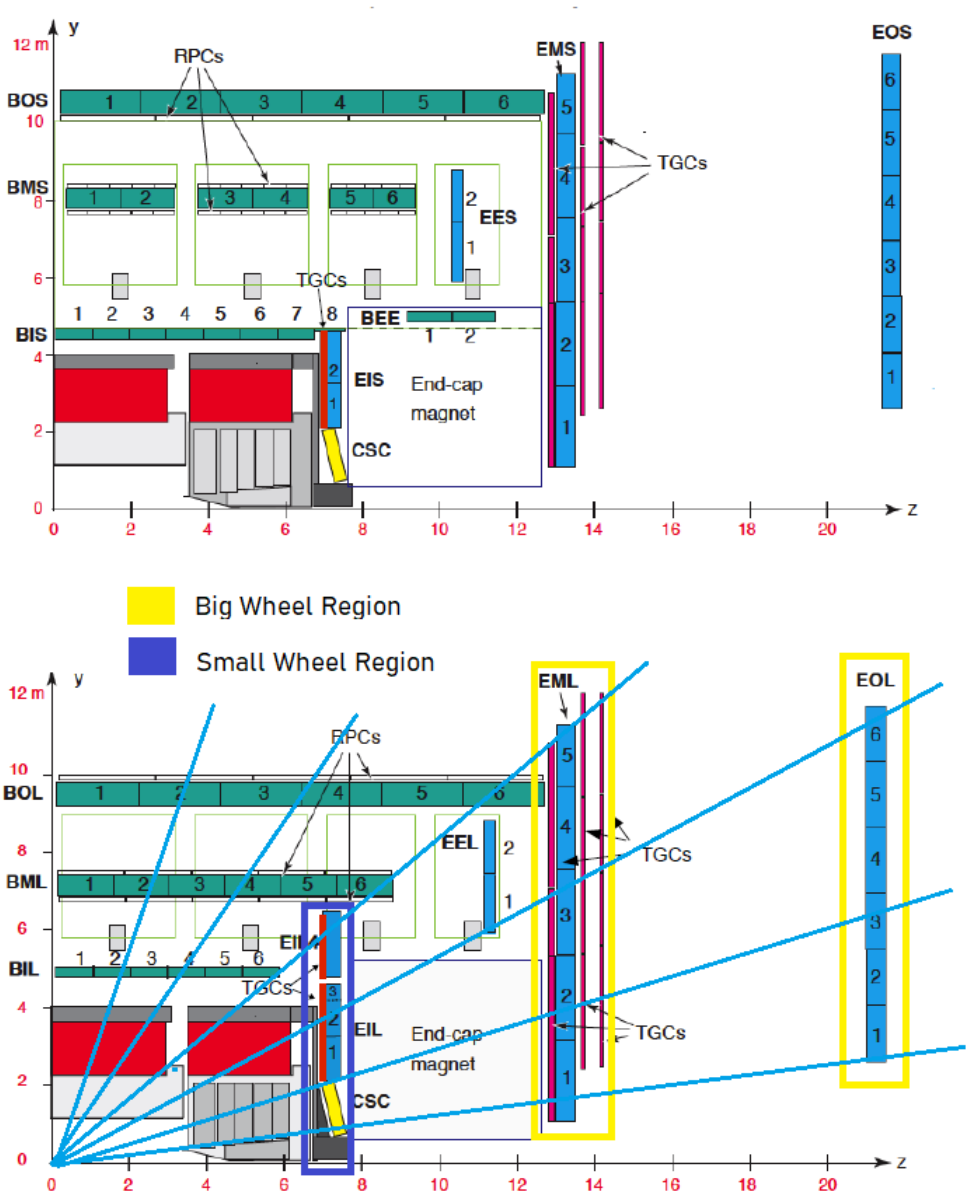


Figure 2.2: The abbreviations read: **B** for Barrel, **E** for End-Cap, **I, M, O** for Inner, Middle and Outer. The distinction between **S, L** stands for Small and Large.

The muon spectrometer, made up of 4,000 individual muon chambers uses four different technologies by 48 institutions in 23 production sites around the world [46]. The muon spectrometer upgrade will also feature a new, fifth technology. These subdetectors are listed below:

1. Thin Gap Chambers (TGC): Used for triggering and 2nd coordinate measurement (non-bending direction) at ends of detector. They utilize 440000 channels for readout. An upgraded version of the TGC is the small TGC (sTGC) chamber aimed at the New Small Wheel. [48]

2. Resistive Plate Chambers (RPC): Used for triggering and 2nd coordinate measurement in the central region. They utilize 380000 channels and their electric field reaches 5000 V/mm. [49]

3. Monitored Drift Tubes (MDT): Used to measure the curves of tracks. ATLAS Experiment consists of 1171 chambers with a total of 354240 tubes with tube resolution of 80 µm. [50]

4. Cathode Strip Chambers (CSC): Used to measure precision coordinates at ends of detector. They utilize 70000 channels with a resolution of 60 µm. (Deprecated technology after the upgrade) [51]

5. Micromesh Gaseous Structure (Micromegas): Used for triggering and 2nd coordinate measurement in the central region. Micromegas will replace the MDT chambers in the Inner End-Cap region. They utilize 524288 channels for readout. [52]

2.1.2 The ATLAS magnet system

By bending the trajectories of charged particles, ATLAS can measure both momentum and charge. The ATLAS magnet system is composed of three main sections: the Central Solenoid, the Barrel Toroid and End-Cap Toroids. The magnet systems are cooled to about 4.5 K (−268 °C) to bring the magnetic fields to their strongest [53].

Central Solenoid Magnet

The ATLAS solenoid surrounds the inner detector at the core of the experiment. This powerful magnet is 5.6 m long, 2.56 m in diameter and weighs over 5 t. It provides a 2 T magnetic field in just 4.5 cm thickness. This is achieved by embedding over 9 km of Niobium-Titanium superconductor wires [54] into strengthened, pure aluminum strips, thus minimising possible interactions between the magnet and the particles being studied. The nominal current of the magnet is at 7.73 kA. The ATLAS Central Solenoid magnet can be seen in Figure 2.3.

Figure 2.3: ATLAS Central Solenoid magnet during installation.

Toroid Magnets

The ATLAS toroids use a series of eight coils to provide a magnetic field of up to 3.5 T, used to measure the momentum of muons. There are three toroid magnets in ATLAS: two at the end-caps of the experiment and one massive toroid surrounding the centre of the experiment.

At 25 m in length, the central toroid is the largest toroidal magnet ever constructed. It uses over 56 km of superconducting wire [54] and weighs about 830 t. The End-Cap toroids extend the magnetic field to particles leaving the detector close to the beam pipe. Each End-Cap is 10.7 m in diameter, 5 m in axial length and weighs 240 t. To summarize the information, the Barrel Toroid magnet utilizes

- 8 separate coils
- 1.08 GJ stored energy
- 4 T magnetic field on superconductor
- 56 km Al/NbTi/Cu conductor cable
- nominal current at 20.5 kA
- 4.7 K working point temperature

Figure 2.4: ATLAS Barrel toroid magnet during the installation of one torus.

The End-Cap Toroid magnets, each utilizes

- 8 coils in a common cryostat [55]
- 0.25 GJ stored energy
- 4 T magnetic field on superconductor

- 13 km Al/NbTi/Cu conductor cables
- nominal current at 20.5 kA
- 4.7 K working point temperature

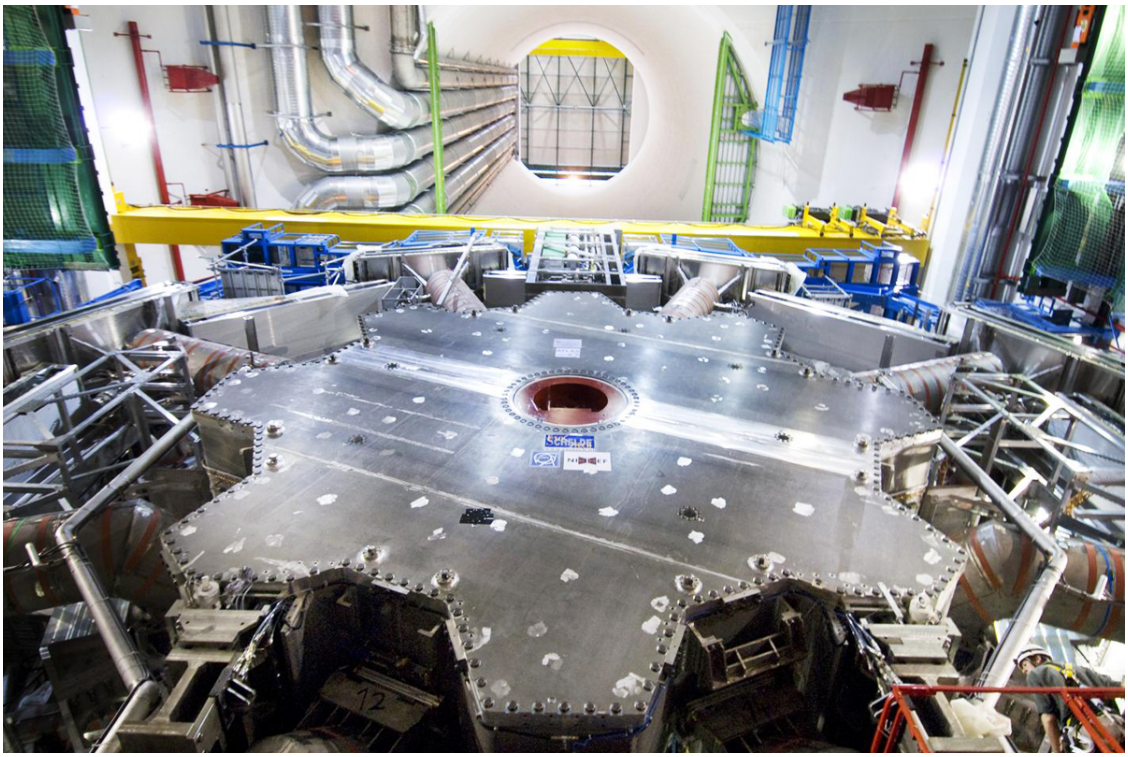


Figure 2.5: ATLAS End-Cap toroid magnet bottom-up view.

2.2 ATLAS upgrade: New Small Wheel

In order to fully exploit the HL-LHC physics potential, the detector experiments require major upgrades to cope with the new conditions. The ATLAS experiment was designed for a broad physics programme, including the capability of discovering the Higgs boson. However, two major issues represent a serious limitation on the ATLAS performance beyond the original design lumonisity [18] [56] [57]:

- The performance of the muon tracking chambers (in particular in the end-cap region) degrades with the expected increase of cavern background rate.
- Unacceptable rate of fake high p_T Level-1 muon triggers (approximately 90%) coming from the interaction point (IP) due to low energy particles (mainly protons) that are produced by the material around the end-cap region in similar angles.

In order to solve the two problems together, ATLAS proposes to replace the present muon Small Wheels with the "New Small Wheels" (NSW). The NSW will feature a set of precision tracking (Micromegas) and trigger (sTGC) detectors able to work at high rates with excellent real-time spatial and time resolution. These detectors can provide the muon Level-1 trigger system with online track segments of good angular resolution to confirm that muon tracks originate from the IP. In this way the end-cap fake triggers will be considerably reduced. NSW can be seen in Figure 2.6.

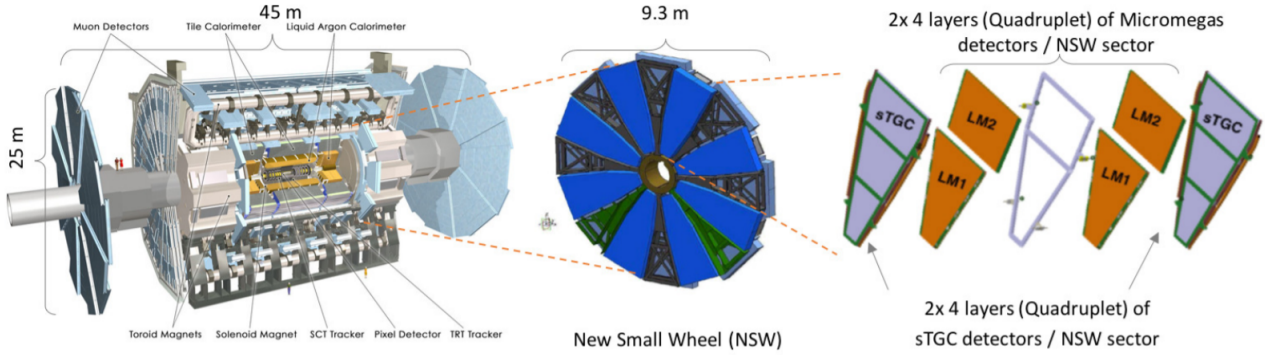


Figure 2.6: The NSW position in ATLAS and it's components.

2.2.1 Detector layout

Figure 2.7 shows the Small Wheels that were replaced in the End-Cap region of ATLAS detector. It was composed of two distinct parts:

- A detector wheel with the appropriate mechanical structure to hold the previously used precision tracking detectors (MDT and CSC) together with its alignment system. The mechanical structure consists of an inner massive hub to which radially extending, interconnected spokes are attached.
- The JD shielding, a disk shaped shielding covering the full extend of the detector wheel with a cylindrical extension (plug) around the beam pipe at the inner radius [58]. The JD shielding has two feet to support it inside the ATLAS detector. Triggering detectors are mounted on the JD disk.

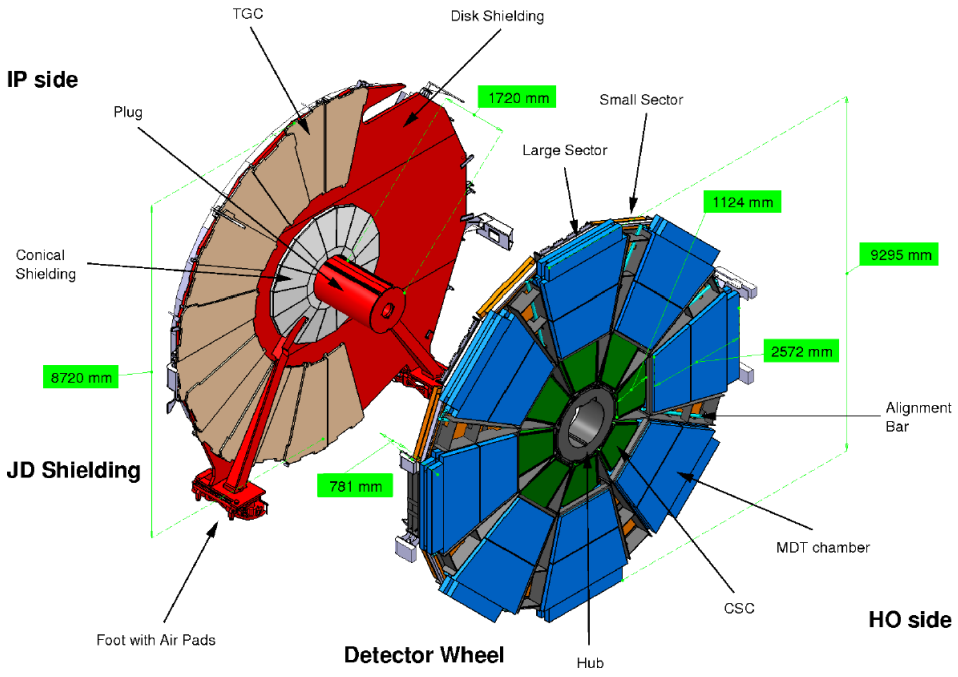


Figure 2.7: Components and layout of the Small Wheel that was replaced.

The main part of the forward muon shielding, referred to as the JD shielding, is located between the end-cap calorimeter and the end-cap muon chambers of the Small Wheel. It will

be re-used for the NSW, but slightly modified to accommodate the new planar geometry of the detector wheel.

The present JD shielding consists of three parts, the central cylindrical plug made of brass with an outer stainless steel jacket surrounding the beam pipe, the disk shielding of carbon steel which covers the SW muon detectors, and the conical shielding, made of brass and clad by borated polyethylene and a lead layer, at the inner radius of the disk shielding. The modification of the JD shielding only affects the conical part, which will be replaced by a flat cylinder, again made of high-Z metal material and clad by borated polyethylene covered by a lead layer. The new JD shielding can be seen in Figure 2.8.

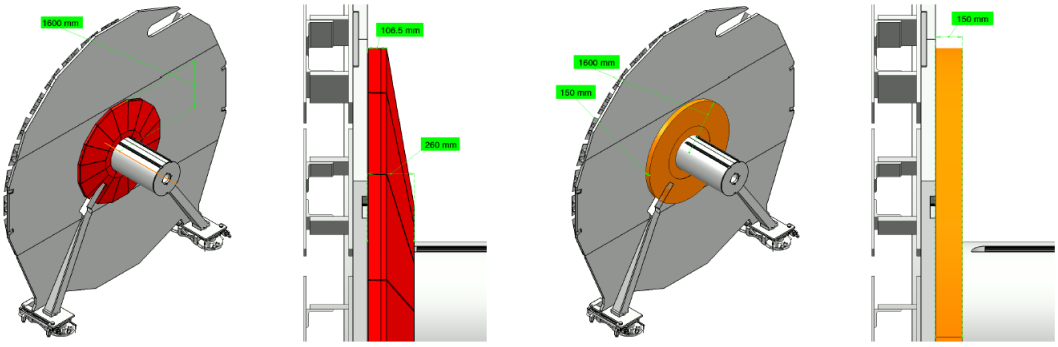


Figure 2.8: Side view of the JD shielding disc designs. The right side shows the modification of the conical design into a disk-like geometry to provide space for the full detector coverage.

The detector mechanical support structure consists of an inner cylindrical hub with radial spokes which hold the detectors in position. The hub is also made of brass and provides additional shielding from the beam pipe. The mechanical structure also hosts 16 alignment bars with optical sensors to monitor the detector positions by forming a grid [59]. The structure holds in segments, 8 small sectors of detectors on the inner side (IP side - facing the interaction point) and 8 large sectors of detectors on the outer side (HO side - facing away from the interaction point).

Each of the 16 sectors of the NSW structure hosts a layered slice of the detector with the Micromegas and sTGC technologies. This is shown in Figure 2.9.

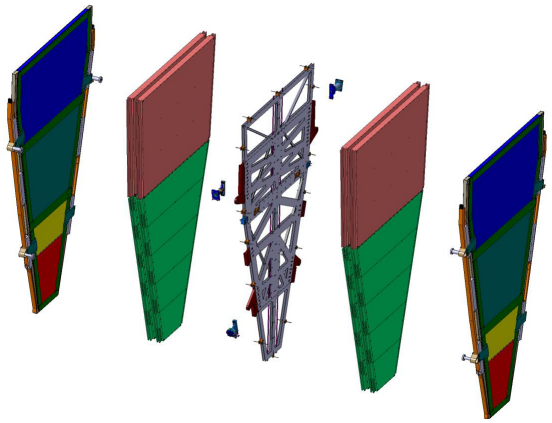
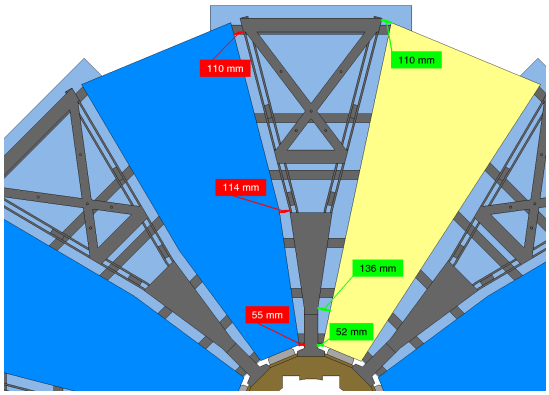
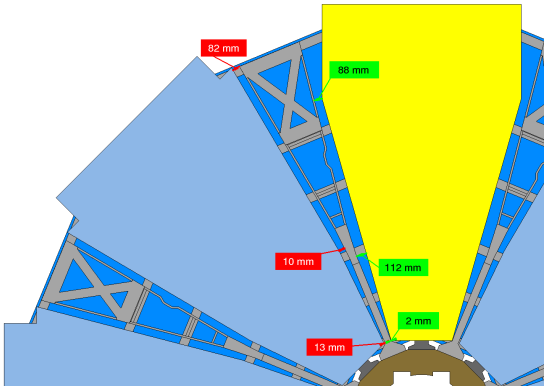


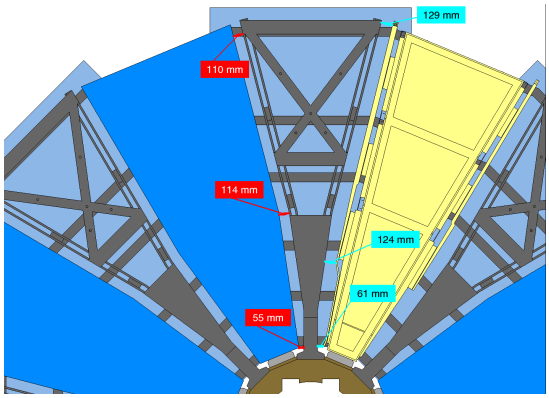
Figure 2.9: An NSW sector with two sTGC detectors on the outside and two Micromegas detectors on the inside. In the middle, a support structure holds the two technologies bound as one.



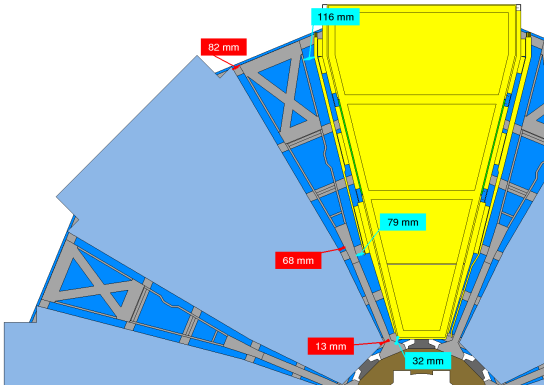
(a) Micromegas (small sector).



(b) Micromegas (large sector).



(c) sTGC (small sector).



(d) sTGC (large sector).

Figure 2.10: Layout of the Micromegas and sTGC detectors on the New Small Wheel support structure.

The assembled sector is installed on the support structure using special cranes and the alignment bars. They are positioned as shown in Figure 2.10.

2.2.2 The Micromegas detector

The Micromegas detector is an acronym of MicroMesh Gaseous Structure detector. It is a gaseous detector that offers for several fields of application substantial advantages in energy, space and time resolution. It is a detector that offers microscopic granularity on large surfaces and important insensitivity to discharges. Micromegas will server as the leading technology for tracking purposes on the New Small Wheel.

Principle of operation

Micromegas is composed of two parallel plates that feature a narrow amplification space (typically 50−60 µm) between two parallel electrodes, the cathode and anode conducting plates. The cathode is made of a thin metallic micromesh and the anode is made of microelements of a conductor printed on an insulator board. The challenge of building such detector lies with keeping the small gap constant over the active detector area. 1 % of the detector surface is covered by small insulating pillars that hold the anode and cathode distance [60]. Micromegas chambers will use the gas mixture of Argon ($Ar - 93\%$), Carbon Dioxide ($CO_2 - 5\%$) and isobutane ($iC_4H_{10} - 2\%$) for operation [61].

For the high energy physics experiments, a third electrode is placed parallel to the mesh that defines a larger gas-filled regions where electrons, released by any conversion process are drifted towards the mesh. The electrons then travel inside the amplification region where they are multiplied in an avalanche like process that results in a large number of electron-ion pairs in opposite directions in the detector. Figure 2.11 shows this design:

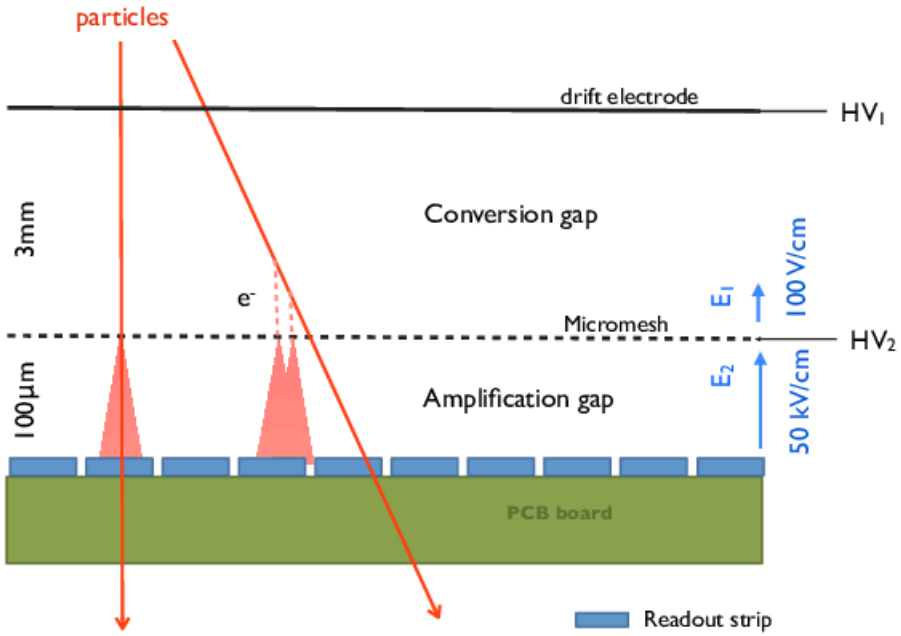


Figure 2.11: The above image shows the areas of the detector. The micromesh splits the detector between the drift region and amplification region.

For simplicity, let's suppose a grounded PCB board and that voltage is applied in the two plates: V_{mesh} and V_{drift}. In the detector, a capacitor like structure creates electric fields in the drift and amplification regions. If d_c and d_a are the conversion and amplification gaps:

$$E_1 = \frac{V_{drift}}{d_c}, \quad E_2 = \frac{V_{mesh}}{d_a}$$

The charged particle ionizes the gas molecules producing electrons that travel, although accelerated by the electric field, with a constant speed because of their collisions with the gas molecules [62]

$$u_{drift} = \mu \frac{E_1}{P}$$

where P is the pressure of the gas and μ the mobility of the electrons. The electrons pass into the amplification region meeting the significantly larger E_2 which creates a Townsend discharge [63]. An avalanche of electrons creates an amplification process which showers the resistive strips with a charge. The auxiliary current on a readout strip [64] is therefore

$$I_{p,mesh}(t) \approx -\frac{eu_p}{d_a}\left[e^{\alpha d_a} - e^{\alpha\beta u_p t}\right] \tag{2.1}$$

where e is the electron charge, u_p is the drift velocity of the positive ions towards the mesh, $\alpha = u_p/(u_n + u_p)$, $\beta = u_n/(u_n + u_p)$ and u_n is the drift velocity of the electron towards the readout strip. Time $0 \leq t \leq d_a/u_n$ is measured from the time the Townsend discharge is produced.

Data acquisition

The architecture of the Micromegas data acquisition chain is composed of three front-end electronic boards: The Level-1 Data Driver Card (L1DDC) [65], the ART [66] (Address in Real Time) Data Driver Card (ADDC) and the MicroMegas Front End with 8 VMM [67] chips (MMFE8). A connection diagram of these components is shown in Figure 2.12. In total, each Micromegas sector houses 128 MMFE8s, 16 ADDCs and 16 L1DDCs.

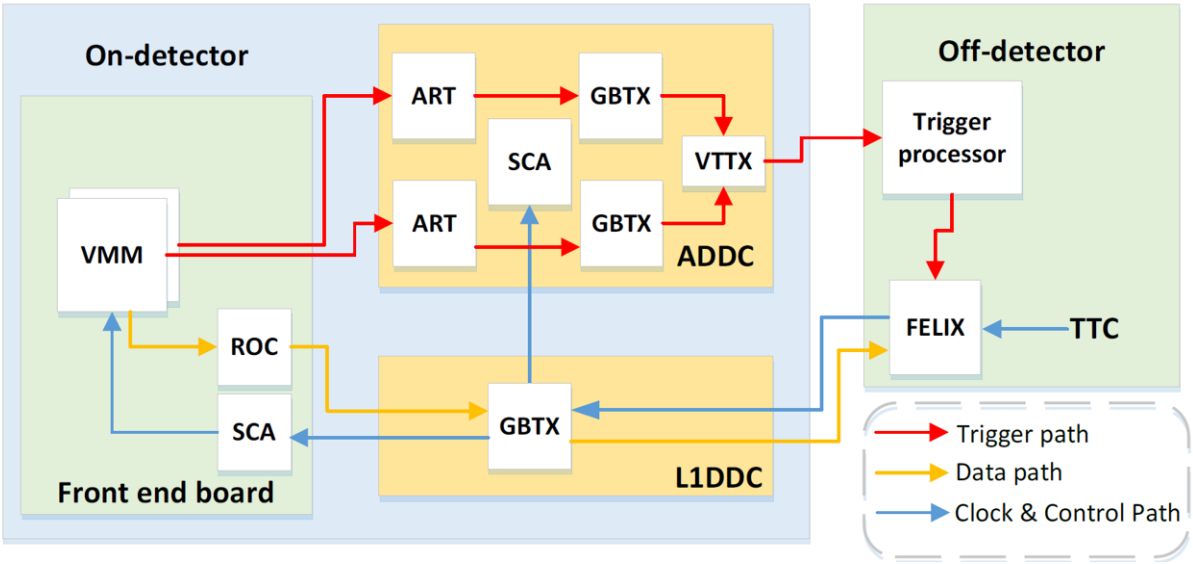


Figure 2.12: The pathing and connections of the Micromegas detector front end boards.

On the trigger path, the ADDC deserializes the ART data received by the VMMs and phase aligns them to a central Timing Trigger and Control (TTC) [68] clock. Then, it identifies which strip is hit and sends the ART data to the MicroMegas Trigger Processor (MMTP) [69] with fibers. Off-detector, the MMTP makes the trigger decision based on the strips that had a hit.

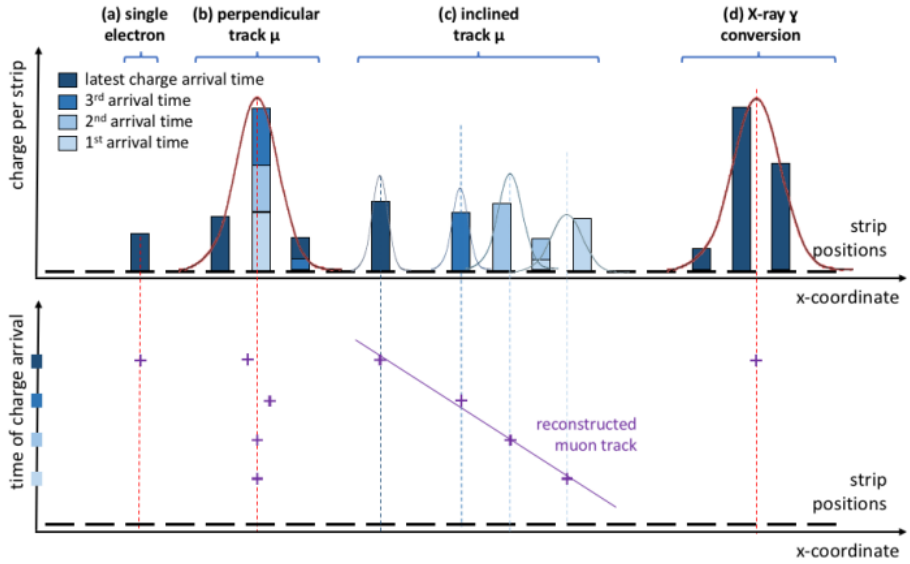


Figure 2.13: Track reconstruction logic applied for four type of tracks: a) a single electron, b) a perpendicular track of a muon, c) a inclined track of a muon and d) an X-ray to γ conversion.

On the data path, the current measured in 2.1 is transferred with fiber via the L1DDC to the FELIX interface [70] which stores all the data. Track reconstruction happens offline with statistical techniques that take into consideration timing and current induced, on which readout strip. A scheme of this logic can be seen in Figure 2.13.

2.2.3 The sTGC detector

sTGC (small Thin Gap Chamber) is the newly designed upgrade of the TGC detectors (Thin Gap Chamber). It is a multiwire gaseous chamber detector that offers very high precisions in angular resolution in a very short amount of time. sTGC technology is deployed as the main triggering technology for New Small Wheel.

Principle of operation

The basic design of sTGC technology is shown in Figure 2.14. The detector consists of an array of 50 µm diameter gold plated tungsten wires held at potential 2.9 kV with 1.8 µm pitch. This structure is sandwiched between two cathode planes located at a distance of 1.4 mm from the plane. The operational gas mixture of the detector is Carbon dioxide (CO_2 − 55%) and n-pentane (45%) [72].

On one side of the anode plane, copper strips that run perpendicular to the wires are responsible for precise coordinate measurements. On the other side, there are cooper pads used for fast trigger purposes. The copper strips and pads act as readout electrodes. Each pad comes with a shielding ground on one side and a thick, 1.5 mm, PCB on the opposite side. The pad occupancy for each colliding bunch of protons in the LHC is expected to be around 1.0−1.3 %. [71]

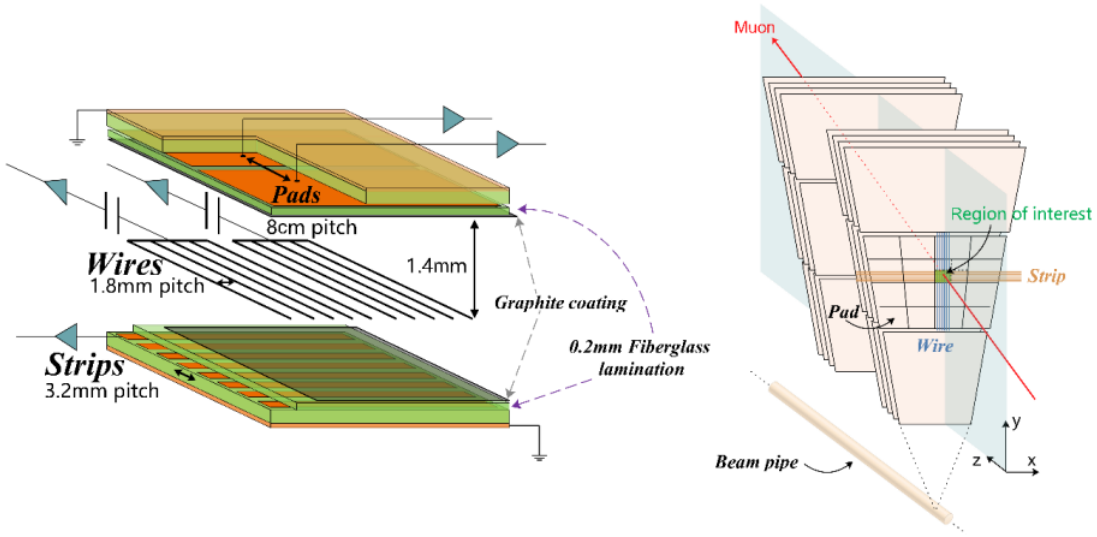


Figure 2.14: Basic design of the sTGC detector technology (left) and orientation of this design in New Small Wheel (right) [73].

The sTGC detector features different drift velocity and current induced on the readout strips but overall, the principle of operation and Townsend discharge process are almost identical.

Data acquisition

The architecture of the sTGC data acquisition is of similar logic to the Micromegas but the hardware is more complex. The sTGC detector requires strip Front End Boards (sFEB) and

pad Front End Boards (pFEB) [74]. Readout data is transferred with the similar, sTGC L1DDC boards [75].

The trigger chain of the sTGC detector has a more sophisticated approach than the Micromegas detector. Trigger data from the front end boards reaches the Pad Trigger [76] (from pFEBs) and the Router [77] (from sFEBs). The Router eventually sends the data to the Trigger Processor. Figure 2.15 displays this layout:

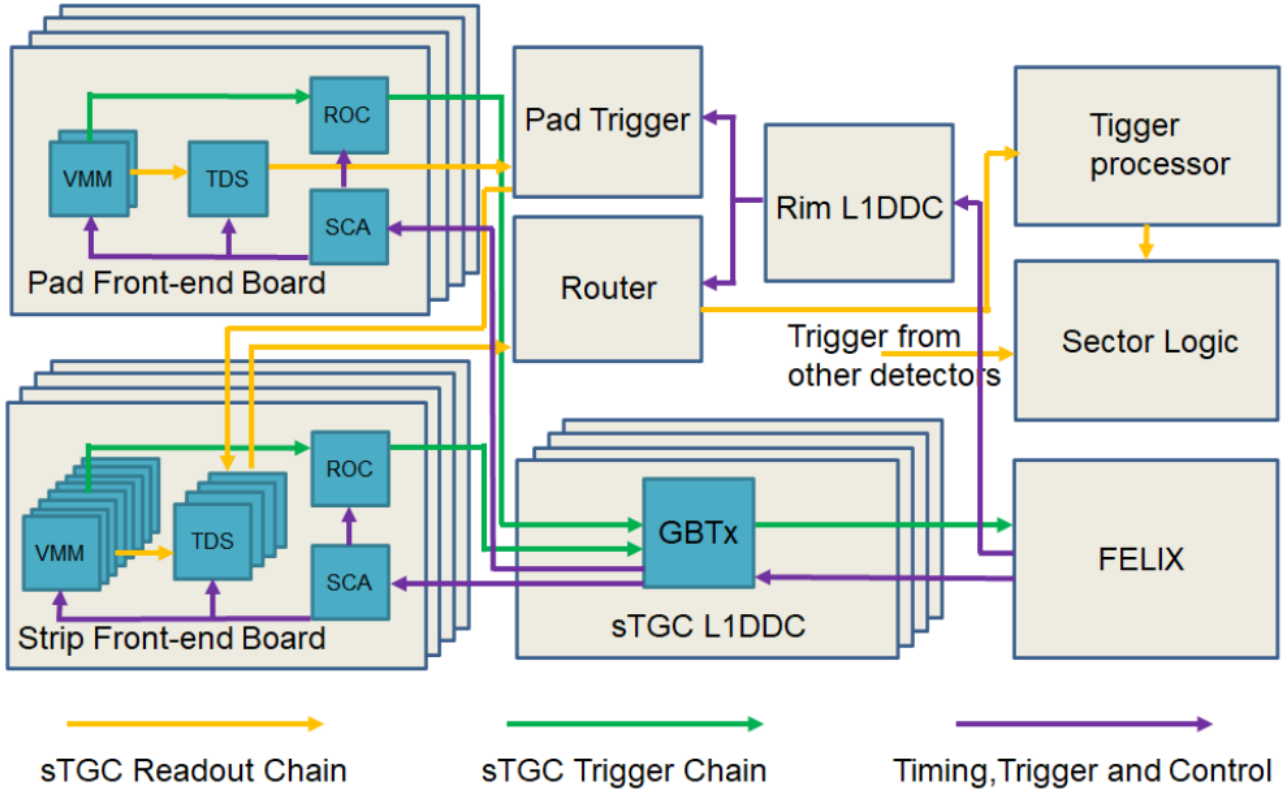


Figure 2.15: Trigger and readout chain of the sTGC Front End Electronics.

2.2.4 The combined technologies: Level-1 Trigger

Figures 2.12 and 2.15 feature the components of each Front End Board which are all custom-made state of the art chips that have been developed to cope with the high volumes of data produced by the upgraded LHC. The Read Out Controller (ROC ASIC) [78] receives event data from the VMMs both for the MMFE8 and sFEB, pFEB. The data is then transferred to a GigaBit Transceiver Link (GBT) [79] which sends the data to the general purpose network with high availability interface, FELIX. All chips interface with the Slow Control Adapter (SCA) [80] which passes configuration and control with TTC signals via a module named as ATLAS Local Trigger Interface (ALTI) [81]. Together, these components lay the foundation for all data acquisition implemented in the above mentioned Front End Electronics for both technologies.

The Front End Boards in both technologies are connected with the Twin-Ax cables [82] and all GBT links are connected via fibers. An overview of all connections, links and components can be seen in Figure 2.16. This scheme represents detailed information of every component together with the final data output in USA15 (Underground Service Area 15). Logic from both technologies is combined to the Sector Logic [83] and all the data is processed offline.

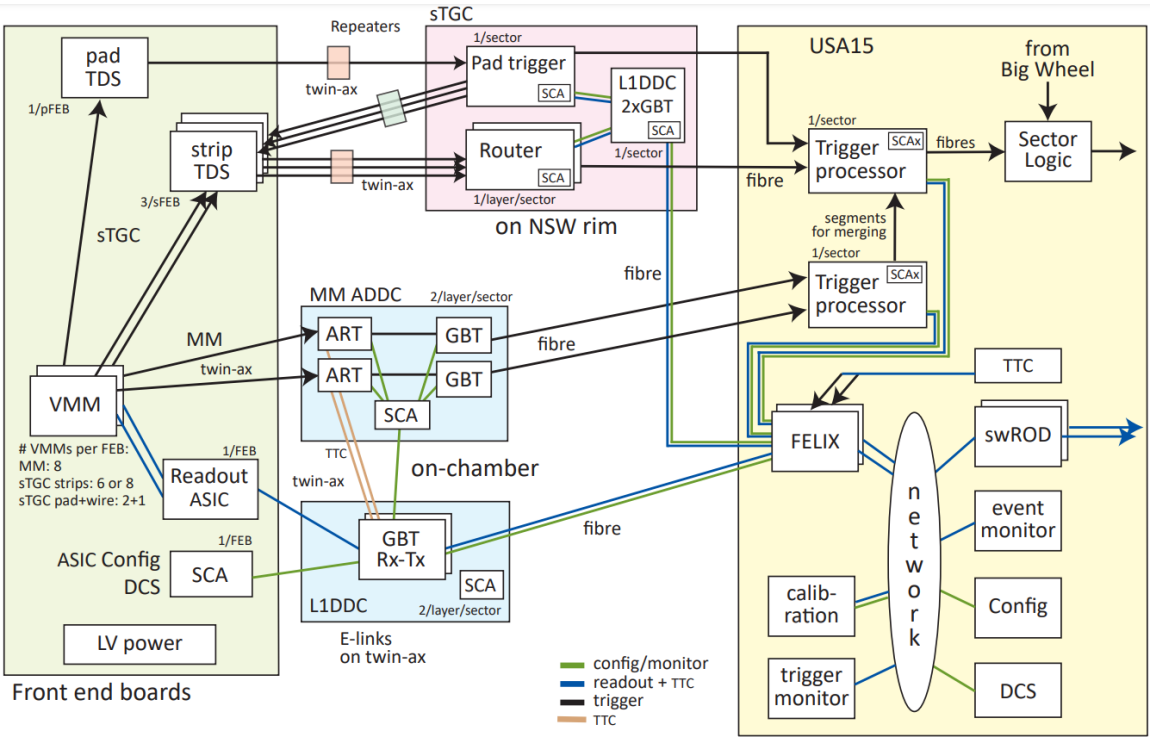


Figure 2.16: Overview of the NSW electronics scheme.

The ATLAS Level-1 Trigger is the first stage of event selection for the ATLAS experiment [84]. A Level-1 trigger decision is made from the combination of both Micromegas and sTGC technologies with a latency of less than 2.5 µs. Signals from the Calorimeter and Muon Trigger System are combined in the Central Trigger Processor. An overview of all sector logic elements can be seen in Figure 2.17.

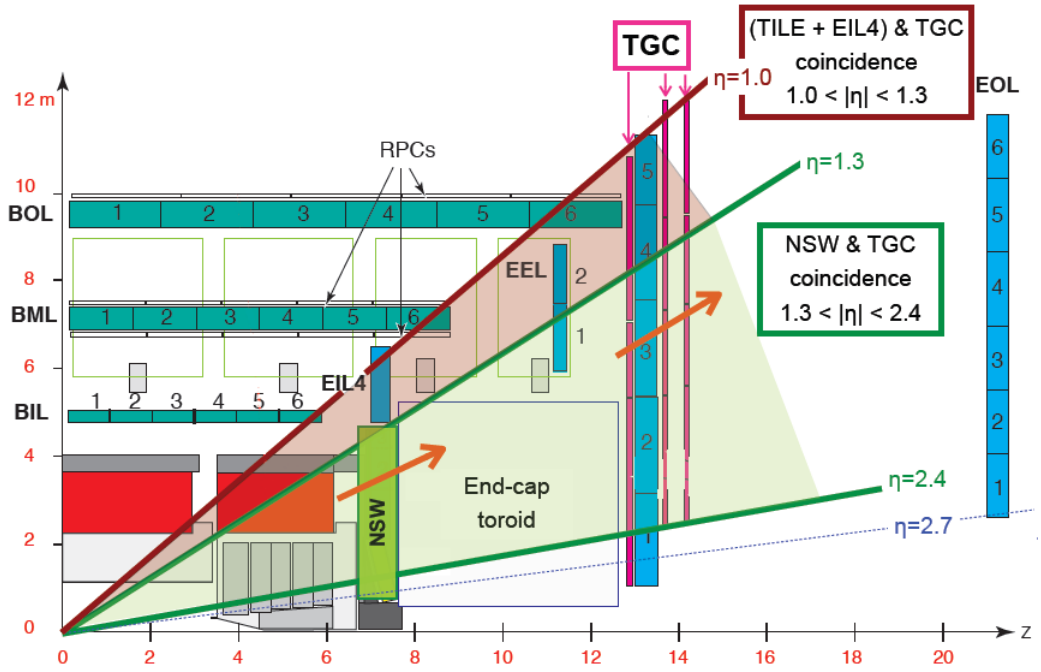


Figure 2.17: Overview of the Calorimeter and Wheels which is taken into account for the Trigger Decision

As of March 2022, more than 5500 scientists from 245 institutes in 42 countries work on the ATLAS experiment [85]. The deployment of all this technology required several years of calibration and testing. At the same time, it was thanks to the effort all the commissioning teams that this project was brought to completion.

Chapter 3

Industrial Control and Automation at CERN

CERN possesses unique know-how in design, building, commissioning, operating and maintaining complex industrial machines and tools. This requires identical expertise in industrial controls and automation in order to facilitate the smooth operation and coordination of all machinery and software.

The industrial solution used to implement all Supervisory Control and Data Acquisition (SCADA) applications is SIMATIC WinCC Open Architecture (WinCC OA) by SIEMENS [86]. CERN must implement custom-made SCADA solutions that conform to each of the experiment needs in a flexible way. Therefore, WinCC OA is the tool (not the control system) for Control System building at CERN.

WinCC OA is an object-oriented SCADA system that allows for solution implementation with various aspects of it's architecture. It uses the scripting language called `Control` which is an object-oriented version of `C` programming language. The user can implement complex interfaces and automations from low level to high level components by scripting on objects. WinCC OA is platform independent and available for Windows, Linux, iOS and Android.

Two key advantages of WinCC OA are its openness and its scalability. WinCC OA allows for integration of a wide variety of components or the implementation of new ones. This integration can take place from the automation level right up to operation and management level. Systems used in experiments are subject to constant change and new demands. WinCC OA is adaptable and new features and components can be added and scaled to the requirements. This can even be applied from the small single-site system for machine operation to the networked, redundant high-end system. Currently, WinCC OA is used by:

- LHC Experiments: ATLAS, CMS, ALICE, LHCb
- Fixed target experiments: COMPASS HARP, NA60 , NA62
- LHC Accelerator: Cryogenics, QPS
- The electrical network of CERN
- CERN's cooling and ventilation

The main content of this thesis describes two SCADA applications (or Detector Control Systems) that were built with WinCC OA. The first one is for the magnetic field monitoring in NSW and the second one is for the Micromegas gas system. In order to make the reader familiarize with the developing process, most parts of WinCC OA will be briefly explained in this chapter. The Detector Control Systems will be thoroughly described in their dedicated chapters.

3.1 Architecture overview of the WinCC OA software

WinCC OA is a modularly built system [88]. The required functionalities are handled by specific units that were created for different tasks. In WinCC OA these units are called managers which are own software processes. This architecture can be distributed with multiple computers forming a system (even with different managers running on each one) that can run also on different operating systems. Figure 3.1 shows all the different WinCC OA managers and their communication:

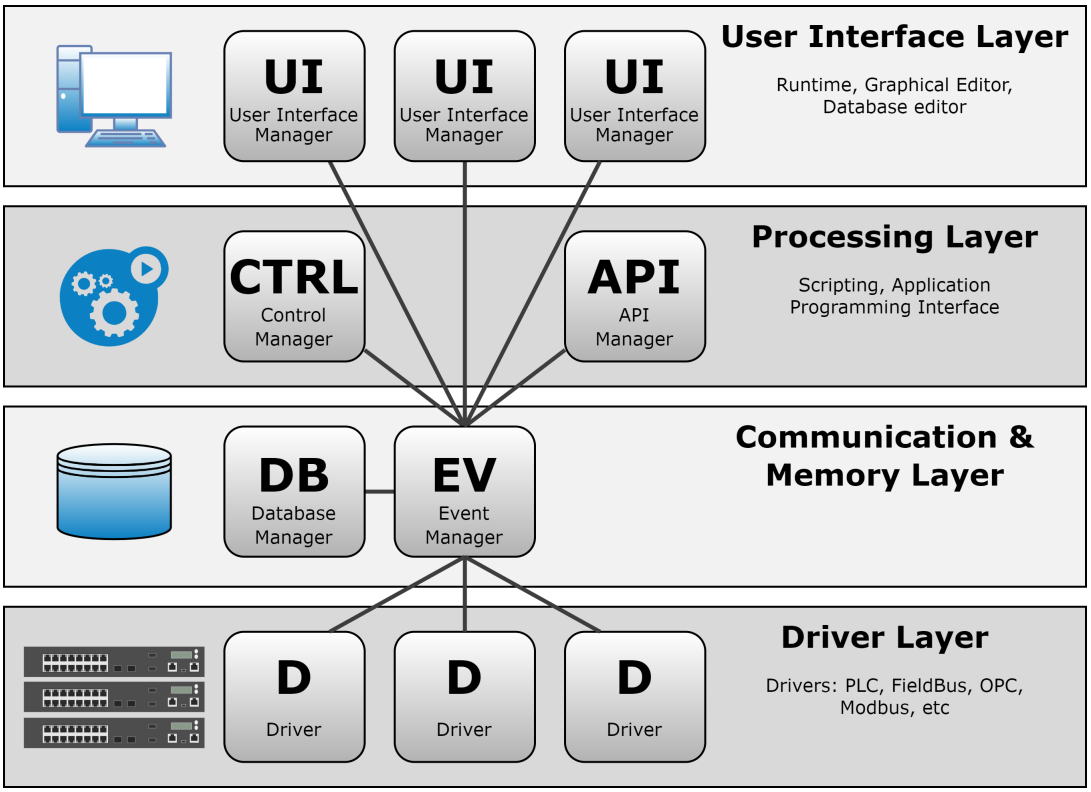


Figure 3.1: All layers of WinCC OA architecture.

The most important managers of the figure above will be briefly described. Although the above diagram looks complex, configuring a manager is quite simple and can be much more comprehensive in practice. In fact, there are a number of managers that are not even dealt with while building an application.

3.1.1 The WinCC OA drivers

The process interface modules, which are called drivers in WinCC OA, form the lowest level in a WinCC OA system. The drivers are special programs that handle the communication with the control and field level. Since there are numerous different communication forms with PLCs or remote control nodes, there are several different drivers available.

The used PLC or the associated communication bus defines which WinCC OA driver should be used. To put it simply, the driver is a module that converts a certain protocol into the internal communication form of WinCC OA. The driver reads current states, measured values or counter values from the field and passes commands and set values to the subordinate controlled units.

WinCC OA drivers are diverse to cover different usages. The implemented drivers are in the form of a manager and the user only needs to run and configure the appropriate manager. These drivers include:

- Serial protocols: RK512, 3964R, Muehbus, ...
- Field bus: Profibus DP, Profibus FMS, ...
- Ethernet: ModbusTCP (OpenModbus), Industrial Ethernet (S7), Ethernet IP, ...
- Telecontrol system: SSI (Ethernet), IEC 60870-5-101, IEC 60870-5-104, ...
- Multivendor interfaces: OPC DA, OPC A&E, OPC UA, ...

A driver can be added in the same way as each other manager to the start list of the console.

3.1.2 The Event Manager

The processing center in WinCC OA is called Event Manager (EV). This unit always keeps a current image of all process variables in the memory. Each other function unit (manager) that wants to access data, receives the data from the process image of the event manager and does not have to communicate directly with a control unit. A command from an operator station is only set as a value change in the process image of the Event manager at first. The forwarding to the appropriate target device (for example PLC) will be executed by the responsible manager automatically.

The Event manager is a kind of central data distributor, the communication center for WinCC OA. Furthermore, this manager also executes the alert handling and is able to execute different calculating functions stand-alone.

The Data Manager supports the Event manager and acts as the link to a database. The Data manager handles the configuration data of an application that is saved in the database. Furthermore, the historical data-value changes and alerts are saved in a database. If an operator needs to query historic data later, the query is also executed by the Data manager and not the database itself.

The concept for archiving process data handles the saving and readout of information that occurs during process control or visualization. This involves values and messages that are generated by value changes. The process data is saved in Value Archives (VA). Each archive is managed by a separate archiving process. Each archive consists of a series of chronologically ordered archive files.

The processing of data and the forwarding between the individual processes (managers) in WinCC OA is event orientated normally. This means that only value changes are processed or forwarded immediately. On the contrary, without value changes, there is no communication or processing in steady operation.

The managers communicate via the TCP/IP message interface. This secure and fully developed communication form also allows a data exchange beyond the computer and operating system limits. The worldwide used standard TCP/IP guarantees highest reliability, compatibility and performance.

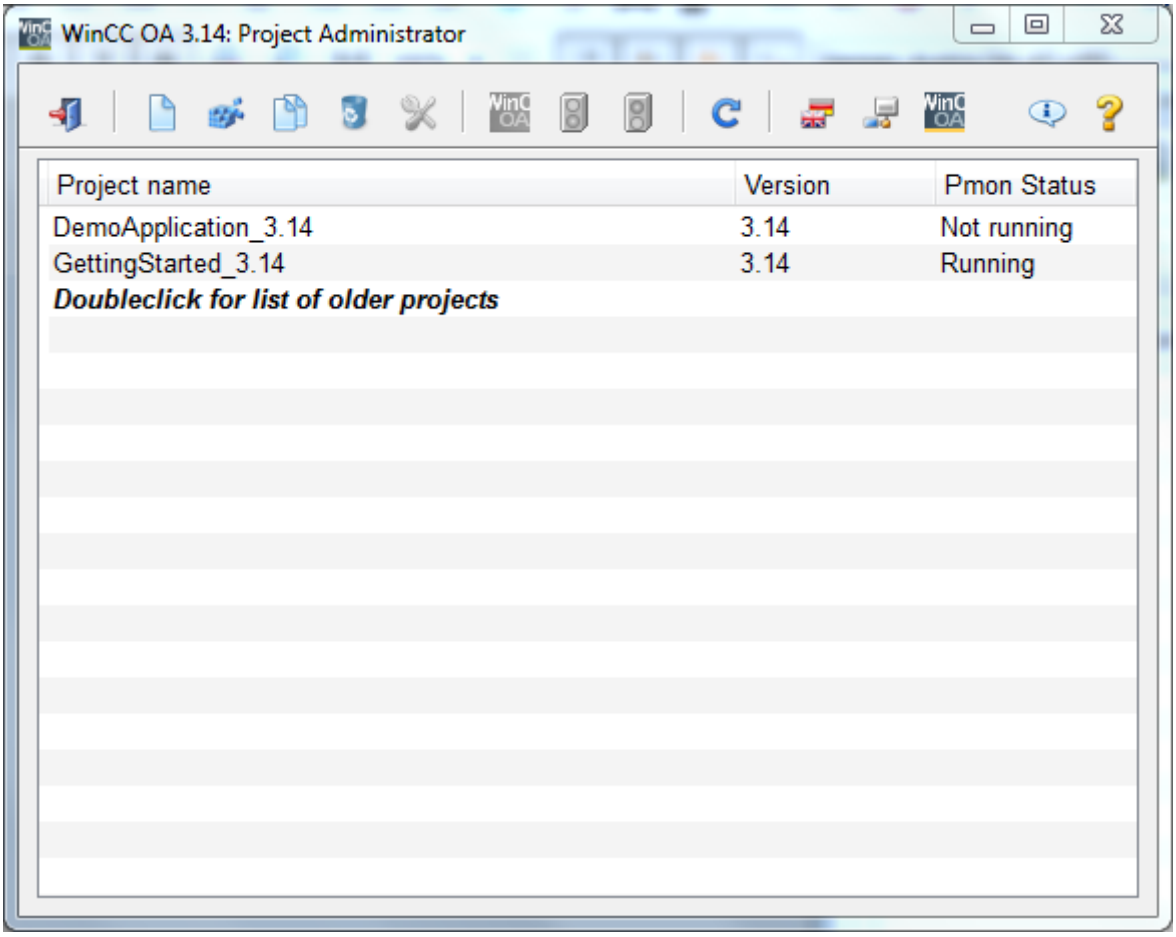
3.1.3 The User Interfaces of WinCC OA

Trends or reports are normally part of the User Interface (UI). In WinCC OA the user interaction is technically completely separated from the background processing. It is merely a view of the data of the current process image or the history.

The User Interface Manager (UI) is the interface to the user. Elements of the WinCC OA developer as well as elements for the operator of the applications developed lay in this layer. In this environment, the user writes the Control scripts for the graphical objects and uses the various menus available for other infrastructure work. Through the UI the settings of most managers can be displayed and configured. However, the most crucial parts of the UI are:

- The graphics editor (GEDI) containing multiple shapes (Control objects) with different attributes and event triggers.
- The database editor (PARA) in which data is stored in "Datapoints"
- The User Interface application window (a module called VISION) which shows values, alarms and executes commands.

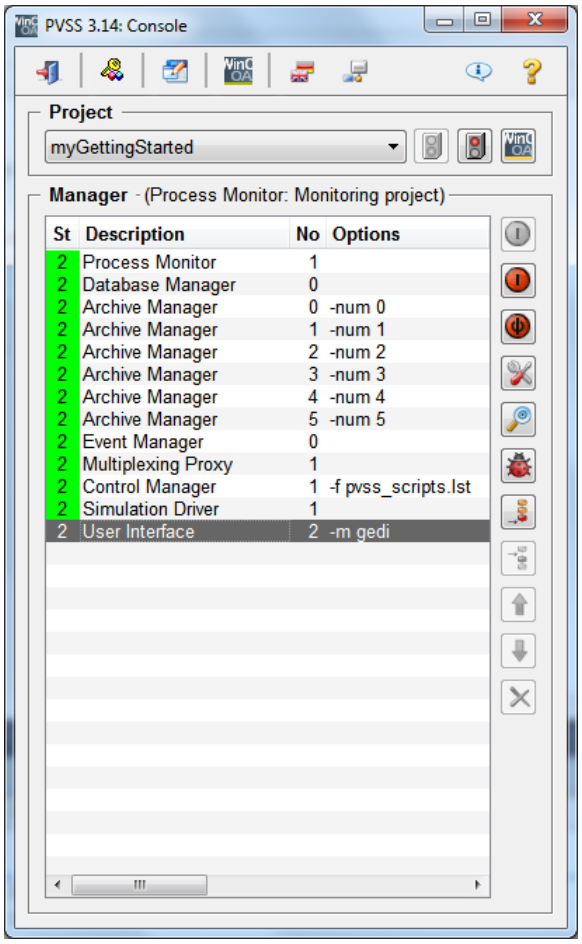
From user's view there are the following relevant user interfaces - the interfaces are listed so that they are easy to understand, regardless of the strict manager architecture. In a typical project process (here using the example of GettingStarted_3.14) there is the following interaction with the user. After starting WinCC OA the "Project Administration" is opened:

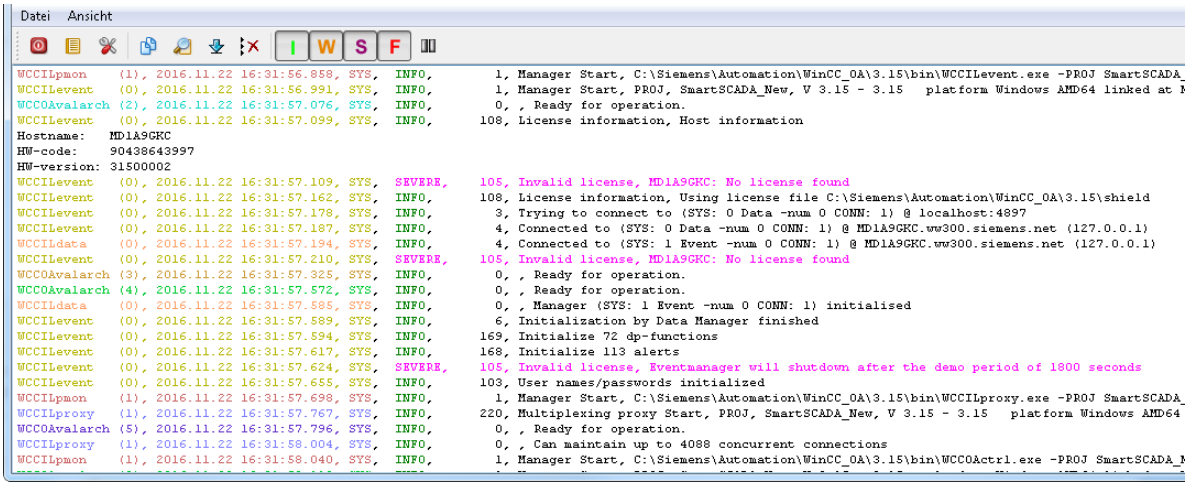


All WinCC OA applications are called "projects". Project administration and console are tools for the project administration. The project administration is responsible for the creation and administration of all projects on a computer. Various settings for each project can also be accessed, like the name, port number, if it is a distributed project or not, etc.

When a project is created successfully, the user can start the project from the project administrator. This prompts the next user interface called "Console" and the "Log Viewer".

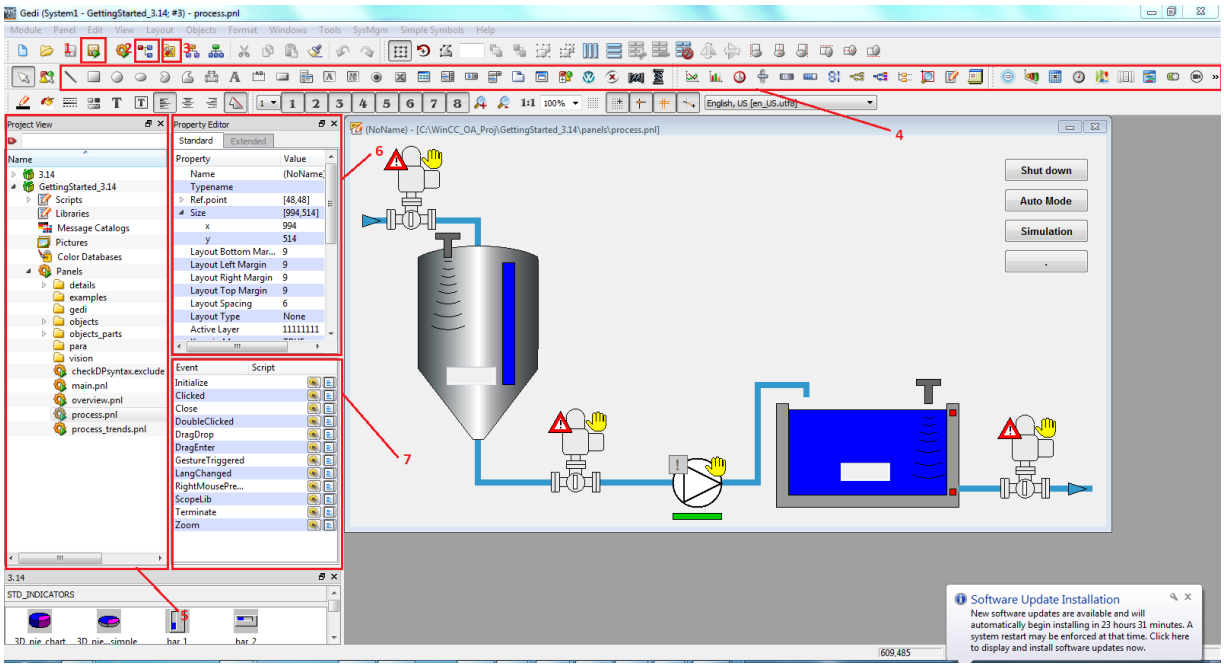
The project console contains all the managers that the project uses to function. It is used for the start and monitoring of a specific project. Through it, the user can add, delete or configure managers into the project. The logviewer is a panel that acts like the output unit for system information of single managers. It also displays compilation errors and is used as a debugger. Both of them can be seen below:





Both the global project administration and the console usually remain hidden for the end user. The log viewer is for the use of project engineers and administrators and isn't meant for normal users of the system.

The next interface is GEDI (the graphical editor). Graphical user interfaces can be created using the graphic editor GEDI. This editor designs everything, from process images or symbols up to operation and information dialogs and also determines the dynamic behavior. GEDI is composed of multiple menus that are available for functions of the project or a graphical object itself. GEDI is shown below:



In the above picture, most important parts of gedi for the developer are listed. They are described below:

1. VISION module. Pressing this button runs the current panel being developed for testing.

2. PARA module. The Database Editor panel is prompted when this button is pressed and the list of Datapoints is displayed. PARA will be separately described in the next subsection.

3. System Management module. Various settings of communication, drivers or WinCC OA reside here. The system management is another project specific administration user interface. It is used to set a number of project specific settings. Furthermore, different administration and diagnosis tools can also be accessed through the system administration.

4. This is a list with some of the most useful (but not all) WinCC OA graphical objects. Like in most editors, the user can select an object and draw in the panel section of GEDI.

5. Project Viewer. This part of GEDI interface shows all the files and folders that are inside WinCC OA and current project.

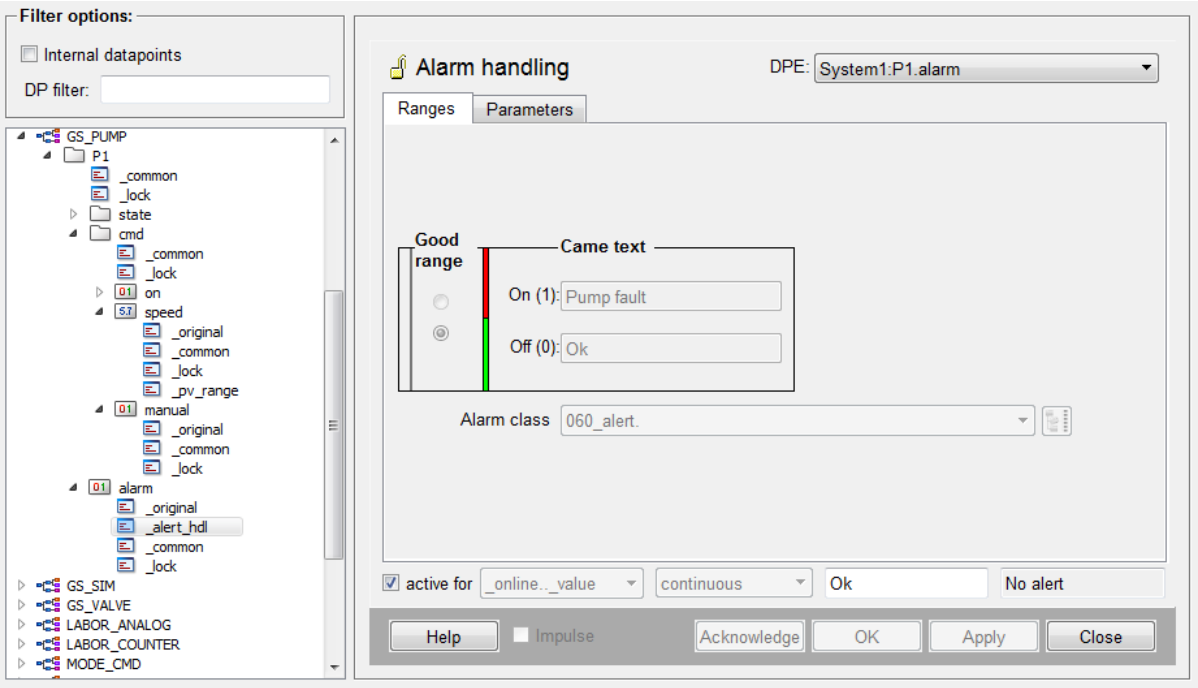
6. Property Editor. This menu provides all properties of the graphical object class in question. Selecting a graphical object will prompt this area with all of its properties. The user can use this window to either edit a property directly or to see a property name and edit it through a script.

7. Event Editor. This menu is also graphical object-specific. Each graphical object has some event triggers that can be scripted either with a default wizard or by hand. Most of the time, scripting by hand is the preferred way to achieve the desired outcome as it is more precise.

The GEDI module hosts many more interfaces or shortcuts to settings which are not necessary to be described here. A usual development cycle includes the developing of a graphical panel with graphical objects, scripts of the objects or the panel itself and testing of the whole interface.

3.1.4 The Database Editor

The Database Editor (PARA) is perhaps of equal importance to GEDI and thus it is described in a separate subsection. PARA can be accessed through GEDI and is shown below:



The module PARA of the WinCC OA User Interface Manager is a graphical interface for editing Datapoint Types and Datapoints. With this tool the user can access the internal database and make modifications simultaneously. The Datapoint Type with its structure serves as a template for the similar Datapoints derived from it.

Datapoints in process-control system conform to requirements of concrete counterparts of a system. This means that an engine or a valve Datapoint contains a structure, which conforms to requirements of a particular engine or valve. The structure for a particular engine type or valve type is the Datapoint Type. On the left of the above picture is the list of Datapoint Types followed by the Datapoint branches and their Datapoint Elements.

Configs are groups of properties that may be assigned to data point elements in WinCC OA. These properties are defined with the aid of config attributes that carry the actual data point values.

Datapoint Types and Datapoints are of similar logic to a `STRUCT` in `C` programming language. Datapoints are instances of a Datapoint Type and contain structure. Data is finally carried by a Datapoint Element which can be of any Control Data Type. More on data types will be explained in the next section.

Except from using Datapoints as a means of storing data, the user can implement actions and configurations between Datapoints and the WinCC OA environment. Original values of a Datapoint element can be converted, replaced with predefined values or have an alert handler. The user can also smooth out and archive changes in values of variables. Configs are also used

to connect Datapoint Elements to periphery addresses. Some of the most important configs are listed below:

Datapoint Config	Outline
`_address`	Used to manage the peripheral addresses of Datapoint Elements. Drivers can be selected and configured.
`_alert_class`	Contains the parameterizable properties of the alert ranges of data point variables. As a default WinCC OA provides 5 alert classes.
`_alert_hdl`	If an alarm needs to be sent it is necessary to define the alert range, alert text, etc through this config
`_archive`	Used to archive values of Datapoint Elements for trends or reports. Data may also be compressed.
`_dp_fct`	In order to implement mathematical links of Datapoints, a function is assigned to a Datapoint via this config. The Datapoint function contains one or more parameters used to calculate the value of the function. There are also numerous statistical functions available.
`_smooth`	Data supplied by the periphery can be smoothed in the driver before being further processed as original values in the control system. Smoothing is used to reduce the amount of communication and data volume in the system.

3.2 Introduction to the Control (CTRL) language

A Control program (script) is used to control WinCC OA and can be programmed by the actual user of the control system. In this way the user is employing a script to specify the response of WinCC OA to an event. The syntax of the internal controller language "Control" has a similar structure to that of the C programming language.

These programs are capable of multi-program operation (multi-threading). Blocks can be defined in the CTRL script that are processed when a specific event occurs (for example, message that a data point attribute has changed). Since WinCC OA can process more than one data point at once, the Control program must also be capable of letting different processes run in parallel. The Control program must perform the following functions:

- Control functions for technical processes
- Enable dynamic handling of graphics objects

The Control program processes data points and controls the visualization of process states. The Control syntax is based on the procedural programming language C. Control, however, is implemented as an interpreter, so that the user does not have to perform compiling or linking. This means that changes can be tested immediately. Control provides optimum support for the flexible data point concept of WinCC OA.

3.2.1 Structure of CTRL scripts

Each Control script contains a `main()` function. This function is processed first when a CTRL script has ran. The way in which scripts are called from graphical objects depends on what type of graphics object attribute the script has been written for, namely whether it is a "passive" or "active" attribute.

Scripts for passive attributes

Passive attributes of a graphical object are for example its border color, its visibility, the font used, etc. They are used for visualizing process states and are thus not necessarily associated with a user input. Scripts for passive attributes are run when a panel is opened (event: Initialize), so their `main()` function is processed once at this time.

Below is an example of the Control function `dpConnect()` used to register the interest of the `add()` function in the Datapoint Elements. Every time the online value of the Datapoint Element A or B changes, the `add()` function is executed.

```
main() {
    dpConnect("add", "A.:_online.._value","B.:_online.._value");
}

void add(string dp1, int a, string dp2, int b) {
    dpSet("C.:_original.._value", a + b);
}
```

This example calculates $C = A + B$. The name of the work function is `add()`. The parameters passed are `"A.:_online.._value"` and `"B.:_online.._value"`. The Datapoint Element C is set by a `dpSet()` command.

Scripts for active attributes

Active attributes are those that require a user input. Examples of such are the "when clicked" attribute that is triggered by a mouse click on the object and attributes for the "command" fields in the attribute editor that send text input or messages from radio and check boxes. Scripts for active attributes are executed whenever the user has activated the object, i.e. clicked on it with the mouse, or entered text (relative events in Event View). The `main()` function is rerun each time. In general, it is not necessary to register Datapoint Elements, although it can still be done.

3.2.2 Datapoints, Data types and Data Structures

As it has already been mentioned, Datapoint Types and Datapoints are of similar logic to a `STRUCT` in `C` programming language: Datapoints are instances of a Datapoint Type `STRUCT`. Datapoint Elements can be value fields and thus can be of a Control data type.

Control supports all the basic data types of most programming languages. Numerical data types can be:

Name	Value Range	Comment
`int`	`-2147483648` to `+2147483647`	32-bit integer value.
`uint`	`0` to `+4294967295`	32-bit positive integer value.
`double`	`-1.79769e+308` to `+1.79769e+308`	64-bit floating point value.
`float`	`-1.79769e+308` to `+1.79769e+308`	32-bit floating point value.
`long`	`-9223372036854775808` to `+9223372036854775807`	64-bit integer value.
`ulong`	`0` to `+18446744073709551615`	64-bit positive integer value.
`unsigned`	`0` to `4,294,967,295`	Positive integer value.

The following logical data types are also included in WinCC OA:

Name	Value Range	Comment
`bit32`	0 to +4294967295	32-bit pattern.
`bit64`	0 to +10146744073709551615	64-bit pattern.
`bool`	0 or 1, `true` or `false`	1-bit binary value.

WinCC OA also features some other rather interesting data types like:

Name	Comment
`anytype`	Any of the WinCC OA data types depending on the first variable assignment.
`errClass`	Specifies the error classes.
`file`	File pointer.
`mapping`	Saves arbitrary key/value pairs. The pairs are saved in two arrays (like a dictionary).
`time`	Internal time format.

The character specific data types are also available, `char` and `string`.

Another difference of WinCC OA Control and C programming language is in data structures. One data structure has already been mentioned and described above, `mapping`. In Control, all arrays are dynamic and can be of any data type. The default word to assign a type as an one dimensional array is `dyn_<data type>`. A two dimensional array would be `dyn_dyn_<data type>`. Some examples are: `dyn_int`, `dyn_float`, `dyn_dyn_string`, etc. There are also internal WinCC OA functions that can create dynamic arrays like this:

```
main() {
    dyn_int a = makeDynInt(5, 8, -2);
    dyn_float c = makeDynFloat(2.68, 12.3, -12000.01);

    <code body>
}
```

Thus, in WinCC OA it is possible to create Datapoint Types and Datapoints by using the internal script language Control and using the above data types. Appropriate functions are available for accessing, creating as well as configuring Datapoint conifgs for these value fields. Datapoints or individual configuration of them can also be deleted by using scripts. Because of the online configuration of the system, such processes can be executed without stopping the control station. The most essential functions for the script controlled engineering of Datapoints are `dpTypeCreate()`, `dpCreate()`, `dpCopy()`, `dpDelete()`, `dpSet()` and `dpSetDescription()`. To receive data from a Datapoint element the function `dpGet()` is used. Similar functions are also available for the configuration of Datapoints config. More information on these functions and their usage is available in the WinCC OA help module and online.

3.2.3 Operators in Control

Arithmetic operators

The binary arithmetic operators are `+`, `-`, `*`, `\` and the modulus operator `%`, which returns the remainder after an integer division. The `%` operator cannot be used on float values. Only the `+` and `-` operators are allowed for time values.

Increment and decrement operator

The increment operator is `++`, placed either before or after the variable and the decrement operator is a preceding or following `--`. These operators, which can only be used on variables, increase or decrease the variable value by one. If this type of operator is placed in front of a variable its value is increased (or decreased) before it is used. If an increment (or decrement) operator is placed after a variable then its value is increased (or decreased) only after it has been used.

Conditional operator

```
expr1 ? expr2 : expr3
```

Depending on `expr1`, either `expr2` or `expr3` is evaluated and returned as a result. This is similar to the control structures `if ... else`

Relational and logical operators

The relational operators are `>, >=, <, <=, ==, !=`. The logical operators are `&&` for logical `AND` and `||` for logical `OR`. The `!` character is the negations operator `NOT`.

Bit-wise operators and shift operators

The bit-wise operators are `&` for bit-wise `AND`, `|` for bit-wise `OR` and `^` for a `XOR` operation (bit-wise exclusive `OR`). $\sim$ is the negations operator (NOT, bit complement). The shift operators $<<$ and $>>$ can be used to shift to left or right respectively all bits in the binary representation of the left-hand operand by that number of bit positions specified in the right-hand operand. Zeros are added to the end. The right-hand operand must be positive.

Assignment operators

In the expression

```
i = i + 2
```

the variable on the left-hand side is repeated immediately at the beginning of the right-hand side. This can be written in a more compact form as

```
i += 2
```

The operator `+=` is an assignment operator. Other assignment operators are `-=`, `*=`, `/=` and `%=`. There are also assignment operators for bit-wise operators. Thus `a = a & b` could also be written as `a & = b`. The operators `|=` and `^=` also exist. Shift assignment operators can be implemented with `«=` and `»=`.
Multiple assignment is not supported by Control.

3.2.4 Loops and conditional statements

The standard loops of `C` programming language are included in control:

- `for` loop
- `while` loop
- `do ... while` loop

Conditional statements are also included:

- `if ... else if ... else` statements
- `switch ... case` statements

3.3 WinCC OA at CERN - The JCOP framework

The Joint COntrols Project (JCOP) is a collaboration between the LHC experiments, the EP Department and BE-ICS, the Industrial Control Systems Group in the Beams Department [89] [90]. The original mission of the JCOP framework was to implement control solutions in common by the LHC experiments and the abovementioned departments. The project proposal can be found in [91].

JCOP works in parallel with WinCC OA and has the responsibility to support and maintain a common Framework of tools and components. Many subdetectors of the LHC experiments have similar or common functionalities that can be automated and generalized in configurable tools. The JCOP Framework has worked with various projects and studies, the four major ones being:

Sub-project	Comment
DSS	Detector Safety System
Framework	Tools and Guidelines for implementation
GAS Control	In collaboration with Gas Working Group
Rack Control	In collaboration with EN/EL

JCOP Framework tools and components can be installed through a specific tool called `fwInstallation Tool`. The user can then install the tools or devices required for their application. A number of tools have been implemented for use, like Access Control, OPC UA Tools, RDB Archiver, Trending Tool, XML Parser, FSM (Controls Hierarchy), DIM/DIP componenets, etc. Many devices that are also used broadly around CERN are also installed (their drivers) as components in WinCC OA like CAEN Devices, the Embedded Local Monitor Boards (ELMB), PDU Devices, etc.

ATLAS uses a variety of these tools and components but the most important ones that are also used in this thesis are the Trending Tool and the FSM, DIP components.

3.3.1 The Trending Tool

Trending and visualization of monitored variables or control parameters is very essential for every control system. WinCC OA features a default way to create trends with time series which poses limitations in the scale of the LHC experiments and their requirements. For this reason, a trending tool designed for CERN experiments' needs is designed with the following features:

- Trending tool users: The configurator who prepares the content and the observer who uses already prepared trends.
- Configuring pages (collections of plots): Availability to define parameters like trend name, number of plots, size, color, axis labels, axis scales, datapoint participation, mechanism to replicate trends, etc.
- Configuration UI: All variables mentioned above should be configurable through some User Interface.

- Framework library: The actions desired for the trend must be able to be completed through library functions to allow for automations.
- Tree organization: Trends should be able to be organized in a structure that conforms to the data layout.

All these requirements are finally packaged in a JCOP Framework Tool which is available for CERN users to include in their work. The latest release of the Trending tool was version `8.4.0` in `27-11-2019`. This thesis uses this version of the tool for all monitored variables in the interfaces.

3.3.2 The Finite State Machine (FSM) Tool

Finite State Machines in computer science

There are many ways to describe the functionality of a system in computer science but one of the oldest, best known and successful is the use of Finite State Machines (FSM). FSMs are very useful generally in computer science and especially in control systems.

State machines allow us to think of a system in terms of it's state at a particular point in time and characterize it's behavior based on that state, just like the physical states of a material i.e. liquid, solid or gas. In computer science, a similar technique of categorization is useful to model and design software systems by their state. Generally, there are several key characteristics of a system that can be modeled with an FSM:

- The system must be describable by a finite set of states.
- The system must have a finite set of inputs and/or events that can trigger transitions between states.
- The behavior of the system at a given point in time depends upon the current state and the input or event that occur at that time.
- For each state the system may be in, behavior is defined for each possible input or event.
- The system has a particular initial state.

The FSM modeling of a system is for this reason very closely associated with control systems that usually include sub-systems and control hierarchies. CERN detector, subdetector and subsystem hierarchies conform to this model and thus an FSM Tool for WinCC OA needs to be developed.

FSM hierarchy at CERN

The architecture of the FSM Tool needs to hierarchically design the tree like structure of the detectors and their subsystems. Three main components are adopted:

- The Device Unit (DU): It is capable of monitoring and controlling the equipment to which it corresponds.
- The Control Unit (CU): It contains the Finite State Machine(s) that model and control the behavior of the sub-tree below them.
- The Logical Unit (LU): It contains the Finite State Machine(s) and can contain children but not Control Units. It can monitor the states of it's children but cannot pass control commands.

In general, an FSM hierarchy of a system would consist of CUs at the top of its hierarchy and LUs at the bottom. The lowest parts of the hierarchy must always be DUs that correspond to some WinCC OA datapoint.

In the FSM hierarchy the two aspects that work in parallel and provide all the necessary information of a subsystem are "STATE" and "STATUS" of a unit. The STATE defines the operational mode of the system (i.e. "RAMPING_UP", "ON", "OPEN", etc.) and the STATUS gives more details about how well the system is working (i.e. warns about error or malfunctions). ATLAS DCS has published the FSM Integration Guidelines [92] that need to be followed by all subdetectors and thus, all STATUSES and STATES must conform to the following conventions:

STATE For consistency, common control areas like High Voltage, Low Voltage, Cooling, Gas, etc should have similar states, status, transition and actions in a manner that will not confuse the operator. The color coding for the state is as follows:

Color coding	Description
GREEN	Static state. The system reached the final operation stage.
BLUE	Static state. The system did not reach the final operational stage yet.
DARK BLUE	Static state. The system is in its lowest operation state.
TURQUOISE	Static state. The system is in the STANDBY state (safe for LHC unstable beams).
LIGHT BLUE	Transient state. Signalizes an ongoing transition between normal states.
ORANGE	Error state. Used if state is UNKNOWN (i.e. due to loss of communication).
RED	Severe Error state. For example TRIPPED in case of a power supply trip.

The next table displays a list of states and associated commands defined in the FSMs:

State	Commands
READY	GOTO_READY
NOT_READY	GOTO_SHUTDOWN, GOTO_READY
SHUTDOWN	GOTO_READY
UNKNOWN	RECOVER
STANDBY	GOTO_STANDBY
ON, ON25, ON50, ON75, ...	GOTO_ON, GOTO_ON<xx>
OFF	GOTO_OFF
TRIPPED	RECOVER
LOCKED	UNLOCK, LOCK
RUNNING	RUN
STARTED	START
STOPPED	STOP

The above commands are also associated to the transition states:

Transient States	Commands
RAMPING_UP, GOING_TO_ON, ...	GOTO_ON<xx>, GOTO_STANDBY
CALIBRATION	CONFIGURE
GETTING_READY	GOTO_READY
GETTING_NOT_READY	GOTO_NOT_READY
STARTING	START
STOPPING	STOP

The above STATES are conventions used among the sub-systems of ATLAS but it is up to the developer to name states that better describe other implementations.

STATUS The STATUS follows a strict convention where the names are fixed and all sub-detectors must use these qualifiers. An FSM STATUS is similar to the logic of an alarm within a control system so displaying it in the FSM is useful for faster propagation of information.The colors and names decided in the Framework are:

Status	Description
OK	Healthy status. System is operational with the desired functionality.
WARNING	Low severity. System can continue working but a fix needs to be followed up.
ERROR	High severity. Operation not nominal, a fix needs to be followed up as soon as possible.
FATAL	Very high severity. The system does not function properly or cannot function properly.

An FSM tree node always has a STATUS and a STATE or else it is defined by default as "UNINITIALIZED". The developer can write an initialization script like this:

```
xyzAtlasExampleDeviceMixedDuFsm_callback(string nodeName, string tnode,
                  string dpe1, float value1, string dpe2, float value2,
                  string flagDPE, bool flag, string pingDPE, bool ping)
{
    string domain, device;
    fwFsmAtlas_getNodeNameComponents(nodeName, domain, device);

    // determine State
    string state = "impossible"; // should be impossible
    if (!ping) state = "UNKNOWN";
    else if (flag) state = "TRIPPED";
    else if (value1 <= 0.) state = "OFF";
    else if (value1 < 10.) state = "RAMPING";
    else state = "ON";

    fwFsmAtlas_setDUState(domain, device, state);

    // determine Status
    string targetStatus = "OK";
    if (value2 > 2.) targetStatus = "FATAL";
    else if (value2 > 1.) targetStatus = "ERROR";
```

```
    else if (value2 > 0.5) targetStatus = "WARNING";

    fwFsmAtlas_setStatus(domain, device, targetStatus);
}
```

In the above example, the STATE and STATUS are calculated with regard to `value1` and `value2`. It is evident, that the tool already implements a library of functions that update STATE and STATUS: `fwFsmAtlas_setDUState(Status)`. The FSM Tool implements several libraries with different usages for the FSM:

- `fwDU.ctl`: used to add functionality to DUs inside their scripts
- `fwCU.ctl`: used to operate CUs from panels
- `fwFsmTree.ctl`: used to dynamically create a hierarchy of CUs, LUs and DUs
- `fwFsmUI.ctl`: used to tailor the look and feel of FSM panels trees

Each FSM node is also associated with a WinCC OA panel and thus an operator navigating in the FSM tree will see a panel associated with each node. Such configuration can be accomplished both with the libraries mentioned above but also with user interfaces developed by the Framework. However, to dynamically produce the desired layout, the FSM libraries are preferred instead of the manual by hand configuration of each node. The ATLAS FSM is a huge hierarchy composed of many FSMs for each subdetector that are glued together. As an example, the MUON FSM can be seen below in Figure 3.2:

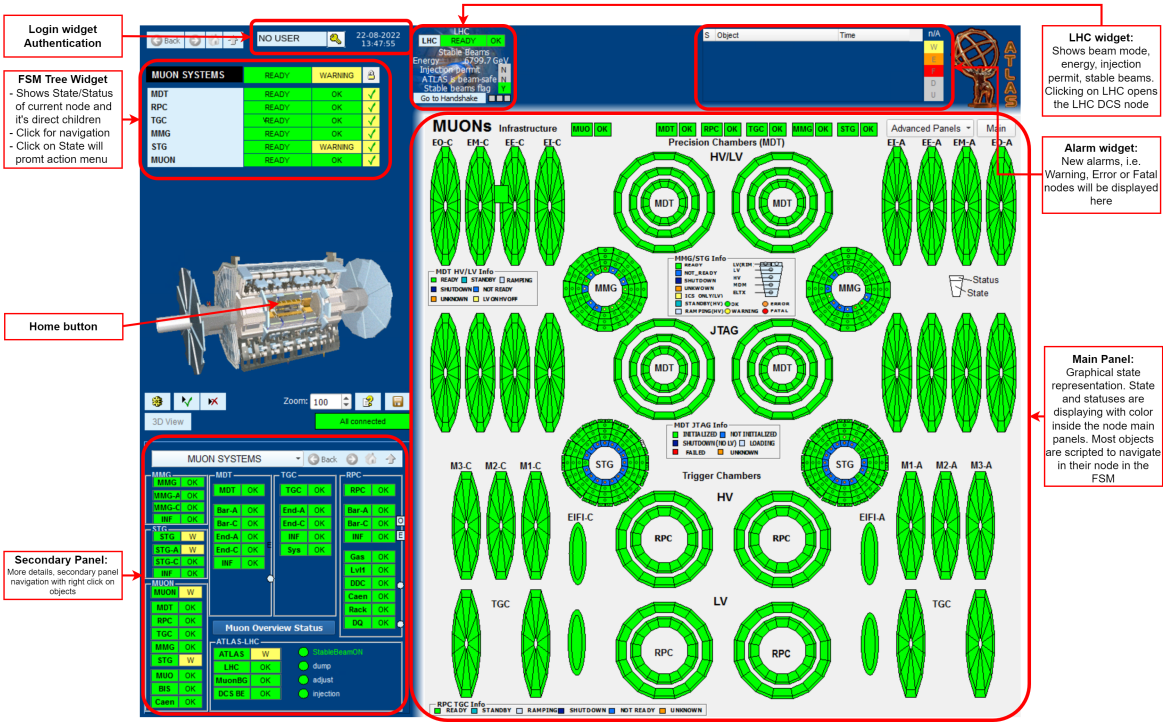


Figure 3.2: The FSM of MUON detectors in ATLAS experiment DCS. [93]

Chapter 4

Monitoring of the MicroMegas gas system

The main functionality of a detector gas system is to create a gas mixture from the different gas components in the appropriate percentages and eventually distribute it to the detector chambers. In order to facilitate the construction and later on, the maintenance, all detector gas systems are designed with functional modules of similar properties. Some example modules are: a gas mixer, the pre-distribution, distribution module, circulation pump, purifier, humidifier, membrane, liquefier, analysis, etc [101]. In this chapter, the MicroMegas gas system will be described together with the control system developed to monitor each of it's physical parameters.

4.1 The MicroMegas gas system in ATLAS experiment

In ATLAS experiment, gas is typically being distributed over three levels: pipes feed the gas from the surface building to the underground service area and then the gas is distributed in the experimental cavern racks. The gas mixture is then finally distributed into the detectors. A representation of this design is shown in Figure 4.1.

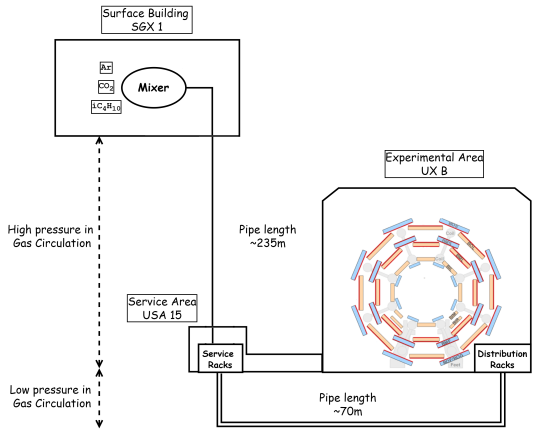


Figure 4.1: Typical layout of the MicroMegas gas system.

Each module of the gas system will be described in order to get an understanding of the monitored hardware. More details on the mechanism that describes the gas flow in the MicroMegas chambers can be found in [99] and [98].

4.1.1 The Mixer module

The gas mixture stability is perhaps the most basic requirement for good and stable long term operation of all detectors. The primary task of a mixer module is to provide the gas system and ultimately the detector with the suitable gas mixture for nominal operation. During particular phases it might also be needed to supply different gases (for example for purging the detector when the standard gas mixture needs to be evacuated, like at the beginning of a long shutdown period) or standard mixture at very high flow (for example when the detector is going to be restarted after a long shutdown, filling mode).

It has been decided that the MicroMegas detector will operate under the gas mixture of $Ar(93\%) - CO_2(5\%) - iC_4H_{10}(2\%)$. Thus, the mixer module of the detector is composed of three lines, one for each gas component. The final mixture is then assembled at the mixer tube and fed to the rest of the gas system. A detailed design of the components is shown in Figure 4.2:

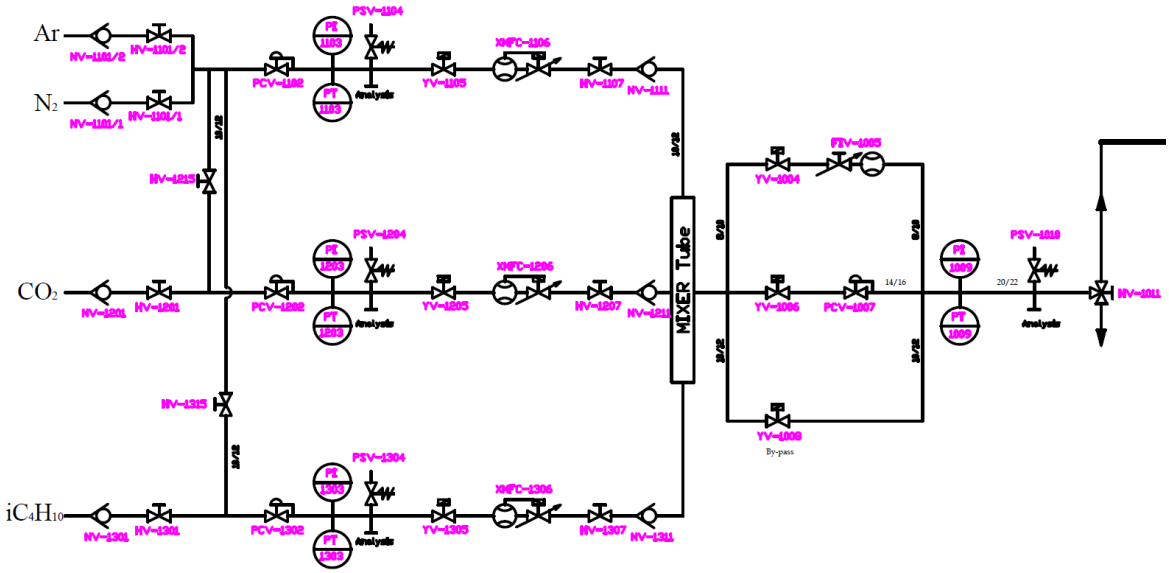


Figure 4.2: Every component of the mixer module's design at SGX1 surface ATLAS building. This schematic uses CERN's conventions for piping and instrumentation drawings. The same convention is used for all diagrams of this nature.

Each gas line starts with a non return and a control valve which pushes each gas component to the next parts of the mixer. Then, a PCV (Positive Crankcase Ventilation) pneumatic valve equipped with a pressure regulator controls the gas emissions and pressure. The gas is then monitored by the PI (Pressure Indicator) and PT (Pressure Transmitter) and then an analysis line is connected in parallel. The gas moves through two valves with solenoid actuators: one state valve and one xMFC (Mass Flow Controller) valve to regulate the flow. Finally, the gas meets a hand actuator valve and a non return valve before entering the mixer tube. The gas mixture is then feed to the rest of the gas system through a state and an FIV (Flow Indicating Valve) valve. The pressure of the gas mixture is then measured with similar PI, PT components.

All values (pressures, flows, mixing ratios, etc.) of the mixer module are monitored and they are also used to generate alarms or interlocks to the MicroMegas gas system. At the same time, they are used to calculate the state and status of the mixer module in the gas system. All of these are described by the functional analysis requirements, according to the MicroMegas needs and will also be explained at the relevant section that describes the control system.

4.1.2 The Analysis modules

Gas analysis modules are used to continuously monitor with automated cycles the gas mixture composition. Two types of modules were developed: the first one makes use of standard electro-valves for the selection of the gas stream, while the second one uses special pneumatic n-way valves [96].

The analysis module is completely automated: it can be programmed to sample the gas stream including references or calibration gases. Expert operators can trigger remotely the analysis of specific lines at any moment. The analysis time of each stream can be tuned in order to compensate for the different pipe length between the analysis rack and the sampling point. Usually, the software control is configured to generate alarms and to exchange data with the specific detector DCS. For the MicroMegas detector, currently, the analysis modules are not connected to the DCS.

In other detector technologies, analysis modules are also used for safety purposes, i.e. like for continuously monitoring of the mixture flammability level.

In particular, the MicroMegas detector has two analysis modules, one that analyzes the gas mixture directly after the mixing (at SGX1) and one that analyzes the gas mixture at the return of the distribution module (at USA15). The SGX1 analysis schematic is shown in Figure 4.3 and the USA15 analysis schematic is shown in Figure 4.4.

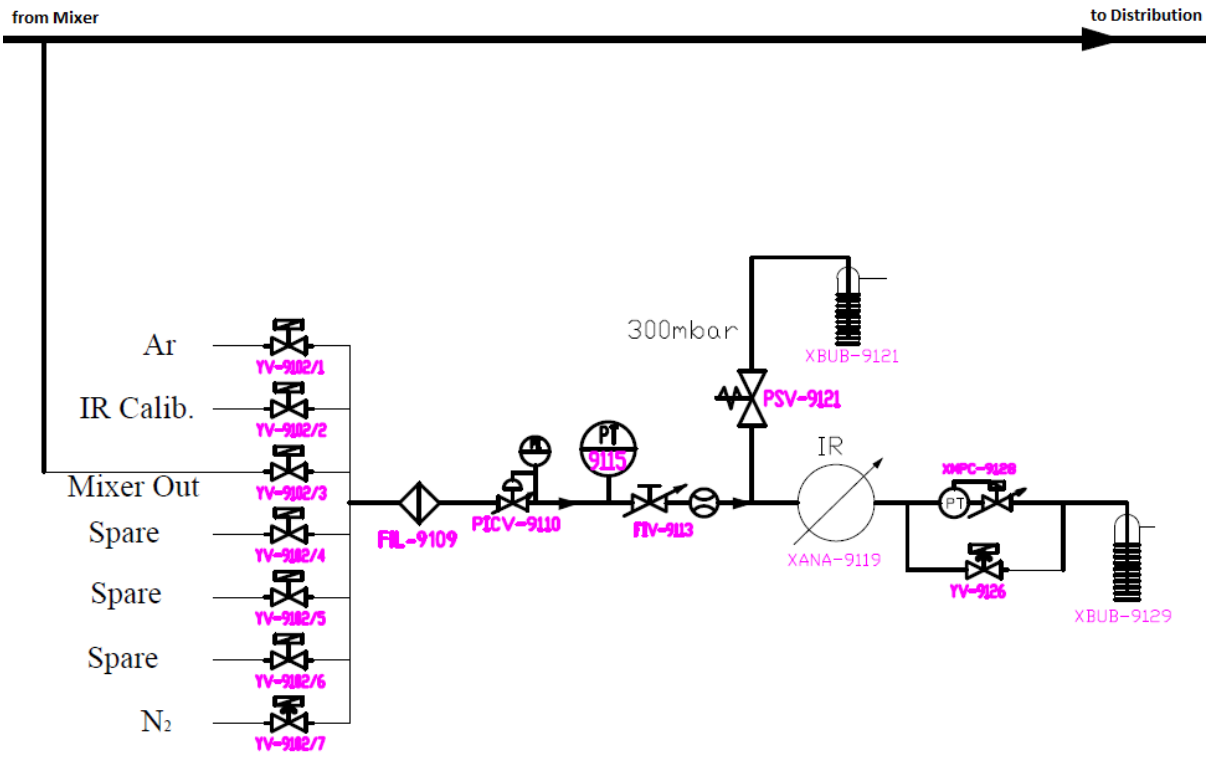


Figure 4.3: Analysis module at SGX1 connected to the mixer output.

The above schematic shows the analysis module at SGX1 which works with an infrared gas analyzer [97]. The mixed gas goes through a filter, a pneumatic control valve together with a pressure indicator and transmitter. The analysis module uses safety bubblers at the outputs for pressure regulation.

In addition, the analysis module in USA15 can be seen below (Figure 4.4). After the gas mixture is flushed through the surface building to USA15, one pipe sends the supply gas to the USA15 analysis module. Another pipe keeps flushing the gas into the distribution module

where it is finally propagated through every detector chamber. Two racks feed each NSW side with the mixture. The gas is then flushed back from UX B to USA15 again where the return pipes from each rack meet the analysis module. Again, using similar modules the and an IR Analyser the supply gas is compared to the gas returning from the experimental area.

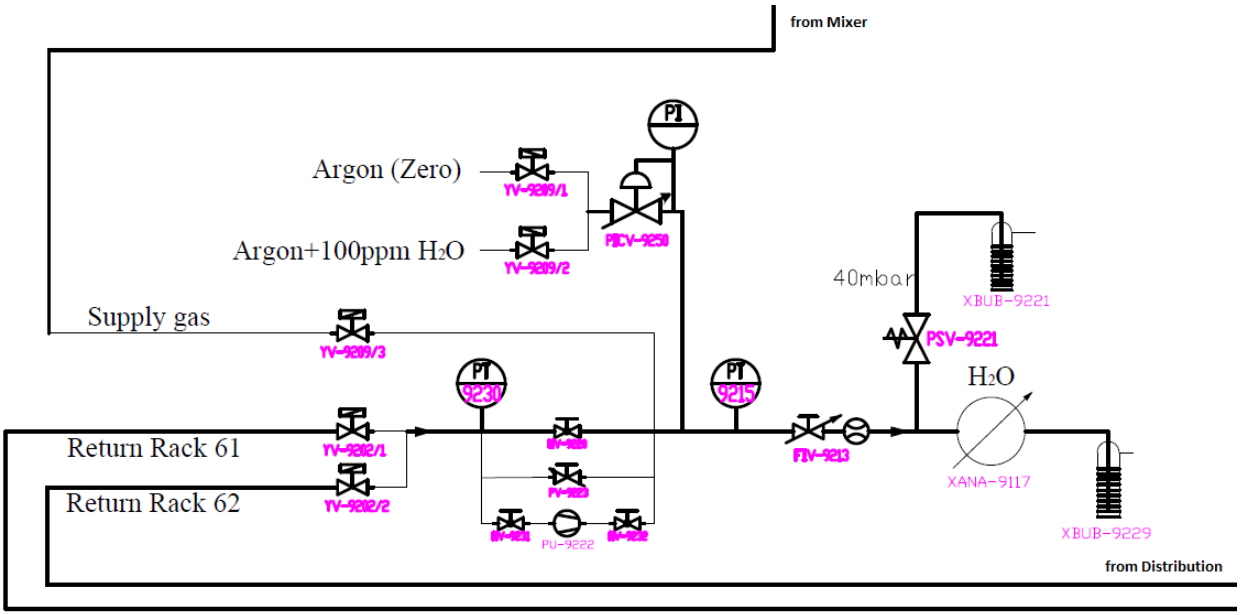


Figure 4.4: The analysis module USA15

4.1.3 The Distribution Module

The distribution module of NSW has to provide gas in 32 detectors in total (16 for each side). The final configuration will use the main gas racks already existing for for MDTs and CSCs which were used in the old Small Wheel. These racks provide 8 gas channels each and therefore each channel has to serve two NSW wedges. However, to retain redundancy, these channels will not serve the same sector. Specifically:

- IP-side Large Wedges are connected to channels 1-4
- HO-side Large Wedges are connected to channels 9-12
- IP-side Small Wedges are connected to channels 5-8
- HO-side Small Wedges are connected to channels 13-16

This design is shown in Figure 4.5. For each wedge it was decided to install a mass flow sensor in order to monitor the gas flow balance during NSW operation. During the operation of the gas system, any quantitative discrepancy between the nominal flow rate and the measured one in the return line of each gas channel (negative balance) should indicate a leak.

Another functional parameter for the gas system configuration is the average static pressure inside the layers which has to be minimized. In the sectors of lower positions the minimum achievable pressure is the pressure due to the gravity of the gas and is in the order of 0.4 mbar. The pressure in a layer is considered to be the average between the inlet and outlet of the detector.

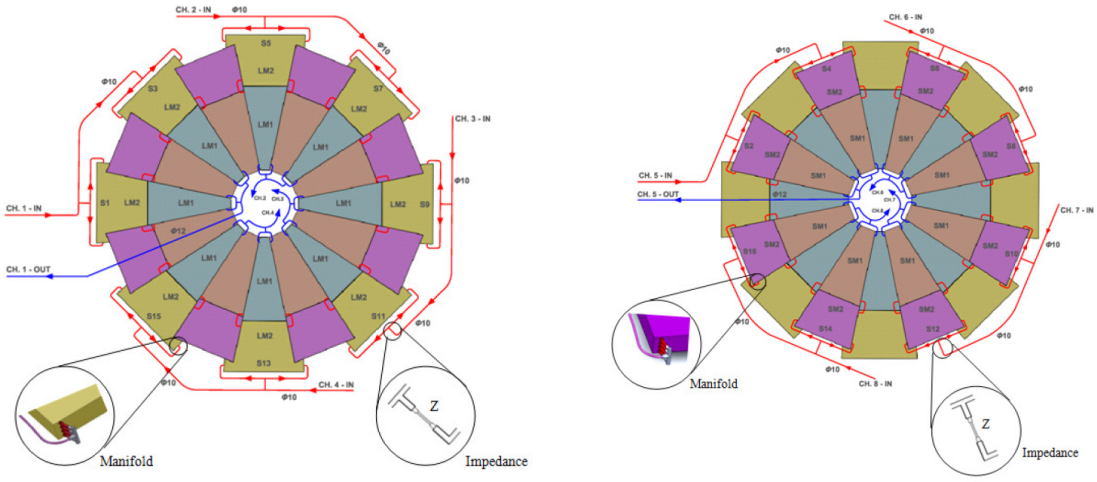


Figure 4.5: Gas distribution system configuration for NSW MicroMegas Large Wedges (left) and Small Wedges (right)

The Micromegas gas system is required to renew the gas mixture 10 times per day. Nevertheless, it has been modified to renew the gas 4 times per day. This leads to a definite gas flow rate for each detector.

The final system most crucial parameters are understandably the gas flow rate and pressure in inlets and outlets of the Micromegas detector layers. Determination of the pressure/flow rate relationships was done with theoretical models and individual simulations using the COMSOL Multiphysics and ANSYS CFX. The actual routing and length of the pipes were also introduced according to simulation results from Pipe Flow Expert [98] [99].

With the efforts of the National Technical University of Athens multiple operations to test this system were completed:

- The impedance design consisting of simulations, production of samples and testing for fewer gas renewals.
- The design of the gas manifolds.
- Simulation studies of the gas insertion together with the design of a buffer zone, pressure drop studies and performance of the gas flow.

4.2 Monitoring of ATLAS gas systems

4.2.1 The Data Interchange Protocol

Data acquisition for all gas systems in ATLAS experiment is performed by the ATLAS Gas Group. All sensor data acquired by the gas group is published via the CERN Data Interchange Protocol (DIP) [100] in the Information Server. DIP is a communication system that allows small amounts of soft real-time data to be exchanged between very loosely coupled heterogeneous systems. These systems do not need to be low latency and the data is mostly summarised rather than low-level parameters from the system. A DIP publication typically contains:

- Data in a key-value map with basic data types and their array variants
- A timestamp
- A quality flag indicating the confidence the data publisher places

- An optional quality string, giving more details about the reason for lack of confidence in the publication

DIP is currently only supported on Windows 64-bit and Linux 64-bit systems. The DIP service is composed of:

- A Central Name Server that provides the list of available publications.
- An API (Application Programming Interface) that allows to publish and receive information.
- A WinCC OA extension API that allows to publish and receive DIP data in WinCC OA.
- A LabVIEW extension that allows to publish and receive DIP data in LabVIEW

It is exactly this communication protocol that allows for data to be accessed by WinCC OA. Typically, low level data acquisition is implemented using PLCs and data is finally published in WinCC OA datapoints.

Data Quality DIP defines specific quality attributes which flag specific bits at the published datapoints. For each datapoint, the flags shown in Table 4.1 have been defined:

Quality Flag	Description
`<dp_name>.quality.invalid`	Indicates that all values of this datapoint are invalid either because of the server data or because of communication failure
`<dp_name>.quality.uncertain`	Indicates that the server has flagged the data as *uncertain*
`<dp_name>.quality.bad`	Indicates that the server has flagged the data as *bad*

Table 4.1: DIP Quality flags

4.2.2 The `fwAtlasGas` component

ATLAS used to have 5 active gas systems, four of which are dedicated to the muon subdetectors CSC, MDT, RPC and TGC and the fifth to the Transition Radiation Trackers (TRT) [102]. The new, sixth gas system that will be added to this list is the MMG (Micromegas) gas system.

The Gas Control System (GCS) is a common software package developed among CERN groups for the control of all gas systems of the LHC experiments (23 in total). Hardware devices are controlled by a set of PLCs while at the same time, a supervisor WinCC OA project provides a user interface with configurations.

Although the GCS is a separate WinCC OA project, there is no direct connection between the DCS projects of ATLAS and the GCS. All gas system parameters of a subdetector is published by the GCS via the DIP protocol and subscription to the data is handled in common by the ATLAS Information Server (IS) projects. The subdetector projects can access this data directly from the IS.

The `fwAtlasGas` component aims at facilitating a standard handling and display of gas data of ATLAS subdetectors. The main implementations of this component are:

- Functions aimed at datapoint and datapoint type creation
- Functions for datapoint attributes configuration (archiving, alerts, ...)

- A generic copy mechanism (script) for retrieving data from the Information Server
- FSM types for gas devices and logic objects plus functions for creating FSM trees for each subdetector
- Widgets for easy building of gas system schematic views
- A set of example panels which can be used "out of the box" for some of the FSM nodes
- Widgets and tools for easy alert configuration panels for people unfamiliar with WinCC OA

The abovementioned functions of the `fwAtlasGas` need to be extended to apply for MMG Gas as well. Detector specific edits in the libraries and scripts to comply with the Micromegas technology will be thoroughly explained in the next section.

4.3 The Micromegas Gas System Monitoring

4.3.1 Project datapoints and the copy mechanism

As it has already been mentioned, monitoring for the Micromegas gas system happens at low level by the Gas Control Group of ATLAS. Data is stored in `ATLGCSIS2` which stands for "ATLAS Gas Control System Information Server 2". Every possible parameter monitored by the gas system is a datapoint in IS2 and is stored either as `float`, `int` or `bool`. The exact datapoint types are called `DipFloat`, `DipFlag` and `DipInteger`.

The convention used by the Information Server is that datapoints and datapoint elements of the DIP type are displayed as shown in Figures 4.6, 4.7 and 4.8:

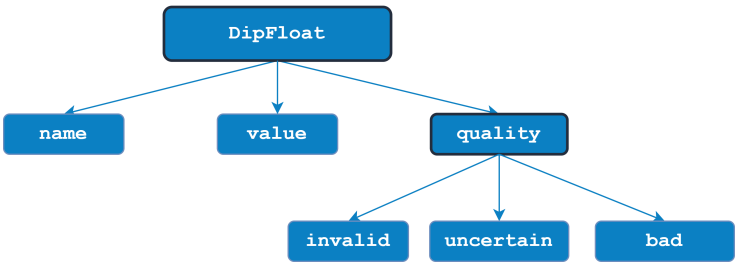


Figure 4.6: `DipFloat` contains the following datapoint elements: `name` (`string`) of the datapoint (unused), `value` (`float`) and the `quality` bits that are all `bool`.

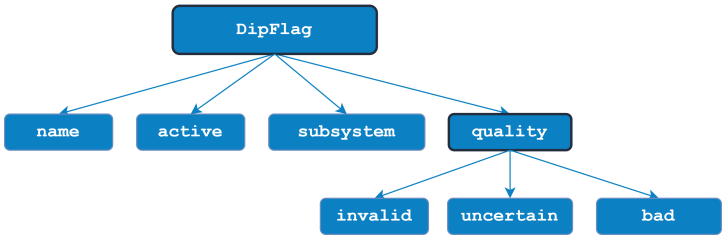


Figure 4.7: `DipFlag` contains the following datapoint elements: `name` of the datapoint (unused), `active` (`bool`) flag value, `subsystem` (`string`) that the flag affects and the `quality` bits that are all `bool`.

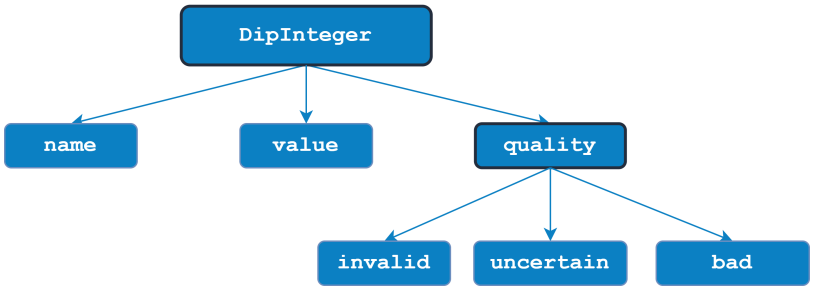


Figure 4.8: `DipInteger` contains the following datapoint elements: `name` (`string`) of the datapoint (unused), `value` (`int`) and the `quality` bits that are all `bool`.

This kind of structure published by the Gas Group aims at listing of all the information in every type available. However, it is not a convenient way to describe the structure of every module in the gas system. For this reason, the structure of the final datapoints used by the monitoring system comply to the `fwAtlasGas` convention which groups data in the datapoint types shown in Table 4.2:

Datapoint Type	Description
`fwAtlasGasChain`	Contains datapoints with information about the analysis chains of the gas system.
`fwAtlasGasChannel`	Contains a datapoint for every gas channel that is being monitored in the gas system.
`fwAtlasGasDistribution`	Contains a datapoint with information about the distribution module.
`fwAtlasGasGasSystem`	Contains a datapoint with information about the whole gas system.
`fwAtlasGasMixer`	Contains a datapoint that monitors all parameters of the gas mixer module.
`fwAtlasGasPlcCounter`	Contains a datapoint with information about the counter of the Micromegas PLC.
`fwAtlasGasRack`	Contains datapoints with information about the gas racks of the Micromegas distribution.
`fwAtlasGasSource`	Contains datapoints with information about the gas sources used for each analysis chain of the distribution.
`fwAtlasGasProjectMonitor`	Contains a datapoint with general information about the monitoring of the gas system.

Table 4.2: Datapoint Types used in the Micromegas Gas System monitoring.

In order to facilitate all these infrastructure datapoints for the gas system to function, the `fwAtlasGas` component uses a panel called `InteractiveSetup.pnl` that helps the developer lay the grounds for the control system. This panel uses a set of buttons to create, update, set user bits, configure, set alerts, start the Finite State Machine tree, set units and archiving for all datapoints used by the gas project. To do this, it uses specific functions implemented in the libraries `fwAtlasGas`, `fwAtlasGasEvent`, `fwAtlasGasFsm` and `fwAtlasGasService`.

Before using the `InteractiveSetup.pnl`, first, some groundwork has to be completed since the Micromegas detector is a new technology that was not yet featured in the `fwAtlasGas` component. Specifically, two changes:

1. the `MMG` technology has to be included in all functions of the library `fwAtlasGas.ctl`
2. the DIP publication format of `MMG` technology has to be adapted in all mapping functions of `fwAtlasGas.ctl`

These changes were completed, tested and later communicated with CERN Central DCS team to include in the new version of the component. Figure 4.9 shows the panel used to produce the datapoints:

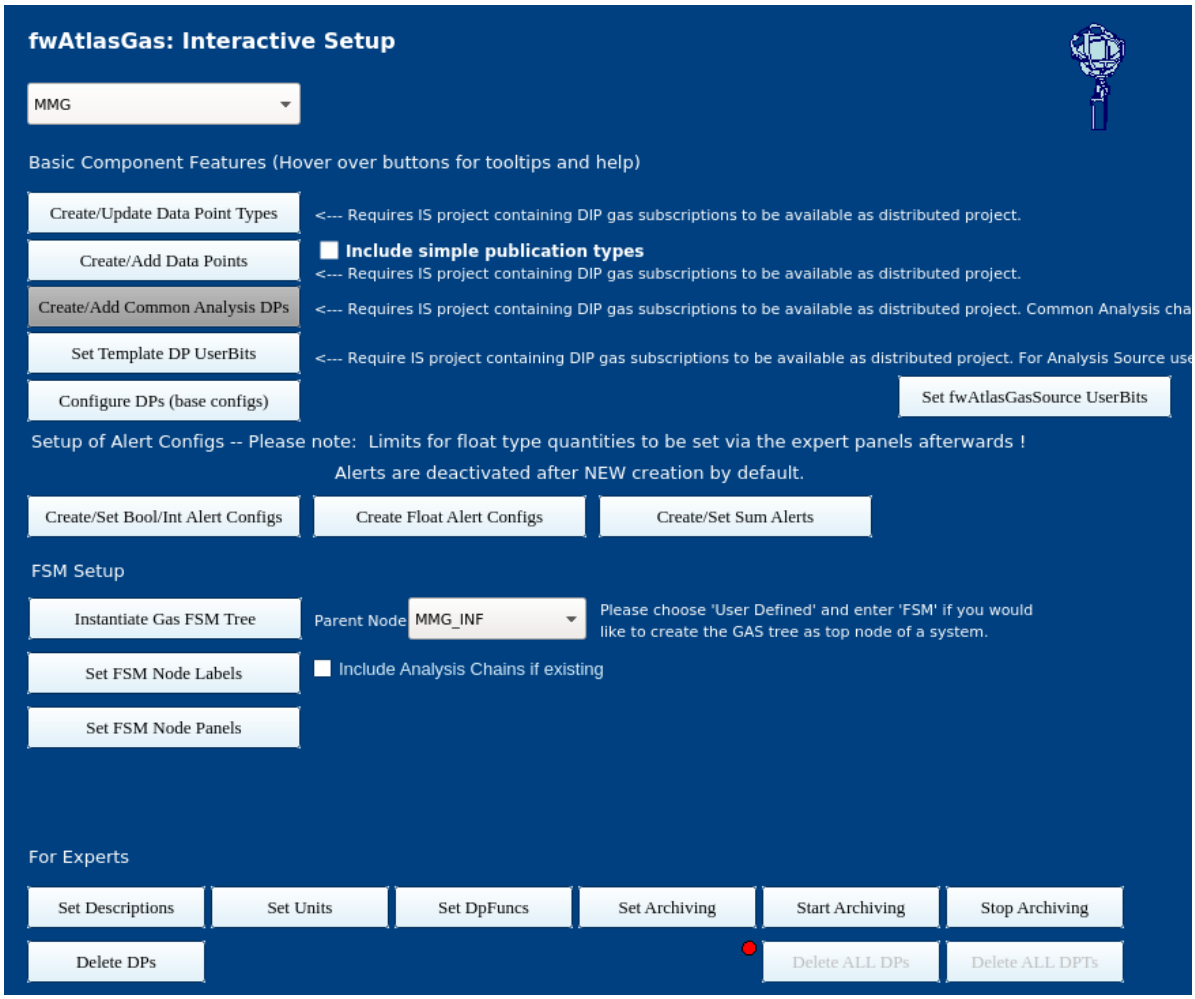


Figure 4.9: The WinCC OA panel (modified for `MMG`) used to produce the datapoints together with all configurations necessary.

A separate change introduced in this implementation is the creation of the Micromegas gas FSM tree. It is not used in the framework yet but has been developed for the project in the library `mmgGas\mmgGas_FSMHandling.ctl` with the function `mmgGas_createFsmTree()`.

Every datapoint element monitored by this project (or the ones copied from `ATLGCSIS2`) can be found with detail in Appendix A. For now, the focus will be on the difference between the datapoint types in the Information Server and the ones used by `fwAtlasGas`. In the Information Server, all datapoints are under their data type Dip specific datapoint type which gives any information about mapping in the name itself. For example, in Micromegas gas system the Input Flow of gas channel 1 in Rack 61 is monitored in the information server by the datapoint `gas_MMG_Distribution_Rack61_Channel1_InFlow_FE6102CH1.value` under datapoint type `DipFloat`. The same parameter is mapped in the gas project in the datapoint `gas_MMG_Distribution_Rack61_Channel1.InFlow` .

The copy mechanism The transition between the datapoints of the Information Server and the project is achieved through the libraries mentioned before. They implement functions that are utilized by the script `fwAtlasGas_copyGasData.ctl`, which is implemented by the Framework.

This algorithm essentially runs once, by using the `dpConnect` function of WinCC OA to track changes on the information server variables and fetch them to the Micromegas Gas System project. The two changes discussed earlier on this subsection for the `MMG` format have already been introduced in the component and thus the script can use the framework libraries to successfully connect the Information Server datapoints with the Micromegas gas project as well.

WinCC OA uses this script in the form of a manager. The monitoring project only needs to include a control manager in the console so it can run in the background indefinitely (or until stopped).

4.3.2 Monitoring infrastructure for ATLAS FSM

In principle, most operations required in the development of this monitoring project are already implemented by the `fwAtlasGas`. Most of the work is based setting the infrastructure and developing tools to better display and monitor the specifics of the Micromegas technology.

Thus, in order to bring off some extra required functionalities, five extra libraries have been developed:

- `mmgGas_AccessControl.ctl`: Implements functions required for Access Control in the FSM.
- `mmgGas_constants.ctl`: Lists and returns all constants of the project.
- `mmgGas_conversions.ctl`: Implements functions that make conversions useful for the project.
- `mmgGas_FSMHandling.ctl`: Implements all FSM related (Micromegas specific) operations.
- `mmgGas_infrastructure.ctl`: Implements infrastructure functions like datapoint building or configurations.

Final operation of the FSM is completed after every panel on each node is designed. The framework already provides very useful tools and objects that speed up the development for the "channel" and "rack" views. However, the rest of the panels have to be developed from scratch as layout of the physical system differ from case to case.

At the same time, the look and feel of the user interface must conform to the already existing monitoring user interfaces of ATLAS gas systems. This is crucial since operators are already familiar with the existing gas systems and their operation. For this reason, the following objects were designed for use:

- The configuration panel `mmgGasConfiguration_complex.xml` under `mmgGas_service`.
- Several objects under `objects/mmgGas` that implement gas alerts, gas channel view and NSW sectors together with their statuses and states. Most physical components of the layout are also designed in this folder.
- The main panels of the MMG FSM under `fwAtlasMainPanels/mmgGas` that include: `MMG_GAS_CHANNEL.pnl`, `MMG_GAS_DISTRIBUTION.pnl`, `MMG_GAS_MIXER.pnl`, `MMG_GAS_RACK.pnl` and `MMG_GAS_SYSTEM.pnl`

- The secondary panels of the MMG FSM under `fwAtlasSecondaryPanels/mmgGas` that include: `MMG_GAS_CHANNEL_info.pnl`, `MMG_GAS_DISTRIBUTION_info.pnl`, `MMG_GAS_MIXER_info.pnl`, `MMG_GAS_RACK_info.pnl` and `MMG_GAS_SYSTEM_info.pnl`

In order to build most value fields of the above objects and panels, some framework objects were used (or recycled). This is a typical convention for all ATLAS FSMs used in the DCS and must be followed by all subsystems:

- `mdtRackChannel.pnl` and `mdtRackControl.pnl` were recycled for Rack visuals.
- Objects from `objects/fwAtlasGas/alertHdl` were recycled for alert visuals.
- Objects from `objects/fwAtlasGas/symbols` were used for piping flow diagrams.
- Objects `parameter.pnl` and `parameter_small.pnl` were used to display FSM textfields that comply with the ATLAS FSM styling.

4.3.3 The Micromegas gas system FSM

Before diving into the monitoring and the interface, Figure 4.10 shows the FSM Hierarchy of Micromegas gas system:

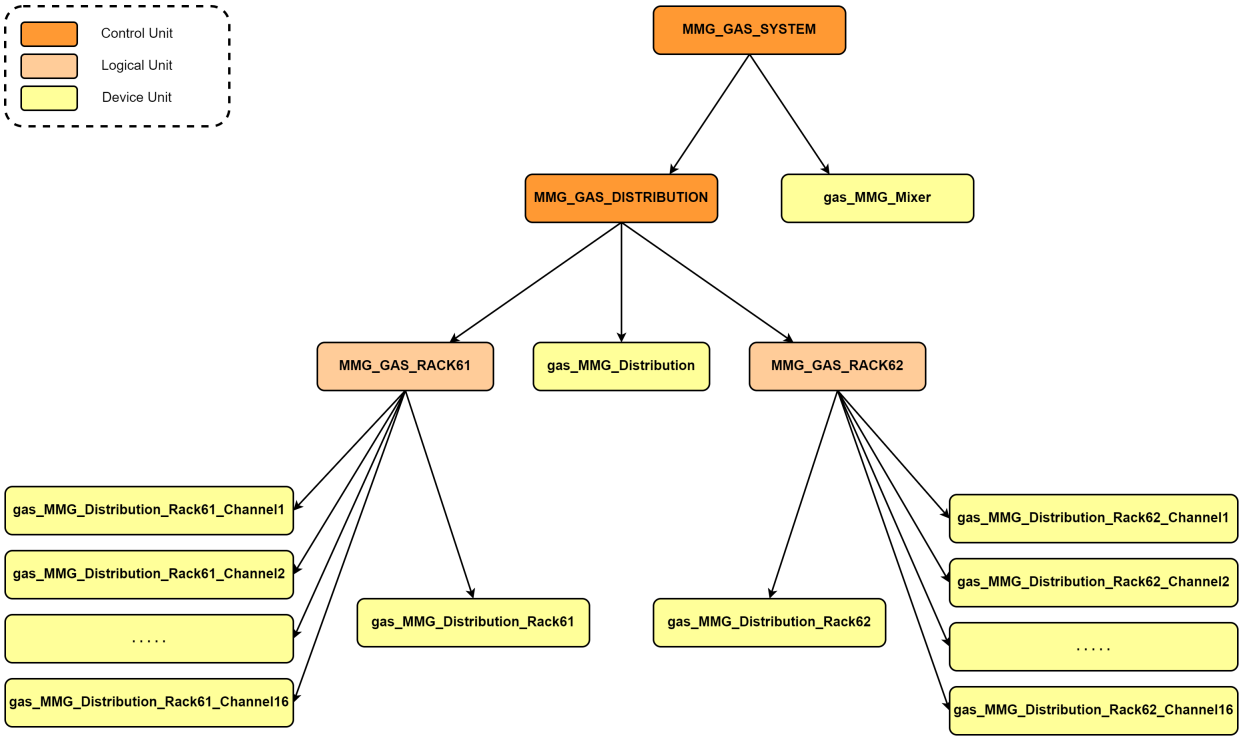


Figure 4.10: The FSM tree of Micromegas gas monitoring.

Each gas channel (Device Unit) is at the lowest point of the FSM hierarchy and it host all readout parameters: `InFlow`, `OutFlow` and `GasLoss`. These values influence both status and state of the gas channel which are propagated upwards to their parent. This influences the status and states of the node above the parent and so on. This pyramid like scheme eventually calculates a final state for the Control Unit of Micromegas Gas which is necessary for ATLAS experiment to go in the state `RUN_READY`.

Note in the above figure how each Control/Logical Unit also parents itself as a Device Unit. This is important so that these units can calculate their own state and status based on their readings as well.

Since the reader is already familiar with the ATLAS DCS from the previous chapter, Figure 4.11 displays the Detector Control System of all muon systems in the ATLAS experiment. This involves MDT, RPC, TGC and now MMG and STG technologies for NSW.

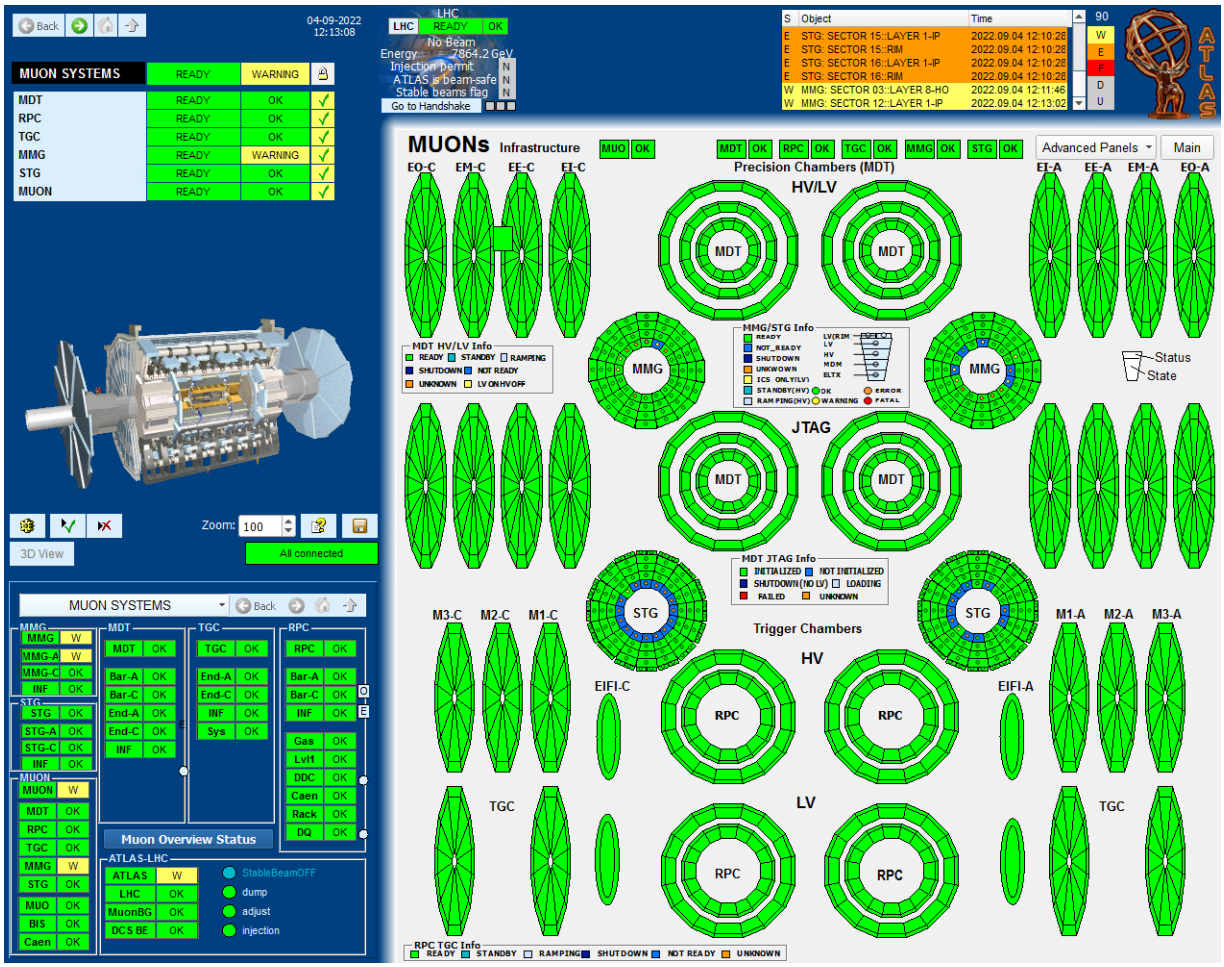


Figure 4.11: The Detector Control System for all muon Finite State Machines. States and statuses are shown in the panel for every detector that is being monitored in the experiment.

Before viewing the gas interface entirely, it is important to grasp how the states of each node are calculated. The reader is advised to first take a look at Appendix B in order to understand what each state and status in the FSM represents. Generally, these states and statuses are commonly used by all gas experiments and identical conventions were followed for all similar modules and systems. The desired operational state of the Micromegas Gas System must be `READY` and with the `OK` status in order for the experiment to begin.

Navigating at `\MMG\INFRASTRUCTURE\GAS` brings the DCS to the Micromegas Gas monitoring, which is shown in Figure 4.12:

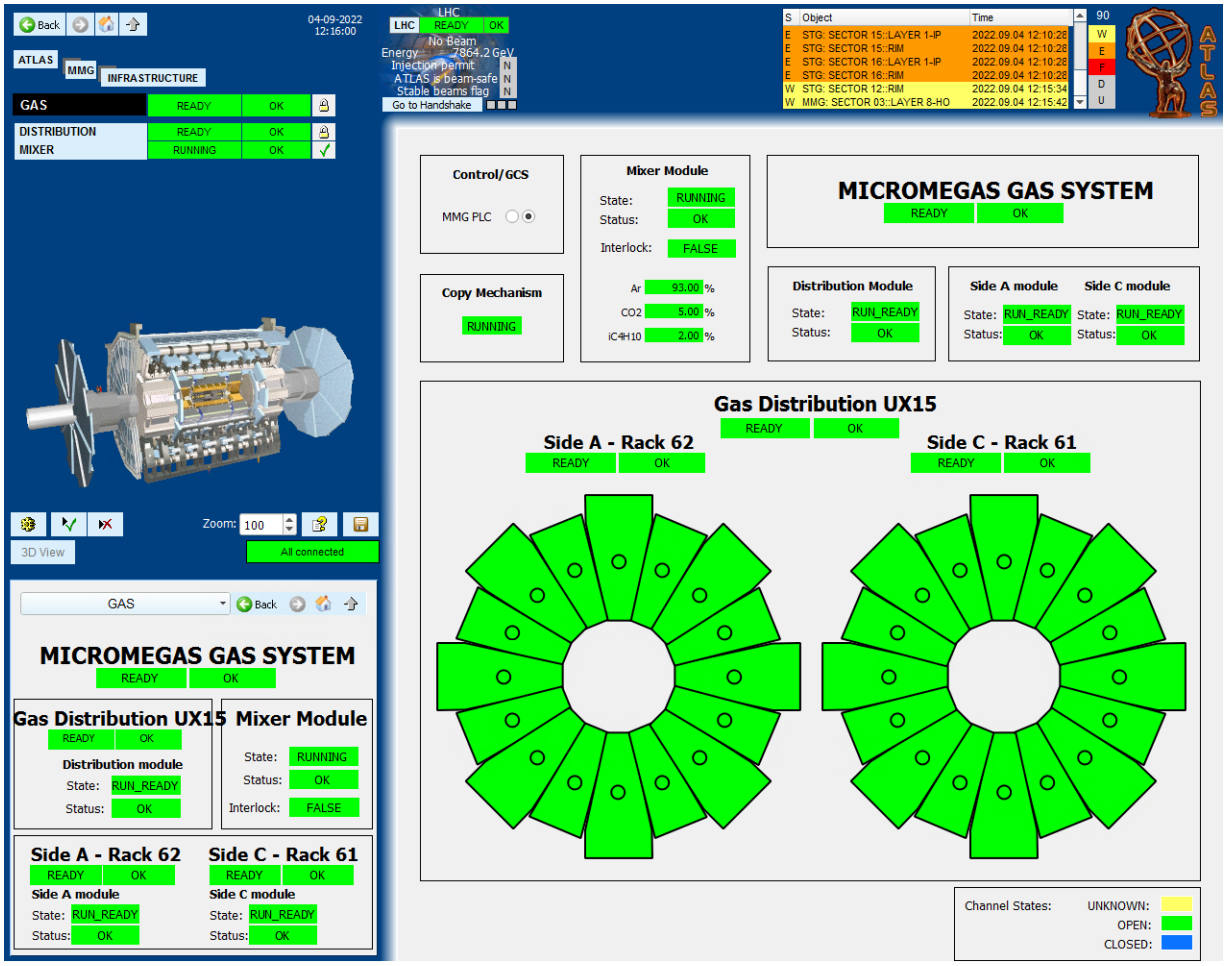


Figure 4.12: The Micromegas gas system DCS.

The figure above shows the gas system panels, which are designed to specifically monitor:

- General information about the mixer, the distribution and both racks together with their status and state. On the left is the state and on the right the status.
- A schematic of both sides and all channels with states and status visible. For the physical detectors represented as wheels, the background color displays the state and the dot in the middle of the detector represents the status.
- The Micromegas PLC alive status and whether or not the copy mechanism is running.

On the main panel, several monitoring actions are available for the operator, other than using the FSM tree on the left:

- Mixer module object: left clicking navigates the main panel to the mixer monitoring, right clicking navigates the secondary panel to the mixer information.
- Distribution module object: left clicking navigates the main panel to the distribution module, right clicking navigates the secondary panel to the distribution information.
- Rack module objects: left clicking navigates the main panel to the rack overview and monitoring, right clicking navigates the secondary panel to the rack information.
- Physical detector objects: left clicking navigates to the gas channel that is monitored for this detector, right clicking navigates the secondary panel to this specific channel information.

Following up the FSM hierarchy, the next module is the gas mixer. For the main panel, four functions have been developed to switch between the views:

- `draw_mixer_schematics()`
- `draw_mixer_alerts()`
- `remove_mixer_schematics()`
- `remove_mixer_alerts()`

The mixer schematic together with the two trending plots are objects that are added or removed on the main panel. The value fields that are seen are represented by a color which displays their alert status. The mixer monitoring can be seen in Figure 4.13.

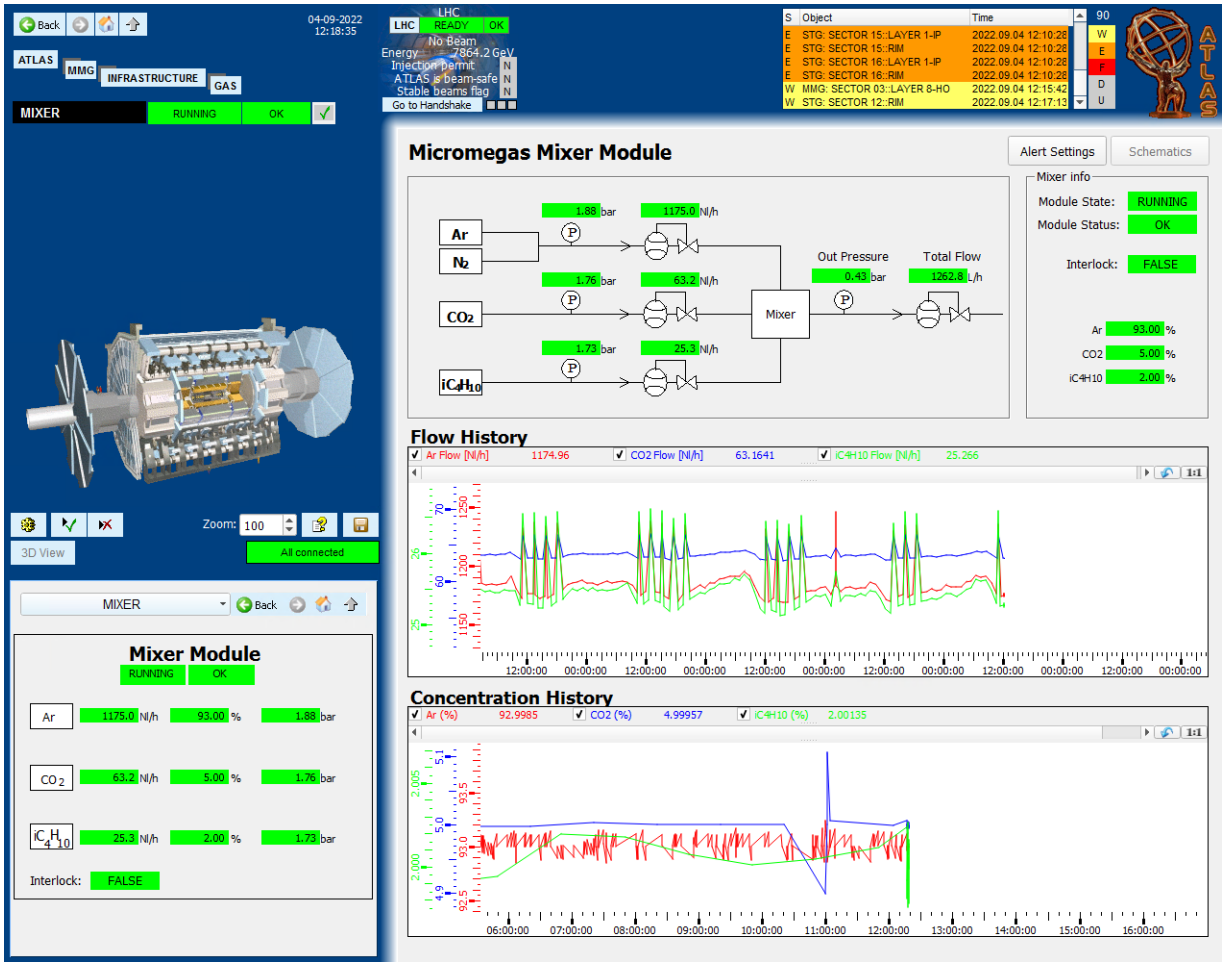


Figure 4.13: The Micromegas gas mixer module monitoring.

The mixer panel is designed to monitor:

- All logical information about the mixer: state, status and interlocking
- The composition ratio of each gas as a percentage
- All three lines of gas supply to the mixing module. For each line, we monitor the input pressure in bar and gas flow in N L/h.

This main panel is mostly used for monitoring but the operator can act on it with the two following ways:

- Making use of the active archiving, the operator can view trends of historical data both for the mass flow sensors and the concentration ratios of all three gas supply lines.
- Configuration of the alert settings.

Practically, the alerts and statuses of a device unit are connected and thus if for example a value is out of nominal range, the alert will propagate at the state of the mixer module. The alert configuration of the mixer module can be seen below in Figure 4.14:

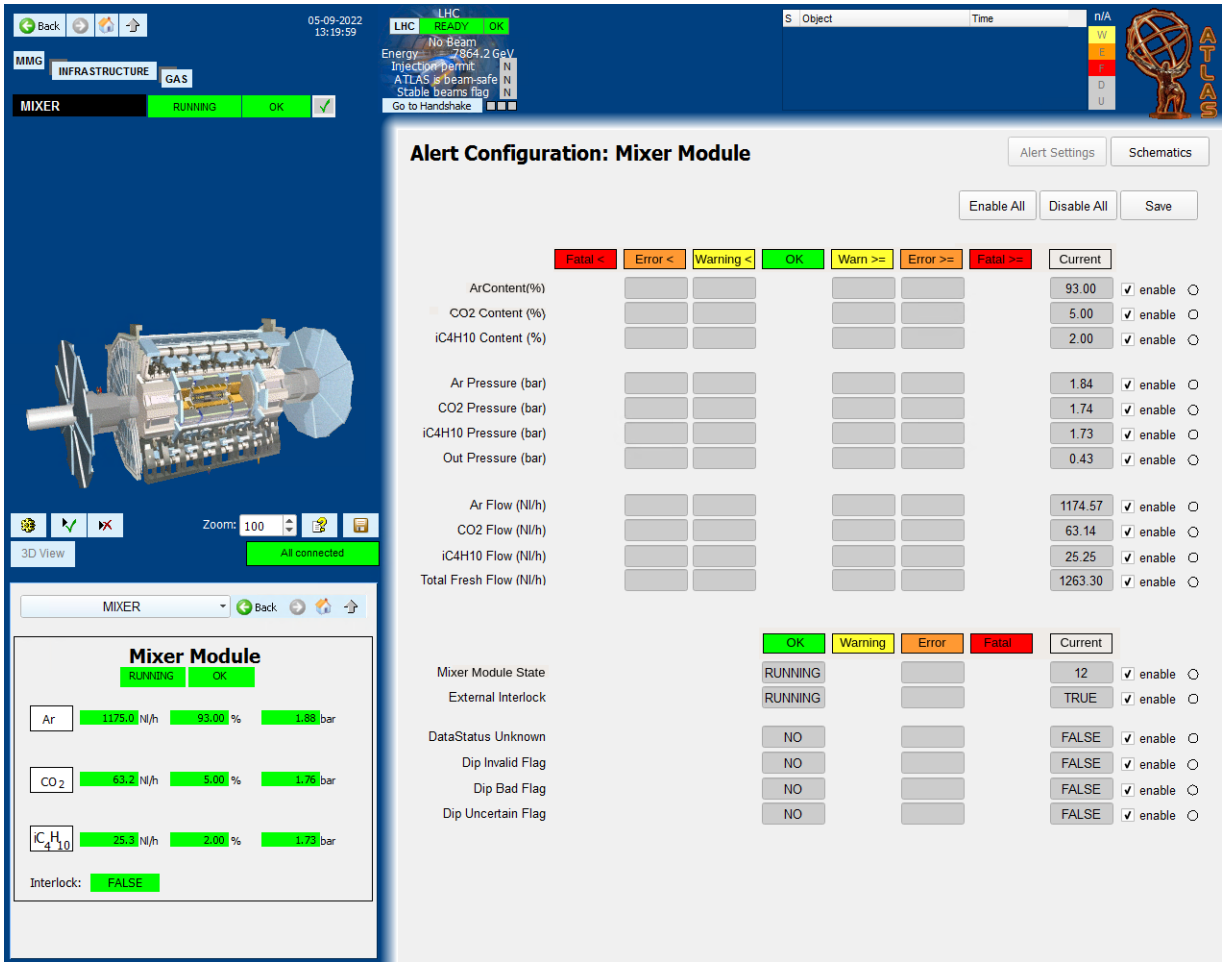


Figure 4.14: The Micromegas gas mixer alert configuration panel.

The alert ranges for all analog values in the gas system (and especially here in the Mixer) have 5 value ranges that are mapped as following:

- `LOW_LOW` which is the `ERROR` lower range status
- `LOW` which is the `WARNING` lower range status
- `OK` which is the `OK` nominal range status
- `HIGH` which is the `WARNING` higher range status
- `HIGH_HIGH` which is the `ERROR` higher range status

The boolean values have only `TRUE` or `FALSE` values and are thus only mapped to the `OK` or `ERROR` values.

The operator (with required access rights) can fill any value field and by pressing save the range will be applied to the specific alert. The operator can also enable or disable alerts from the tickbox and press save. The schematics button takes the operator back to the monitoring panel.

Next in the hierarchy is the distribution module. This panel is designed similarly with the mixer panel. It features a main function `draw_nsw_sides()` which shows the physical layout of both wheels. It is a shorter and more compact version of the gas system panel. The distribution monitoring panel can be seen in Figure 4.15:

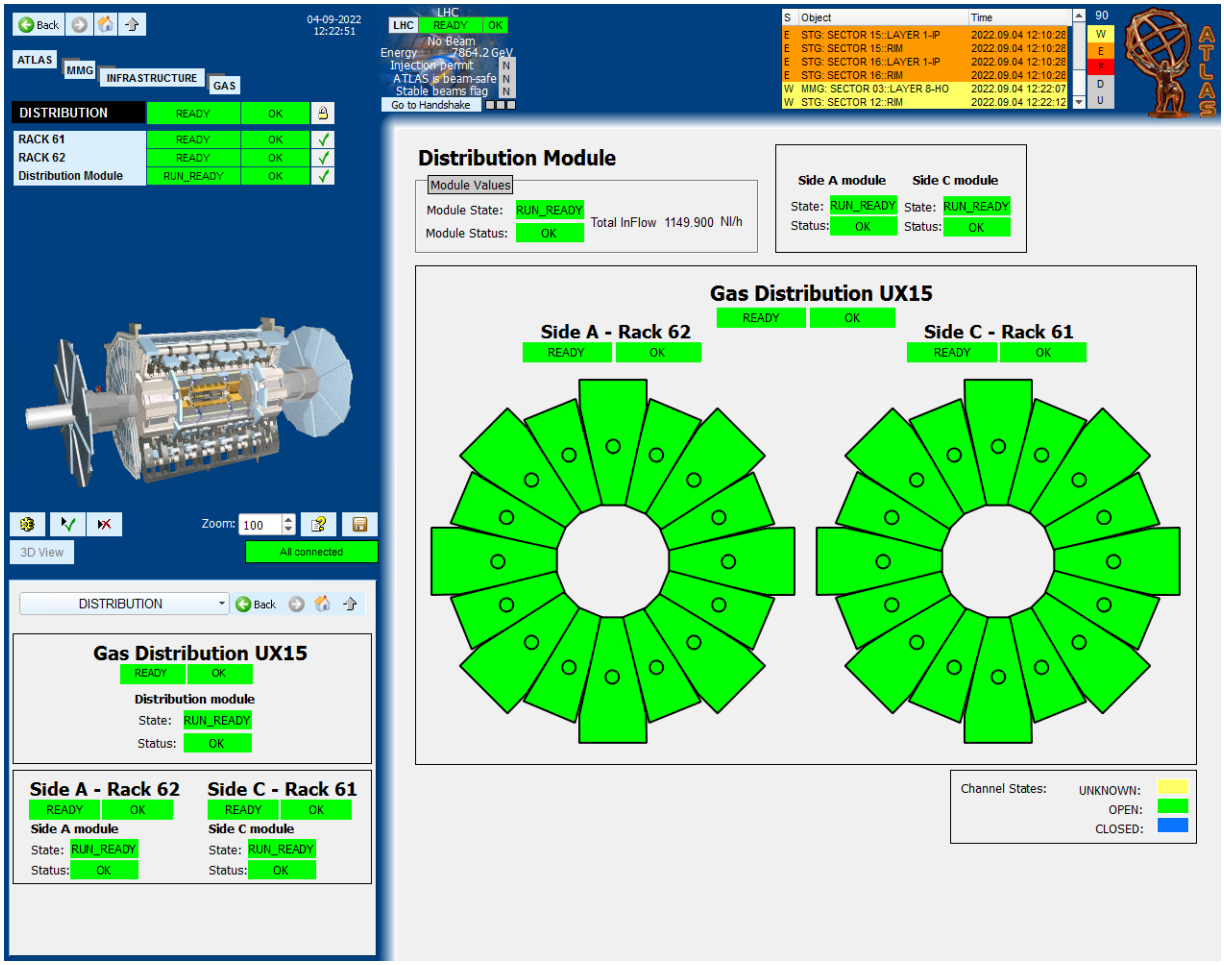


Figure 4.15: The Micromegas gas distribution module monitoring.

The distribution monitoring panel is designed to present to the operator the following information:

- The status and states of the distribution module and both racks.
- Similarly to the gas system panel, all statuses and states of the channels are available with the physical layout of the detectors.
- The `Total InFlow` reading of the distribution module in N L/h which is for both sides.

The operator can navigate lower in the FSM hierarchy by clicking on the graphical objects:

- Left clicking on a rack module navigates the main panel to the rack overview and monitoring, right clicking navigates the secondary panel to the rack information.

- Left clicking a detector on the layout navigates to the gas channel that is monitored for this detector, right clicking navigates the secondary panel to this specific channel information.

From both the distribution panel and the gas system, the rack monitoring panel can be accessed. Similar to logic in other panels, again, four functions are implemented for the schematics:

- `ShowSynopticView()`
- `ShowAlertConfig()`
- `RemoveSynopticView()`
- `RemoveAlertConfig()`

The rack initializes with all gas channels with their schematics and readings. It can be seen in Figure 4.16:

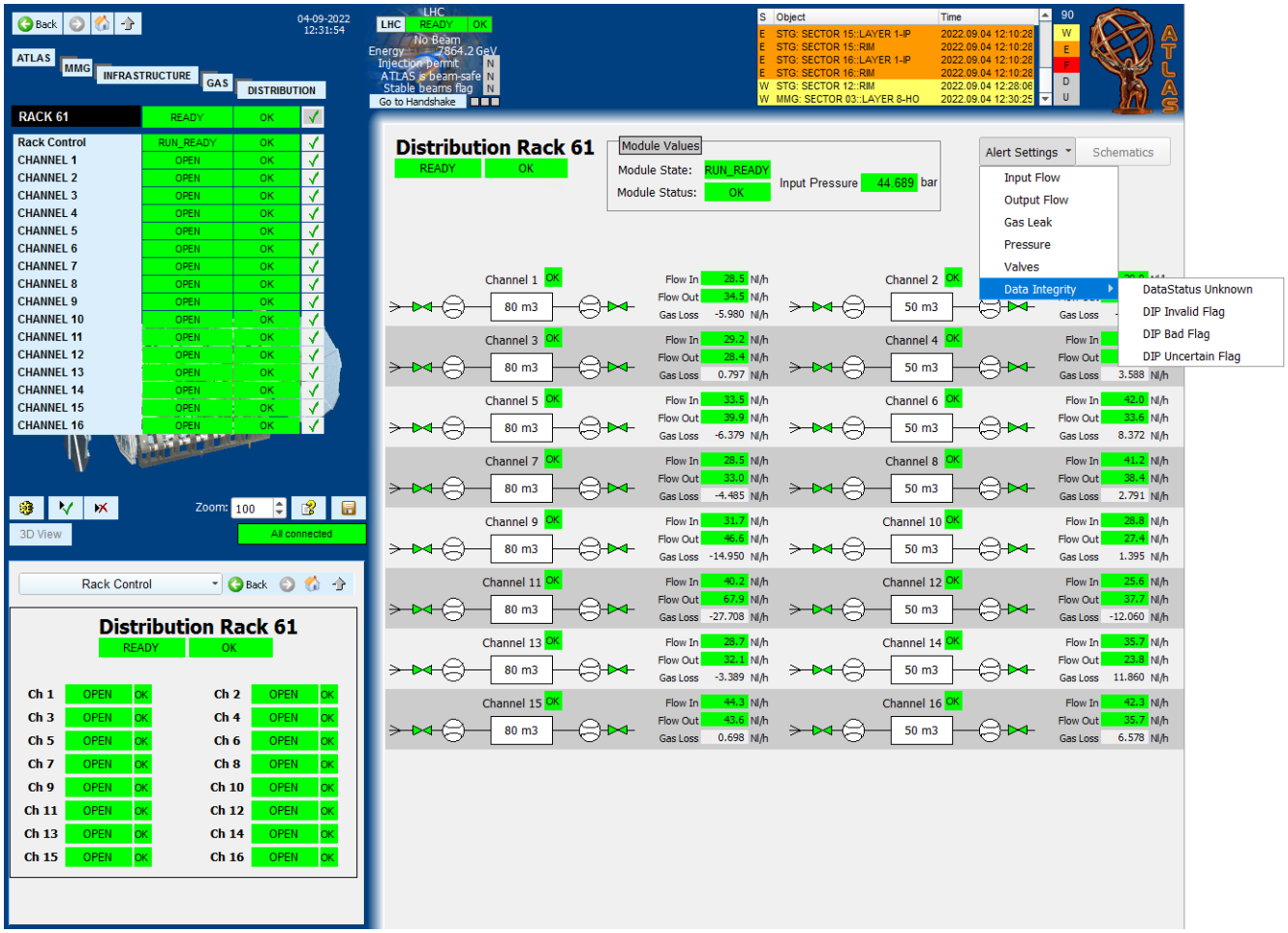


Figure 4.16: The Micromegas gas rack monitoring panel.

The rack panel displays the following crucial for the experiment data:

- Input and output flow of each gas channel in N L/h.
- The gas loss in each Micromegas detector.
- The status and state for each gas channel.

- The gas rack input pressure in bar.

From this panel, the following actions are available:

- Left click on a detector and navigates to any gas channel in this rack, right click to navigates the secondary panel to this channel's information.
- Select an alert setting to configure from the drop down list button.

From the dropdown list of alerts, the operator can select which alert to configure. An example is shown below in Figure 4.17 with the detector gas loss:

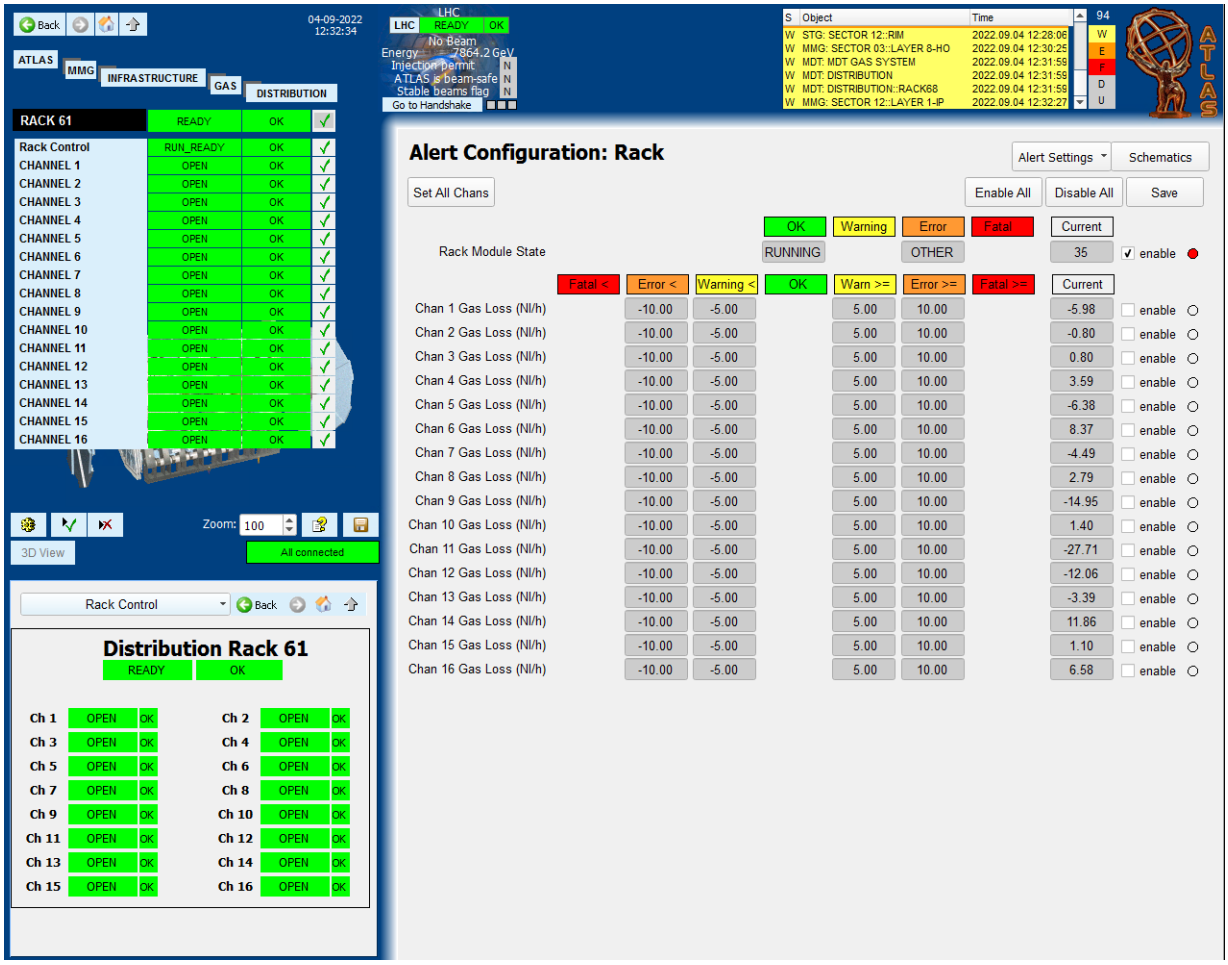


Figure 4.17: The Micromegas gas rack alert configuration for the gas loss.

This alert configuration is the most crucial in the DCS because the alerts of the gas channels directly influence the status of the device unit associated with the detector. Similarly to the mixer alert configuration, the operator can fill any value field and press save to apply changes in the specific alert. Here all gas loss alerts are currently disabled (this will be explained in the last section). The other alerts that the operator can configure can be seen in 4.16 and include:

- `Input Flow`: Input Flow of the gas channel.
- `Output Flow`: Output Flow of the gas channel.
- `Pressure`: The channel pressure (unused by the Micromegas) and the rack Input Pressure.
- `Valves`: The Valve alerts for each channel (unused by the Micromegas).

- `Data Integrity`: The DIP flags for each channel data.

Finally, the last available node in the FSM is that of a gas channel. In this panel the design simply includes the gas channel schematic together with two trending plots:

- The Flow History showing both input and output flow of the detector.
- The Leak history showing the gas leak of this detector.

The gas channel FSM node can be seen in Figure 4.18:

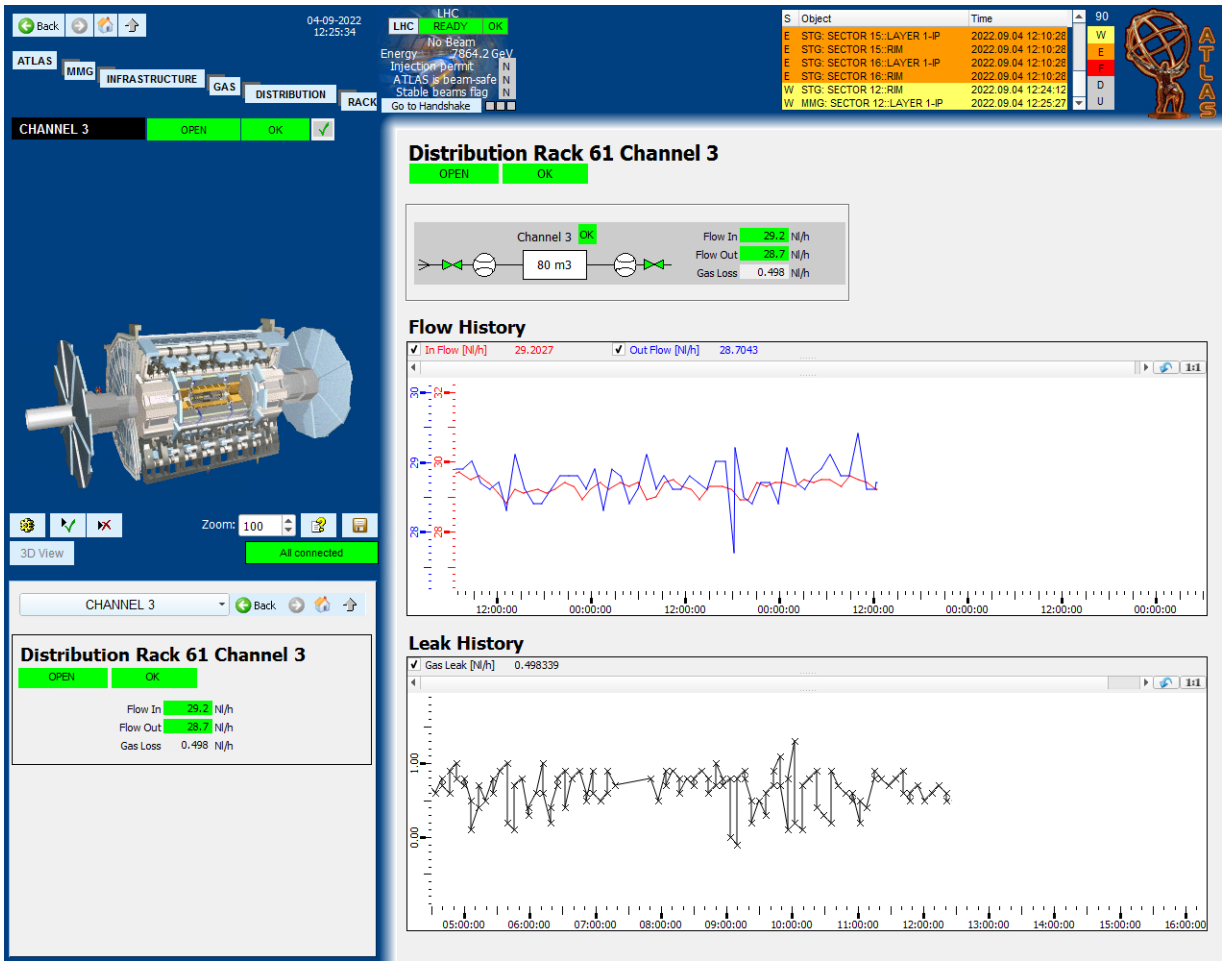


Figure 4.18: The Micromegas gas channel monitoring.

The gas channel is the lowest level of readout in the gas system and thus concludes the monitoring of the Micromegas gas system.

4.4 Maintenance and status of the Micromegas gas system monitoring

All of the code, panels and graphical objects developed for this application is in GitLab in the repository **mmg-gas-nsw-dcs**. In order to install this is ATLAS experiment, the repository is connected to the NSW PC and everything is pulled from the repository to the local folder.

The FSM nodes shown above are the final monitoring panels for ATLAS experiment. However, there is still a little bit of work that needs to be finalized for the monitoring to be finalized:

- The gas loss alerts have been disabled because the gas loss measurement for each channel needs to be calibrated to the actual gas loss of the detector. After calibration, a new datapoint function must be applied to the gas loss datapoint to see the actual gas loss and define the value ranges.
- Some minor modifications in two graphical panels i.e. text not showing up correctly.
- Correction of the detector volume in `mmgGas_constants.ctl` which is 26 L for large sectors and 20 L for small sectors.

Chapter 5

Magnetic Field monitoring near New Small Wheel

Due to it's location, NSW is directly interfacing with the ATLAS Toroid Magnet System which provides the magnetic field for the outer shell of muon detectors. The two end-cap toroid magnets are placed directly behind NSW and the barrel toroid magnet system surrounds the calorimeters. Thus, the effect of the magnetic field on NSW is significant and has to be measured.

5.1 Hardware for magnetic field monitoring in NSW

Over the years, ATLAS collaboration developed the Embedded Local Monitor Board (ELMB) module [103] in order to serve various detector control tasks. It is a credit-card sized plug-on board, designed as a general-purpose CAN-bus based building block for various control and monitoring tasks in the LHC experiment. It provides programmable microcontrollers, differential analog inputs and digital I/O lines.

Based on this board, the National Institute of Subatomic Physics - Nikhef developed the MDT-DCS Module (MDM) as the local monitor-and-control platform for the ATLAS MDT detectors. Firmware development is based on a framework provided by the ELMBio application firmware [104]. To monitor the magnetic field near MDT detectors, Nikhef also developed a magnetic field sensor module. NSW will use the same hardware to monitor the magnetic field in Run 3.

5.1.1 The MDT-DCS CANopen Module

As already mentioned, the MDM is based on the ELMB technology implemented to serve control tasks. In the MDT muon subdetector of ATLAS, the MDT-DCS module monitors MDT chamber environmental parameters, i.e. temperature and magnetic field in and around the chamber as well as MDT front-end electronics voltages and temperatures [105]. The MDM has a JTAG interface connected to the Mezzanine boards and Chamber Service Modules (CSM) mounted on the MDT detectors for electronics configuration. In addition, there are 7 digital I/Os for output control and error status handling along the CSM communication.

In total, about 1200 MDT chambers are connected to 96 CAN-buses, monitored and controlled by the appropriate SCADA running on ten PCs [106]. In NSW, 64 MDM modules and 8 CAN-buses that are monitored and controlled by two PCs will be used to meet the monitoring criteria for the experiment. NSW will not use the CSM and JTAG interface for it's front-end electronics configuration. NSW electronics are validated using other methods and software tools. The MDM in NSW will only be used to monitor magnetic field and temperature.

ATLAS Detector Control Systems have chosen CANbus to be the chosen fieldbus to interface I/Os throughout the detector. For this reason, The CANopen protocol has been adopted as the standard communication protocol to be used on the CANbus [107] [108]. The firmware supported by the ELMB inside of the MDM has a range of additional Object Dictionary (OD) entries and configuration options.

Figure 5.1 shows a simplified block diagram and its connections to sensors and front-end electronics.

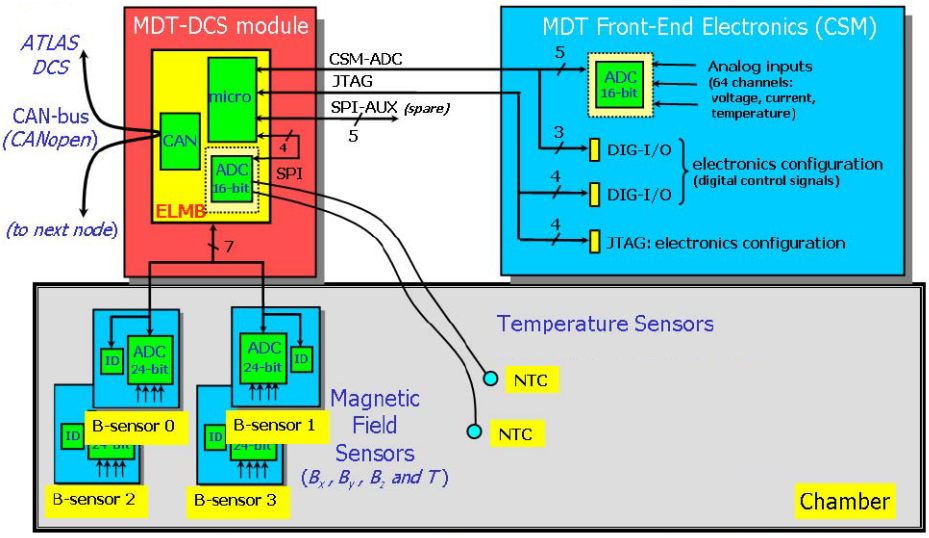


Figure 5.1: Block diagram of the MDT-DCS module with ELMB and its connections. The CSM-ADC and JTAG interfaces are not used to configure the NSW front-end electronics

The actual MDM device with all of its connectors is depicted in Figure 5.2. The T-Sensor inputs can be seen together with the B-Sensor inputs.

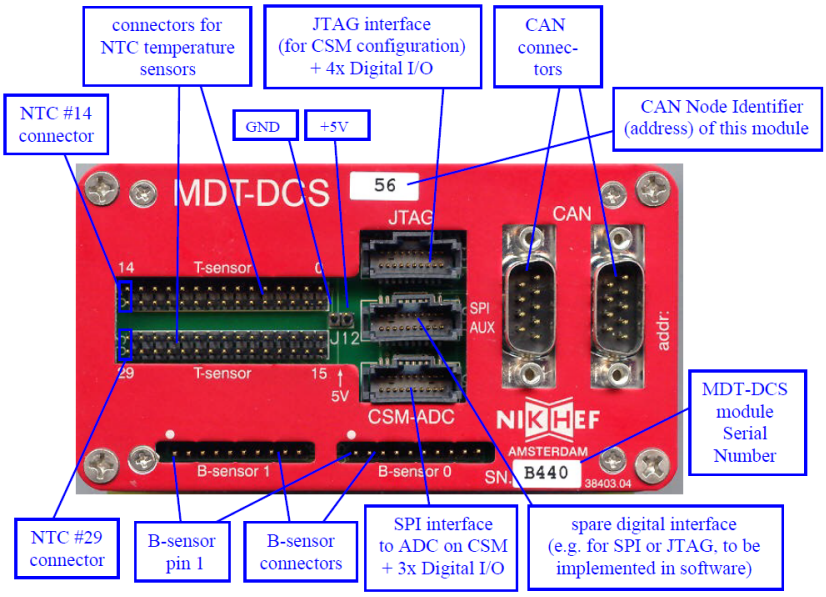


Figure 5.2: All of the MDT-DCS Module front panel connectors and labels

The only difference from the end-configuration is that the new B-Sensors use a different connector than the one displayed here. More details on this topic will be explained in the next subsection.

5.1.2 The magnetic field sensors

The NIKHEF magnetic field sensor module (B-sensor) is a small PCB that is composed of three Hall Effect [Appendix C] sensors and one NTC thermistor to monitor temperature. It also carries an ADC chip for the measurements and an ID-chip. Connection to the MDM is achieved via a 10-pin IDC flat cable [109] connector that utilises the SPI interface [110]. Figure 5.3 shows all of these components. In total, 192 B-sensors will be used in NSW.

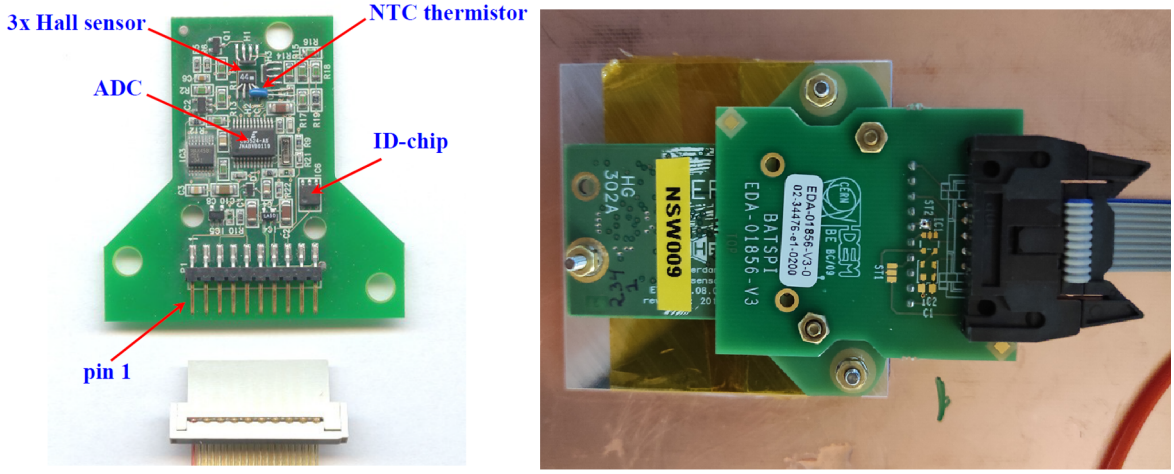


Figure 5.3: The B-sensor module with just the PCB (left) and mounted on a detector (right)

The B-sensor connector has a rather simple layout of 10 pins which is shown in table 5.1. However, due to difficulty of connecting and disconnecting the B-sensor, a mezzanine-like board with another IDC connector is developed to latch the cable on the MDM.

Pin	Function	Description
1	SCLK	SPI Serial Clock (to ADC)
2	GND	
3	SDI	SPI Serial Data In (to ADC)
4	GND	
5	SDO	SPI Serial Data Out (from ADC)
6	GND	
7	CS	Chip Select (to ADC)
8	ID	1-Wire interface (to ID-chip)
9	-	
10	V+	from CAN-connector (pin 8)

Table 5.1: Layout of the B-sensor module connector pins

5.2 Data acquisition

5.2.1 Initialization of the MDM

In order to start data acquisition, the MDM firmware has to be initialized together with other parameters, in order to read data from the B-sensors. When the firmware initializes, all hardware devices are reset and configured. This is also applied to the B-sensors connected to the MDM.

After power-up, a CANopen node sends a so called Boot-up message (defined in the CANopen protocol). As soon as this initialization is complete, a CAN-message is received with the following syntax:

MDT-DCS Module (NMT-Slave) → Host (NMT-Master)

COB-ID	Data Byte 0
700h + *NodeID*	0

NodeID is the CAN node identifier, set by means of the ELMB onboard switches to a value between 1 and 63. NodeID must be unique and in the range between 1 and 127 in each CANbus. To start the MDT-DCS application, the following CANopen NMT message must be sent:

Host (NMT-Master) → MDT-DCS module (NMT-Slave)

COB-ID	Data Byte 0	Data Byte 1
000h	01h *(Start Remote Node)*	NodeID or 0 *(0: all nodes on the bus)*

There is no reply to this message.

Now the MDT-DCS module is operational. This means that now the I/O channels are monitored and thus CANopen PDO (see next subsection) messages containing data can be sent and received.

5.2.2 Data Readout from the MDM

Every data object in the MDT-DCS Module can be accessed through the CANopen Object Dictionary (OD). The CANopen Service Data Object (SDO) confirmed message mechanism is used to read from and write to data objects in the OD.

For the T-sensor and B-sensor data, a more efficient method of data readout is offered by the Process Data Object (PDO) messages. This unconfirmed message mechanism comes without a protocol overhead and thus is much more suitable for regular monitoring of the process data.

Using as a reference the MDM direction towards the devices, data is transmitted by the PDO message called TPDO (T for Transmit) and received by the PDO message called RPDO (R for Receive). In CANopen, the CAN-ID message content and transmission type of PDO messages may be configurable as well using the SDO mechanism with other objects in OD.

By writing to OD index 2800h using an SDO message, none or up to four B-sensor modules can be selected in a form of a bit mask [Appendix D.1]:

Host → MDT-DCS module

COB-ID	Data Byte 0
2800h	*Mask in hex*

The original design was to read out up to two B-sensor modules only. In the case of three or four B-sensor modules, connection has to be made in pairs, using the same IDC cable but with pins 7 and 8 being swapped, as illustrated in Figure 5.4.

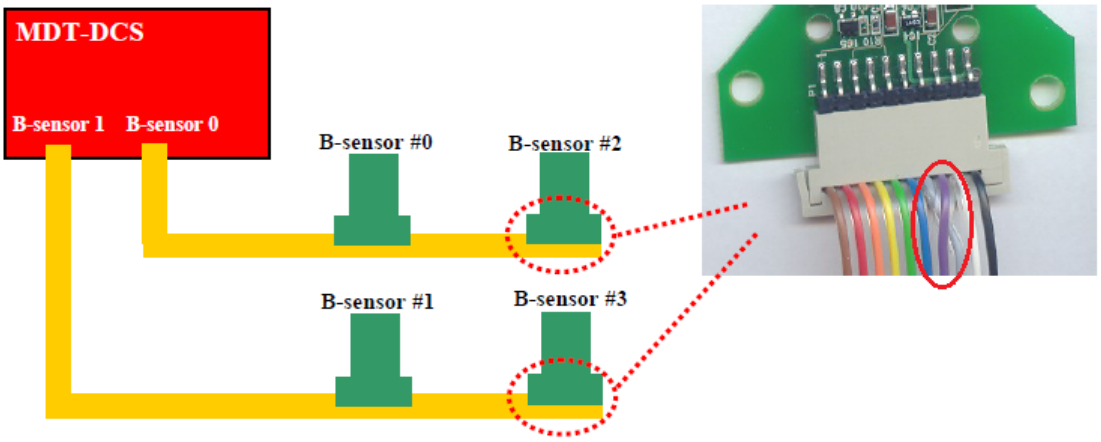


Figure 5.4: Connection of more than two B-sensors to one MDM. (Note that this is not the approved cable-connector used by ATLAS. Pictures from NSW will show the original in the next chapters.)

When the MDM is finally configured, it sends one PDO message containing 5 bytes for each B-sensor input. Per B-sensor module, 4 inputs are read: Hall sensors H1, H2, H3 and the temperature sensor. Data is identified as the following TPDO message:

MDT-DCS Module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-4
480h + *NodeID*	*Channel number*	*ADC-config*	*24-bit ADC value*

Channel number: Between 0 and 15
0-3: Hall sensors H1, H2, H3 and T-sensor for B-sensor 0
4-7: Hall sensors H1, H2, H3 and T-sensor for B-sensor 1
8-11: Hall sensors H1, H2, H3 and T-sensor for B-sensor 2
12-15: Hall sensors H1, H2, H3 and T-sensor for B-sensor 3

ADC-config: bit 7: not used
bits 6-0: ADC configuration conversion word rate (bits W0, W1 and W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B)

BIT	7	6	5	4	3	2	1	0
Meaning	Error	W2	W1	W0	G2	G1	G0	U/B

ADC value: Signed or Unsigned 24-bit ADC value with Less Significant Bit in byte 2 and Most Significant Bit in byte 4
H-sensors: 24-bit signed value that needs conversion (see Appendix D.2)
T-sensor: 24 bit unsigned value (ADC count that can be actually read in millidegrees without conversion)

5.3 Development of the B-sensor testing setup

The development of the final magnetic field monitoring consists of four phases. First, a control system for the testing of every NSW B-sensor was developed. Then, a control system was developed to commission the B-sensor installation in the integration site and then in the cavern of ATLAS experiment. Finally, a control system panel for the ATLAS experiment was developed.

5.3.1 Data Acquisition Setup

In order to avoid readout issues of faulty hardware, every NSW B-sensor was tested at building 180 at CERN. For this application, the following equipment was used:

- An MDT-DCS Module
- 2 IDC flat cables
- 1 CANbus terminal resistance of 180 Ω
- 2 CANbus cables
- 1 CANbus Power Supply of $8 - 12\,V$
- 1 Systec USB-CANmodule2 [111]
- 1 Laptop

On the test bench, the MDM reads up to 4 B-sensors at once connected with the IDC cables. Then one MDM CANbus line is connected to the terminal resistance and the other one on the CANbus power supply input. The later cable both supplies the MDM with power and receives the data via CANbus protocol. Another CANbus cable is connected to the SYS TEC device which finally transfers the data via the USB protocol to the computer. Figure 5.5 illustrates the connected hardware in a block diagram.

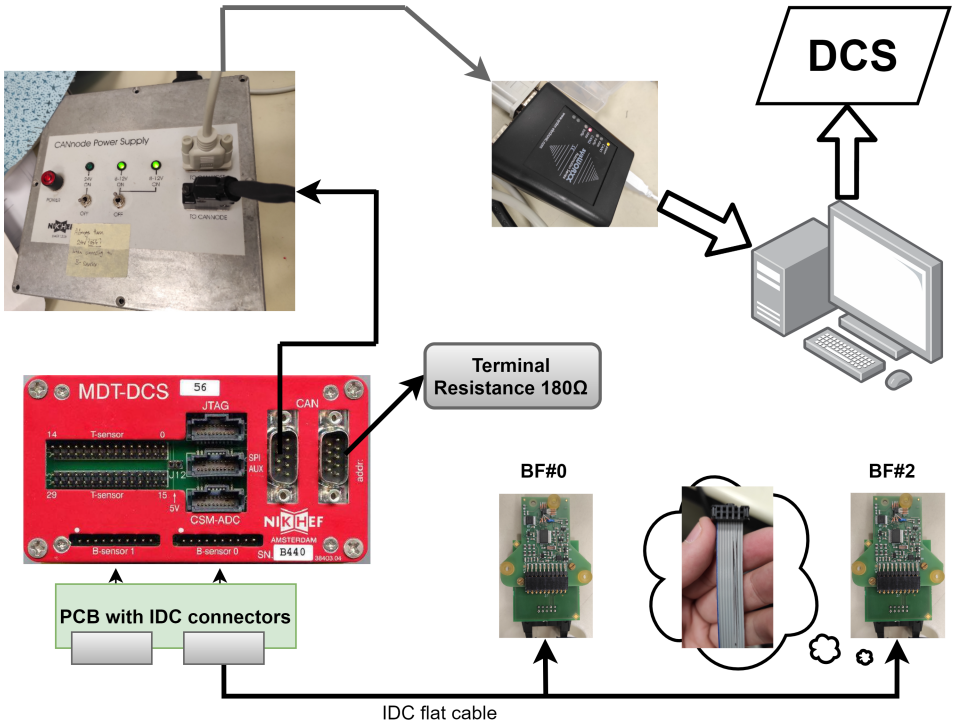


Figure 5.5: Connection diagram of all the hardware for B-sensor testing.

Using a low level tool made for the specific operating system, data can be also sent in the opposite direction. The MDM mask can be configured and data taking can begin, as described in the section above. However, an operating system specific driver has to be installed as well. This relies completely on the data the SYS TEC device sends via USB. Drivers for this application have been developed both for Windows and Linux platforms.

When the setup is complete, the next step is to establish communication with the DCS. WinCC OA features a special manager that implements the OPC UA protocol [95]. Thus, the PC only needs to act as an OPC UA Server. After the data is published, WinCC OA is configured to subscribe to the server and connect the data to the datapoints used in the DCS application.

To bring off this communication jumble, CERN Central DCS provides a compressed installation of the OPC UA CANopen Server, dedicated for such applications. After the installation, every needed component is stored in the folder below:

```
\opt\OpcUaCanOpenServer\bin\
├── AddressSpace.xml
├── CANOpenServerConfig.xsd
├── init-dev.sh
├── libsockcan.so
├── OpcUaCanOpen.conf
├── OpcUaCanOpenServer
├── OPCUACANOpenServer.xml
├── opcuacrontab.txt
├── PKI
├── ServerConfig.xml
└── server_watchdog.sh
```

The most crucial elements here are the XML file `OPCUACANOpenServer.xml` and the executable file `OpcUaCanOpenServer`. The XML file contains all the logic of the readout. Using the XML format, this file is configured to fit the MDM readout from the specific CAN port of the SYS TEC device. Running the command

```
./OpcUaCanOpenServer OPCUACANOpenServer.xml
```

the application matches the Address Space inputs and logic to the XML file and finally connects to the devices using the OPC UA protocol.

5.3.2 Data Monitoring

Data is passed through WinCC OA by configuring the appropriate OPC UA Client. Every OPC UA item is connected to the appropriate datapoint in WinCC OA. The datapoint structure for the testing project is shown below in Figure 5.6:

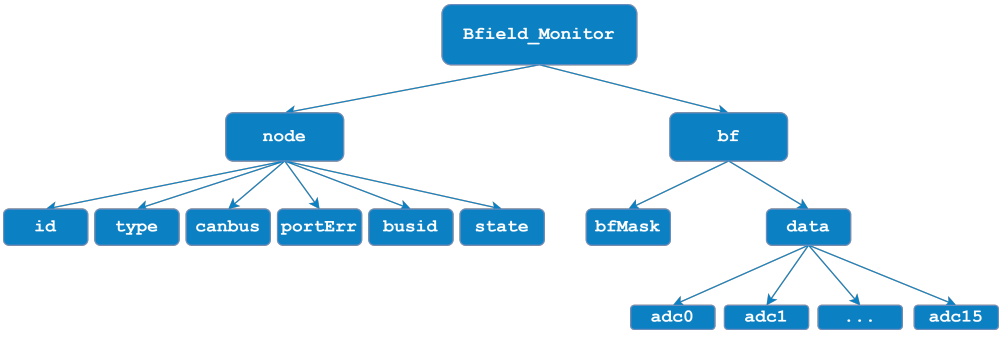


Figure 5.6: Datapoint structure used in the testbench WinCC OA project

To produce this datapoint structure, `stgBmon_service\createDatapoints_180.pnl` has been developed. It utilizes functions from the `stgBmon\stgBmon_sensorTesting.ctl` library which was also developed to serve the needs of the test bench. In this panel, the Datapoint Type and Datapoint can be created or deleted. Also (with some careful editing) the datapoints can be connected to the OPC UA items provided by the WinCC OA client manager.

Finally, the panel `stgBmonSensorTester.pnl` has been developed for the monitoring and testing of the B-sensors. Figure 5.7 shows different instances of the panel with an MDM configured with the mask `0x5`:

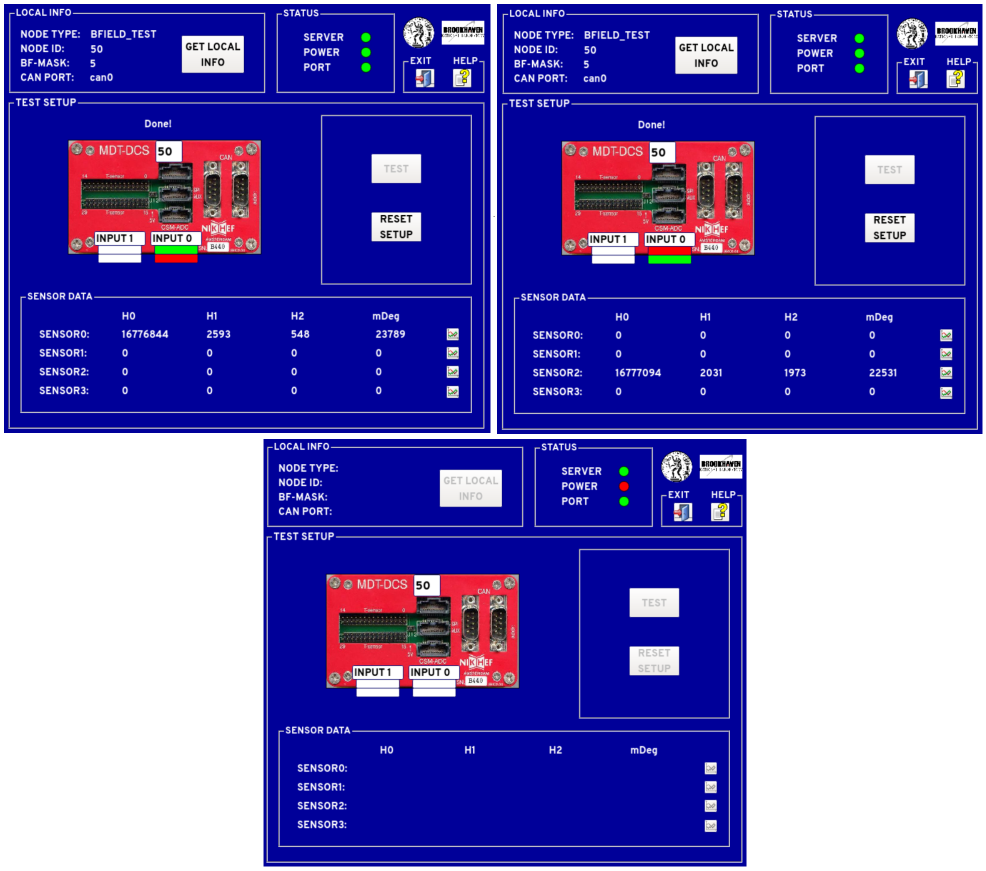


Figure 5.7: Testing of both positions in input 0 (top two images) and the case with power being off (third picture).

Similarly, figure 5.8 shows readout with an MDM configured with the value `0xF`:

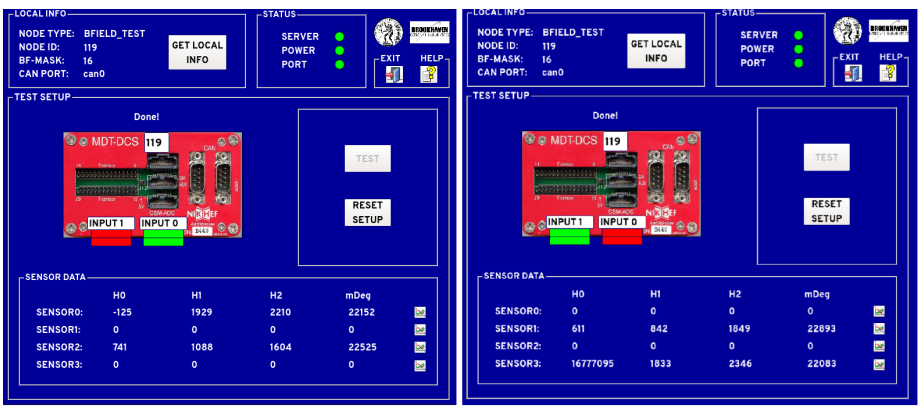


Figure 5.8: Testing of both inputs in MDM 119.

Together with the above main panel, two child panels were also developed. Clicking the trend button displays a time series of all readings and pressing the help button displays some information for the user of the panel. This is shown below in Figure 5.9:

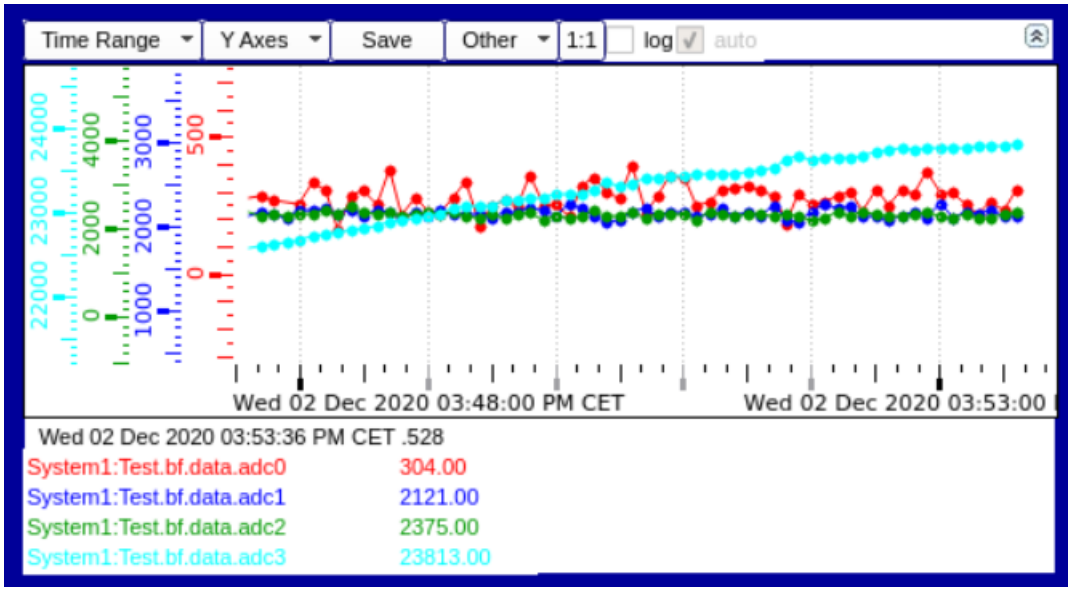


Figure 5.9: Time series of all adc readings of a sensor. Scrolling on each axis changes the axis scale and scrolling on the graph changes the scale of the whole graph.

In conclusion, the person testing the B-sensors must setup their bench as described above and install all the necessary software components. After that, a desktop sortcut is configured to start the WinCC OA monitoring panel. The team testing the sensors in building 180 successfully tested all 192 B-Sensors for NSW using this panel. The sensors were tested on the test bench as shown in Figure 5.10 and later, after installation on each sector, they were tested again in case of damage during installation on the detector (Figure 5.11).

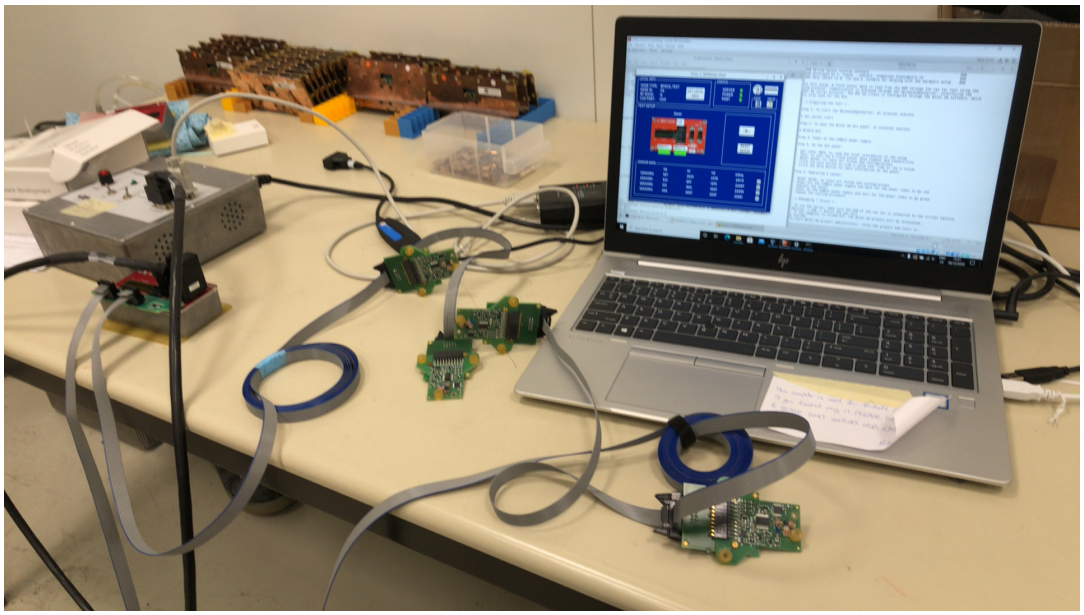


Figure 5.10: Figure shows the panel monitoring four B-sensors on the PC and all the readout components connected on the test bench.

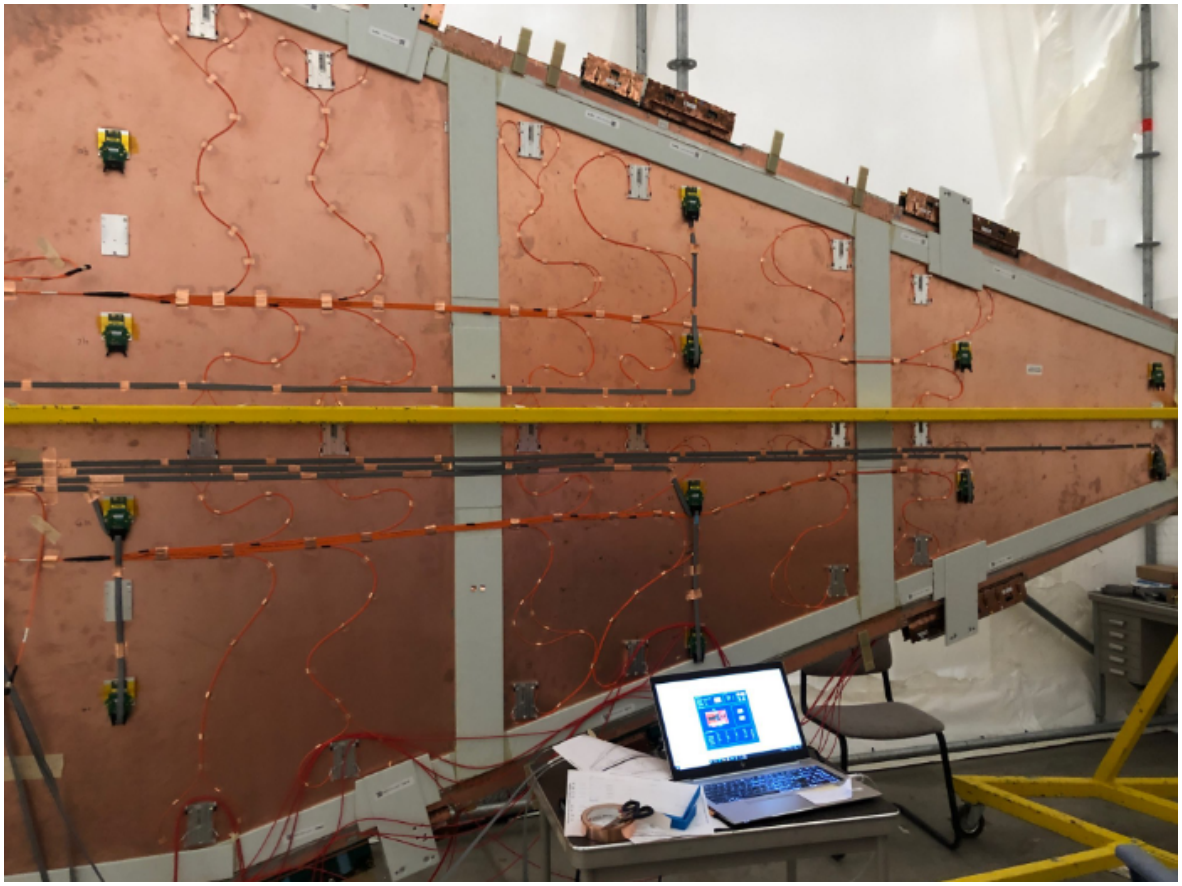


Figure 5.11: Figure shows four sensors installed on an sTGC detector being tested by the monitoring panel. The test bench is now portable by using a trolley-desk.

Pressing the button "GET LOCAL INFO" synchronizes the panel with the OPC UA items that are acquiring data from the local CANbus. As shown already above, if the power of the CANbus is off, the panel buttons are disabled. Similar behavior is displayed by the panel when the OpcUaCanOpenServer is down as well or the port has an error. Pressing "TEST" brings the data from the sensors but the user has to be careful to turn the power supply of, "RESET SETUP" and connect the new sensors.

The design of the testing setup is to read local data from the CANbus setup and display the ADC values. In principle, the user can read only ADC values but this does not pose an issue since a B-sensor malfunctioning is identified by it's bad temperature reading or no reading at all. Sensors that provide bad readout or none at all are tagged with **red** color on the panel and sensors that are healthy and can be used in the experiment are tagged with **green** color. The calibrated B-Field values are calculated later, in ATLAS experiment.

5.4 The Control System for the commissioning site

After testing every sensor that will be installed in the experiment, the next challenge to face is in the commissioning site. Building 191 at CERN is selected as the commissioning site of NSW and thus every part of its functionality was tested there. Each NSW sector was assembled and tested separately at their defined laboratories (building 899 for Micromegas and building 180 for sTGC). After that, both detector technologies were shipped to building 191 and were combined into one NSW sector. The sector's functionality was tested in combination and then it was installed on the wheel. Figure 5.12 illustrates this remarkable sight.

Figure 5.12 also shows the scaffolds used around NSW that help reach every sector and connect their services. The last service that is connected in NSW are the B-sensors and thus they can be commissioned all together.

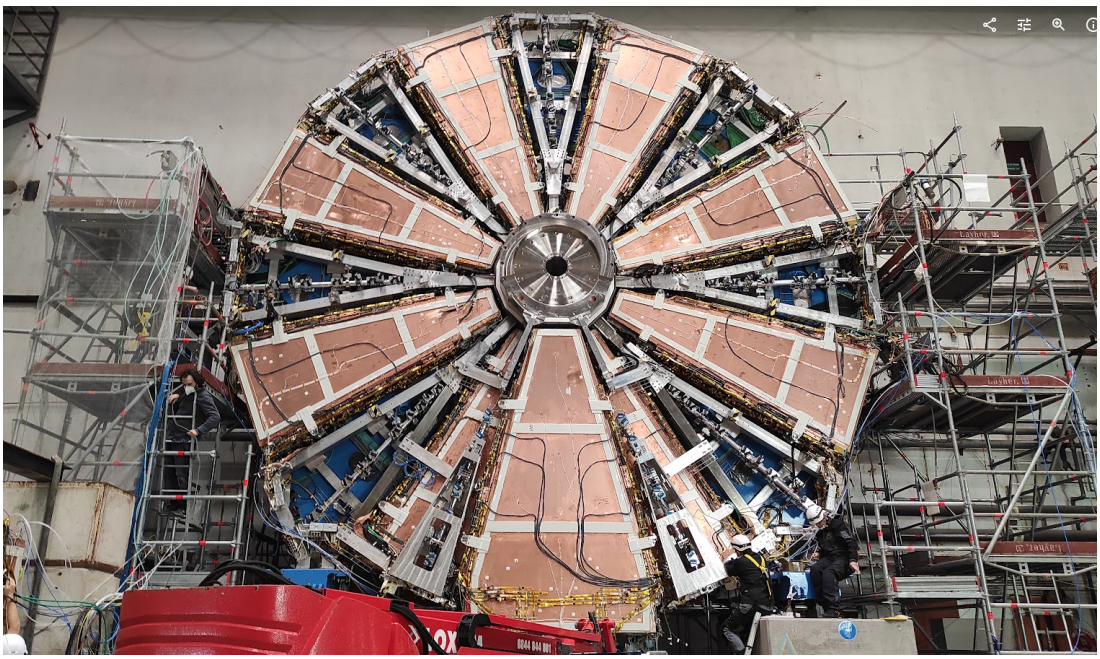


Figure 5.12: NSW side A with all small sectors and one large sector (bottom) installed.

5.4.1 On-wheel mapping and readout

Each NSW large sector features the installation of 12 B-sensors in the exact position shown in Figure 5.13:

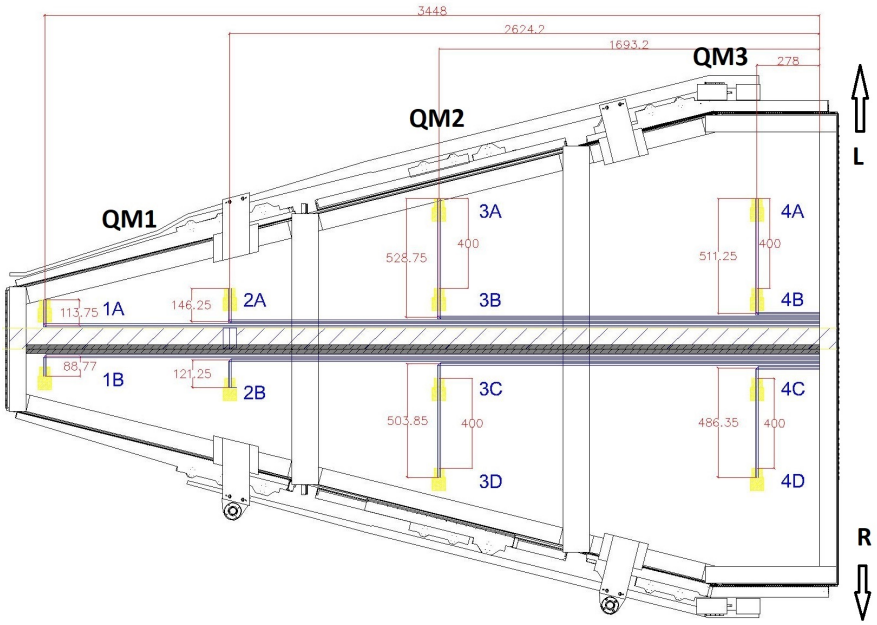


Figure 5.13: B-sensor exact positions on the NSW sector. "L" arrow represents "Lowering" side and "R" arrow represents "Rising" side.

As it has already been mentioned, each NSW side will feature 32 MDMs. Behind each large sector, four MDMs are installed on the wheel shielding disc. It is thus convenient to use these four MDMs to serve each sector's 12 B-sensors.

One might think the best idea is to split three sensors per MDM and thus conveniently use the same mask configuration and mapping everywhere. However, it is evident in Figure 5.13 that the B-sensor cable routing ends up in the outer center of the sector. At the same time, the four MDMs are symmetrically placed on the "L" and "R" sides of the sector not very close to the bisection axis. This makes it especially difficult to mix readouts of sensors that are for example in side "R" with MDMs on side "L" (which is unavoidable because sensors `3A-3B, 3C-3D,` etc are paired in one cable). In order to solve this routing problem, the final mapping was decided to look like in Figure 5.14.

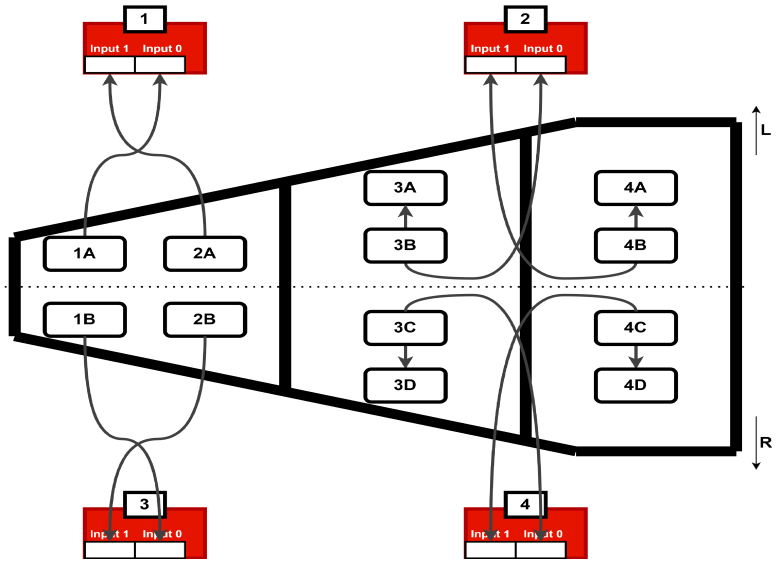


Figure 5.14: MDM installation position on wheel and cable routing of each sensor. Notice the numbers 1 through 4 on each of the MDMs. This is an installation variable called "side on spoke" which identifies the position on the wheel.

When all NSW services are connected, the above mapping scheme is used to connect the B-sensors to the MDMs. Images of this are shown below in Figure 5.15:

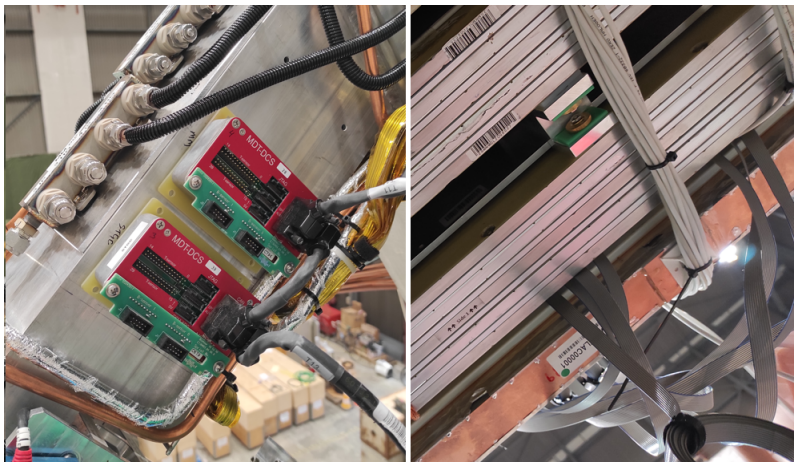


Figure 5.15: Left picture shows two MDMs installed on the wheel with their CANbus cables connected. Right pictures shows at the bottom right the IDC cables being routed.

The final list of every MDM in NSW can be seen in Appendix E with their respective CANbus chain.

Following the test setup exercise and the principles of data acquisition for a single MDM, the application can be extended to follow through for all 32 MDMs of a wheel. A PC for each wheel is configured to monitor all four CANbuses with the following setup:

- All above mentioned software corresponding to the `OpcUaCanOpenServer` must be installed
- Two CANbus power supplies were used with two SYS TEC modules with one of them serving the top half of the wheel and the other one the bottom half
- Two separate `OpcUaCanOpenServer` XML files are similarly used for each half of the wheel

To further elaborate the connections, each XML file dictates two CANbuses, both of them in different port of the same SYS TEC. This means that all 32 MDMs in total for each side are monitored by two OPC UA servers.

In order to achieve this line of multiple master CANbus devices monitoring, the CANbus protocol logic is followed:

- The CANbus power supply CAN cable connects to the first MDM in line
- In a serial manner, another CAN cable connects from the second MDM CAN port to the next MDM
- Every MDM is similarly connected to the next in line
- The second CAN connector of the last MDM in the chain is connected to a terminal resistor of 180 Ω that terminates the transfer line

This daisy chain of MDMs must not be interrupted otherwise readout is impossible. With the significant help of the services team, such hardships were avoided. It is necessary to also note that the terminal resistor value depends on the length of the transfer line in question and largely on the transfer rate desired by the means of the application. Specifically, the bitrate that was decided to be sufficient is `125000`.

5.4.2 Data monitoring and commissioning

In order for the commissioning of the B-sensors to begin, a new monitoring panel had to be designed. It had to fulfill not only containing all the necessary information but to be user friendly for everyone in the commissioning site.

First of all, the new WinCC OA application has to connect to two servers, each of them monitoring a different half of the wheel. After establishing this communication a new set of libraries, objects and infrastructure scripts had to be developed:

- `stgBmon_service\createDatapoints_191.pnl` panel: This panel creates the necessary Datapoint Types and Datapoints for the commissioning. At the same time, it connects them to the appropriate OPC UA server and its items.
- `stgBmon\stgBmonCommissioning191_A(C).pnl`: These panels are used as the interface for each wheel.
- A set of graphical objects grouped as `stgBmon\stgBmonCommissioning191\...` that are utilized by the main interface panel.
- `stgBmon_commissioning191.ctl` library: It implements functions responsible for datapoint handling, server addressing and graphical object automated actions.

The new datapoint structure is similar to the one with the testing setup, only now more of the CANbus node information is stored under *node* (Figure 5.6).

Before the commissioning starts, every NSW MDM has to be configured. Figure 5.14 shows that B-sensor mask should be `0x3` for all nodes at positions 1 and 3 and `0xF` for all nodes at positions 2 and 4. Thus, every mask has to be programmed carefully in coincidence with the mapping.

The interface used to commission the B-sensors can be seen below in Figure 5.16. Note that the nodes with node ID zero are not connected to the server and thus their value is not updated:

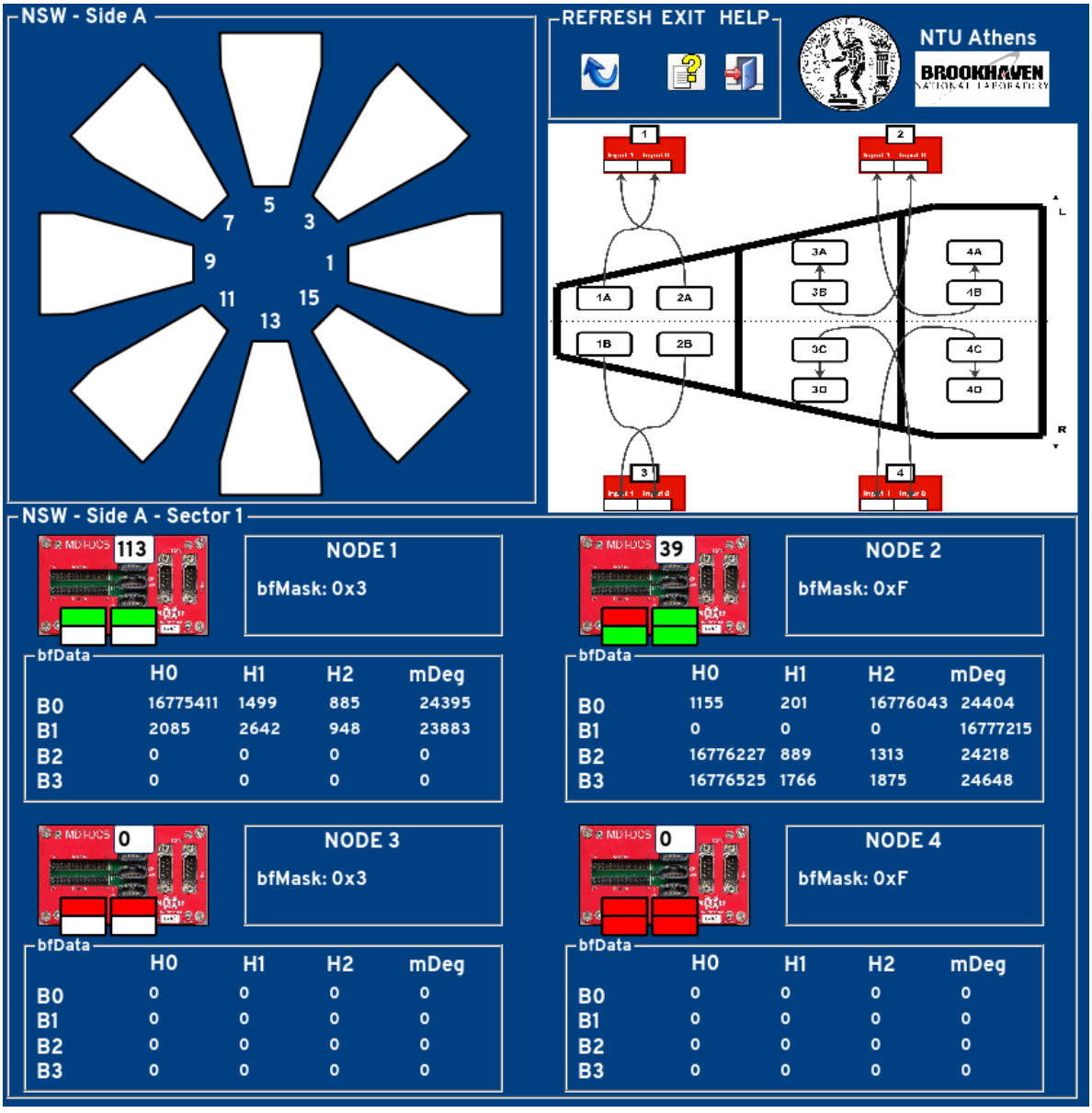


Figure 5.16: Commissioning panel for sector `A01`

In the above interface, the user can click on a sector and view every node available on it. Sensors that are connected and have healthy readout are represented with **green** color. Sensors that are connected but have a bad readout are represented with **red** color. Finally, sensor positions with **white** color are masked.

In the above example, only one of the OPC UA servers is running and thus only the top

half of the wheel is being monitored. The OPC UA items for the node IDs in node 3 and 4 are not there and thus the values are not updating. It can be also seen in Figure 5.16 that sensor `B1` is in the red state. This is because the temperature is directly measured in m°C and thus for `B1` the MDM reads some 16 777 °C which is of course faulty behavior.

Another example of the interface readout is a screenshot from sector `A07` which is shown below in Figure 5.17 below. Notice how sensors can be in bad state either with a bad temperature reading or no data at all:

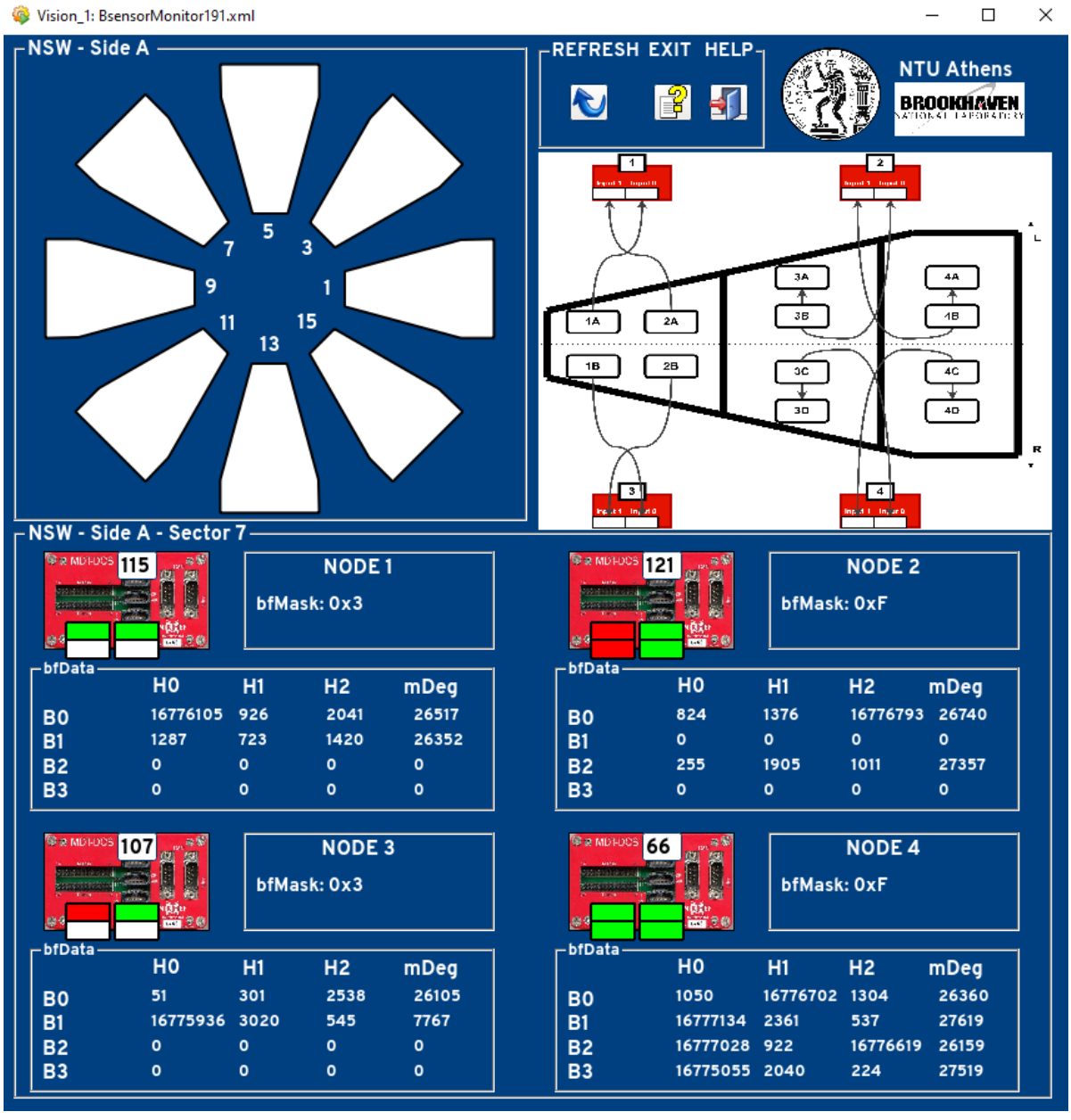


Figure 5.17: Commissioning panel for sector `A07`

Typically, commissioning of the B-sensors meant about 1 or 2 weeks per side of debugging readout errors. The most crucial one found was an IDC cable short circuiting the whole MDM 121 readout. This cable was left unplugged and thus only 2 instead of 4 sensors were lost from readout. In total, 189/192 sensors were successfully commissioned with two belonging to the sort-circuited cable and one known to possibly misbehave since installation at building 180. All three of faulty sensors reside in NSW side A.

5.5 The Control System for B-sensor commissioning in ATLAS

After every service was commissioned in building 191, both wheels where disconnected from the commissioning resources and headed towards the ATLAS experimental cavern (CERN Point 1 or P1). As one can imagine, this is a huge ordeal with unprecedented consequences in the case of an error, as the massive detector is lifted using a specialized crane and is loaded on a specialized truck. Another specialized crane lifts the detector and descends with it to around 80 m below the surface of the Earth. The assembling crew there carefully places the detector in position using several equipment and services designed for this purpose.

It is evident at this point that the overwhelmingly large machinery parts that compose the wheel, need to be connected with surgical precision in the experimental cavern's services. Therefore, another commissioning, or to put it better, validation of each service's functionality is required to take place.

5.5.1 Data monitoring and commissioning in ATLAS

Fortunately, installation of the B-sensor hardware is significantly simpler than other services. B-sensors are already connected to their mapped MDMs on wheel and the only connections that now have to be made in ATLAS are those of the CANbus power supplies to the readout.

This kickoffs the new challenge of monitoring in the actual experimental area with the equipment that will be used for the next decade. The SYS TEC modules are now replaced by one module used by the MDT detector technologies. This module supports the connection of all eight of the CANbus power supplies and sends the readout to two computers handling the data. The complete architecture of this structure will be explained in the next section with the final control system.

The next step after making the physical connections is to install all the software necessary for the monitoring. Like the previous application, the WinCC OA client connects to the server that publishes the data. The application has now new features developed that identify information in a more precise manner. The most important tools developed are:

- `stgBmon_service\createBmon_addressing_P1.pnl` panel: This panel creates the addressing from the ATLAS OPC UA servers
- `stgBmon\stgBmonCommissioningP1.pnl`: This panel is used as the interface, now showing both wheels.
- A set of graphical objects grouped as `stgBmon\stgBmonCommissioningP1\...` that are utilized by the main interface panel. These upgraded objects give more freedom to the interface and show more information.
- `stgBmon_commissioningP1.ctl` library: It implements functions responsible for server addressing and drawing of the graphical objects. New datapoints do not need to be created as the ones from building 191 can be reused.

The newly designed interface can be seen in Figure 5.18.

The panel now features a colormap of **ref** and **green** indicators of where data is bad or not. Moreover, the user can select to monitor either a side A sector or a side C. At the time of this screenshot, side C was not installed yet and thus selecting sectors from side C was disabled in the panel.

The main difficulty of the commissioning inside the experimental cavern is that not all positions of the detector are exposed. This means that if a readout issue is at hand, one should first try to solve it by means of software.

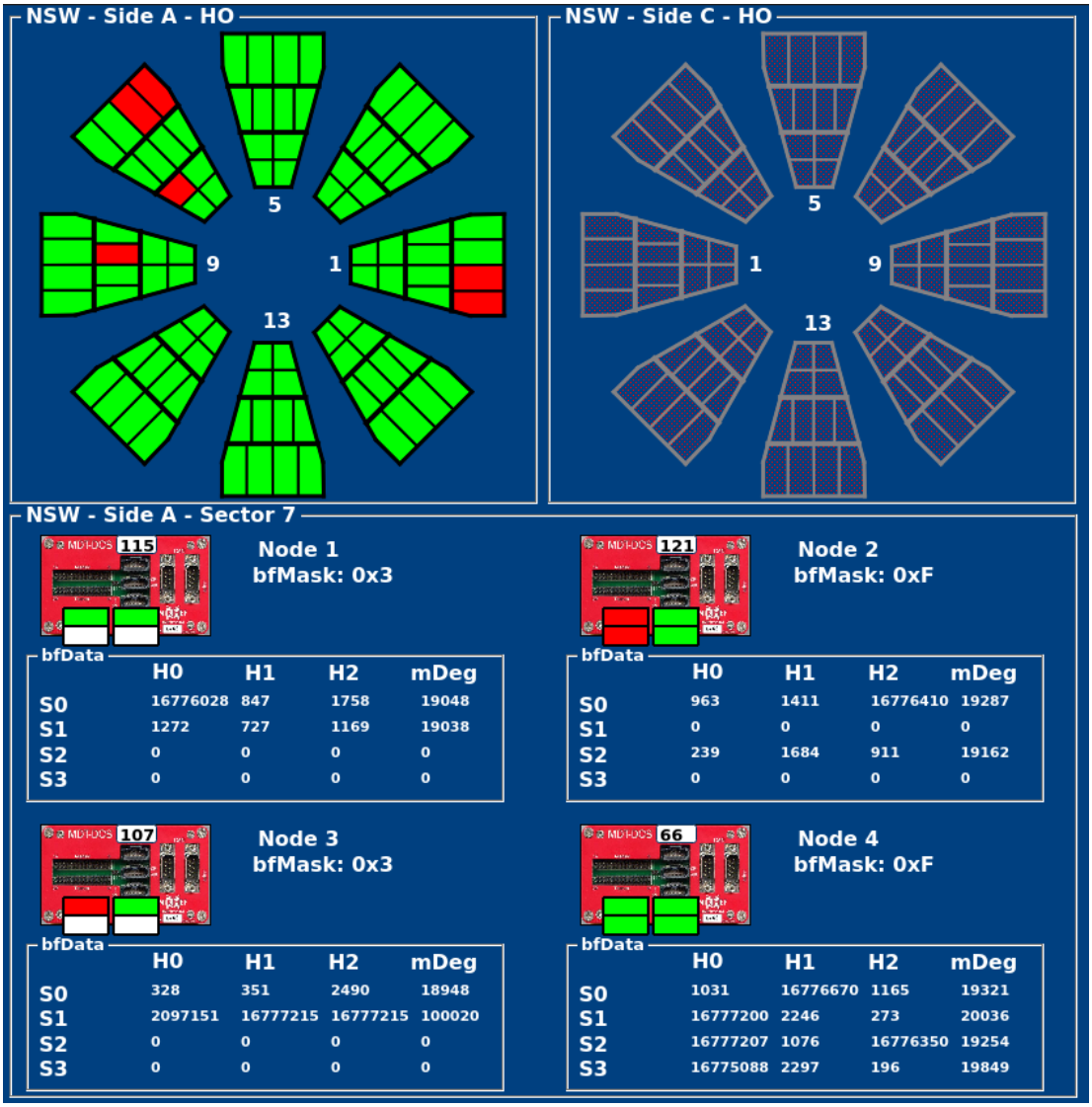


Figure 5.18: ATLAS commissioning panel with only side A installed. Side C is disabled for the user to act on.

The above screenshot does not conform with the image known from building 191. The 3 sensors misbehaving now have become 6. The strategy that was followed was to power on and off the power supplies and reconfigure the MDMs. The issue was not solved by any of these actions and neither by physical intervention.

The final solution was given by sending reset signals individually on each B-sensor with the CANopen protocol. Specifically:

MDT-DCS Module → Host

COB-ID	Data Byte 0	Comment
2600h + *NodeID*	*any value*	B-sensor 0 reset
2601h + *NodeID*	*any value*	B-sensor 1 reset
2602h + *NodeID*	*any value*	B-sensor 2 reset
2603h + *NodeID*	*any value*	B-sensor 3 reset
2700h + *NodeID*	*any value*	all sensors reset

This indeed solves the problem for a few minutes or hours but the sensors come back to a bad state. This B-sensor reset operator must therefore repeat after designated amount of readout cycles. How this is exactly done will be explained in the next section. The complete image of both wheels and the B-sensor readout can be seen in Figure 5.19.

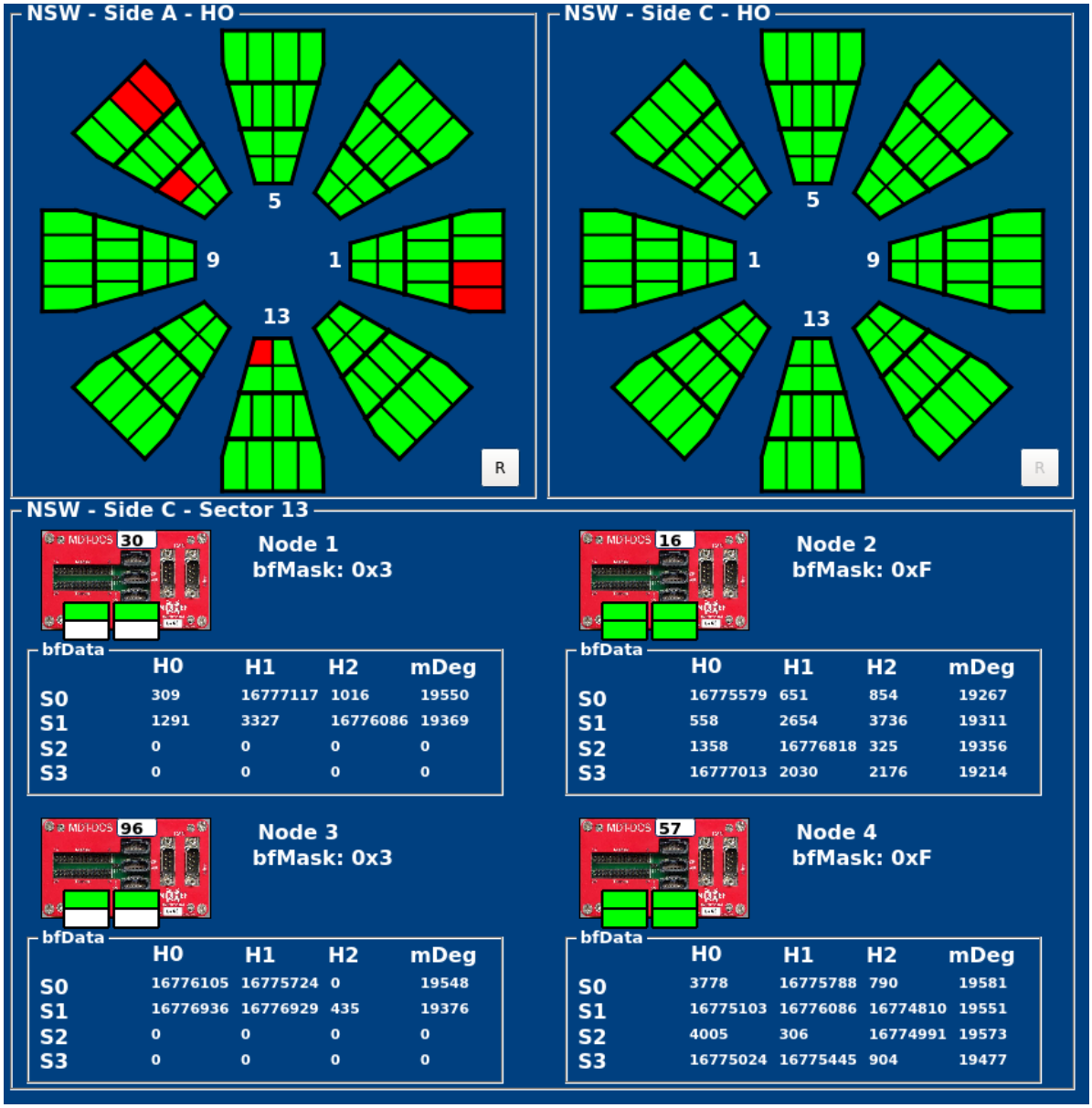


Figure 5.19: Both wheels being read out with side C showing up no readout problems.

5.6 The NSW magnetic field monitoring

So far, the control systems and monitoring applications described are all local and are not a part of the central DCS of ATLAS. The data available for the applications is directly connected to the servers. This means that all the information about any magnetic field calculation is missing. Moreover, some information can be only acquired through the JTAG interface of the MDM and this is not published by the server. This section will thoroughly describe the architecture of the magnetic field monitoring for NSW. It will also describe the WinCC OA application design and development for the ATLAS DCS.

5.6.1 Physical connections and hardware communication

In the ATLAS underground area, the CANbus cables are drawn from UX15 (Underground Experimental area) and are routed inside USA15 (Underground Service Area). USA15 hosts two computers, named `pcatlmdtmdm4` and `pcatlmdtmdm5`. These computers were originally dedicated for the MDT detectors and already implement the readout of the magnetic field and temperature for the MDT detectors of the Muon Spectrometer. At the same time, they use two multihost SYS TEC modules with several ports each, two of them dedicated to serve NSW. The two CANbus cables of NSW side A are thus connected to the multihost SYS TEC module of `PCATLMDTMDM4` and the other two of side C are connected to the multihost SYS TEC module of `PCATLMDTMDM5`.

The XML files of each computer implement the OpcUaCanOpenServer for all MDMs, both of MDT detectors and NSW. Specifically, Table 5.2 show the detailed connections:

CANbus	Port	Computer	Project
EI_A_top	can1	PCATLMDTMDM4	ATLMDTMDM4
EI_A_bottom	can0		
EI_C_top	can0	PCATLMDTMDM5	ATLMDTMDM5
EI_C_bottom	can1		

Table 5.2: All communication parts of the CANbuses.

In total, one server runs for each wheel in the above computers.

The WinCC OA application developed for the magnetic field monitoring is installed inside `PCATLMUO03` and is named **`ATLMUONSWMDM`**. At the same time, the OPC UA client through WinCC OA subscribes to the items of both servers and makes the data available to the Datapoints.

The NIKHEF team that has developed the MDT MDM readout has already implemented a logic which reads all the values, alarms and states sent by the MDM and thus this logic does not need to be re-implemented. However, the Datapoints used by the MDT projects are in list-lookalike format of every sector that does not conform with the NSW detector structure. For this reason a copy mechanism needs to be developed in order to effectively copy the Datapoints from the `ATLMDTMDM4(5)` projects to `ATLMUONSWMDM`.

The library `stgBmon\stgBmon_infrastructure.ctl` was developed, order to meet the needs of this Datapoint handling. This library handles both the Datapoints of this project and those produced by the MDT group. It also implements some functions related to the data archiving and some calculations that are however of secondary importance.

This project was being built over the period of the B-sensor testing and later during the commissioning of the New Small Wheels in building 191. Thus, there was sufficient time to also develop a simulation algorithm that mimics the copy mechanism in ATLAS experiment.

Before going through the simulation script and the copy mechanism, the structure of the Datapoints needs to be drawn out. The MDT group stores the data as shown in Figure 5.20:

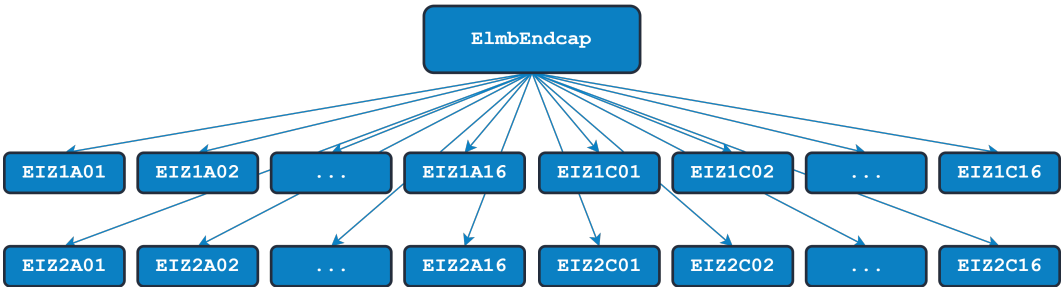
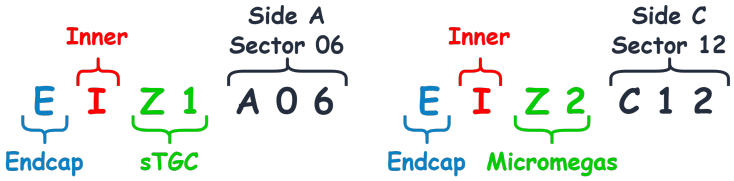


Figure 5.20: MDT-like Datapoint structure with NSW detectors.

The structure shown above, has every sector Datapoint as a branch of the Datapoint Type tree. Below each sector is the ELMB node and below it all the data. This is convenient for the MDT detectors which have only one MDM serving them. Thus, the abbreviation shown above in the Datapoints, specifically means:



In the above image, everything is self explanatory except for the `Zi` part. To give more context, `Zi` refers to the detector "radius", measured from the beam interaction point. Since an NSW sector is composed of sTGC detectors on the outside layers and two Micromegas detectors in the center, the mapping should read:

`Z1` → `sTGC`
`Z2` → `Micromegas`
`Z3` → `Micromegas`
`Z4` → `sTGC`

For NSW however, this is convenient only for T-sensor readout, since sTGC and Micromegas technologies both have their own T-sensor mapping. At the same time, T-sensors are installed in all detectors of NSW and since the mapping is the same, `Z1` is dedicated for sTGC and `Z2` for Micromegas. Contrariwise, for the magnetic field monitoring in NSW, only the outer sTGC sectors are installed with B-sensors. Thus, the datapoint structure for magnetic field monitoring, was decided to look like below in Figure 5.21:

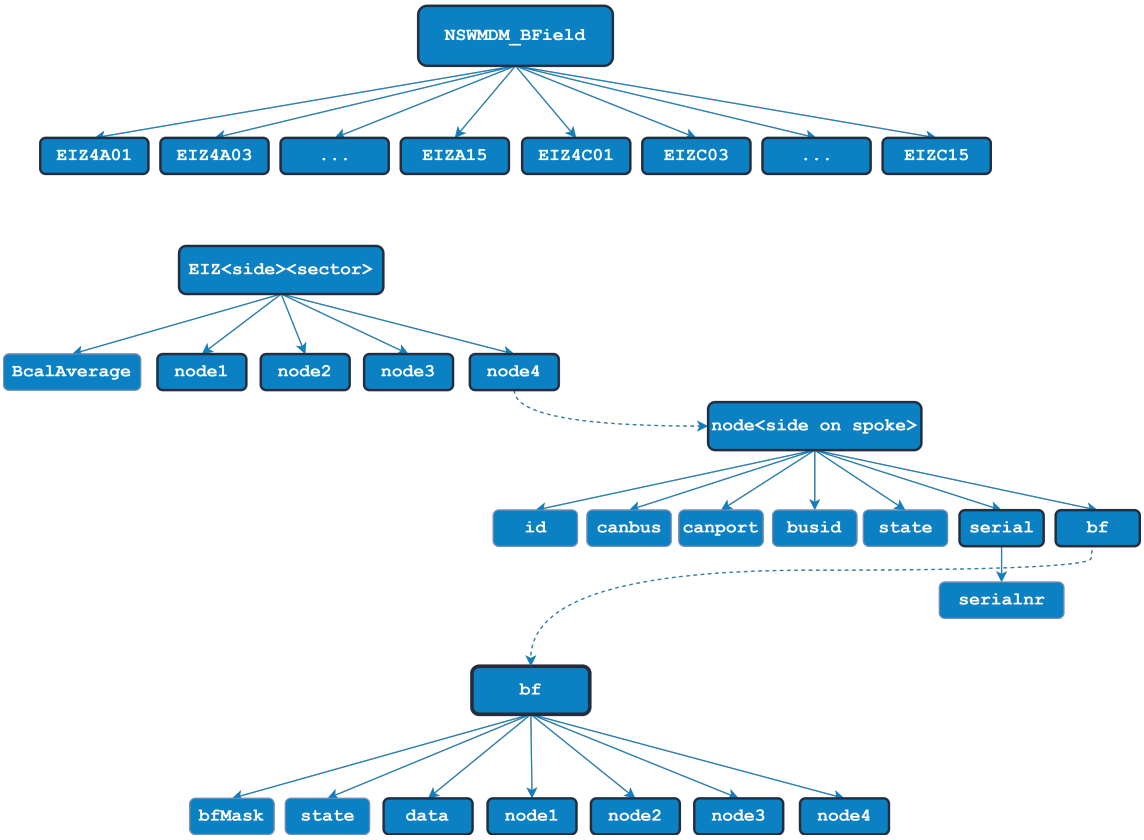


Figure 5.21: The Datapoint structure of the NSW magnetic field monitoring.

Both of these structures shown have the sensor data below the node `bf`. Specifically, the data is stored in the structure shown in Figure 5.22:

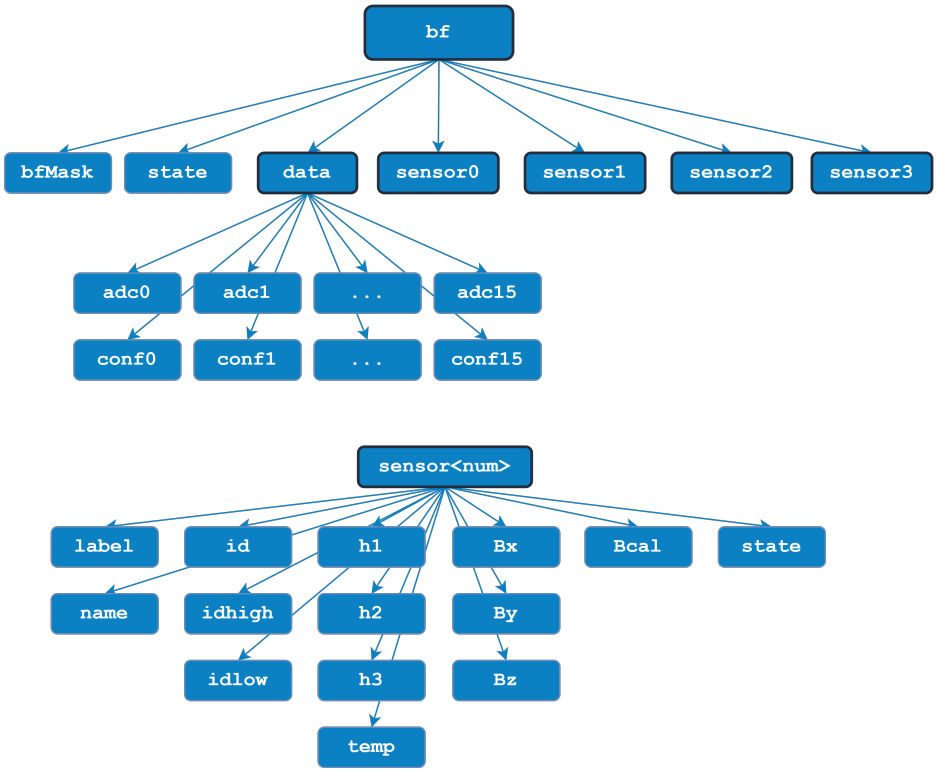


Figure 5.22: Structure of the data stored under the MDM bf node.

The value carried by every Datapoint Element mentioned so far is described in Appendix F.

With this mapping at hand, the copy mechanism has to copy the data from the structure shown in Figure 5.20 to the data structure shown in Figure 5.21. WinCC OA permits projects to be connected in a distributed manner and thus Datapoints of one application are available to others. In principle, the copy mechanism is a script that runs permanently in the background of the project. The logic behind this algorithm is quite simple:

- It utilizes functions from the `stgBmon_conversions.ctl` and `stgBmon_constants.ctl` libraries developed for this project.
- It initializes all variables responsible for the one project to locate the other.
- It checks for connection to the remote systems `ATLMDTMDM4` and `ATLMDTMDM5`.
- Connects the node data, bf information, raw data and then sensor data from the remote projects to `ATLMUONSWMDM`.

We need to be careful about using the term "copied" here as the data is not copied with the original sense of one value field into another. The script, runs using the WinCC OA function

`dpConnect(`*`string`* `workfunction,` *`string`* `datapoint)`

The `workfunction` here is `copyValue(`*`string`* `remote_dpe,` *`anytype`* `remote_value)` and the `datapoint` comes in a loop, getting every remote datapoint that needs to be in the project. `dpConnect` actually connects the two Datapoints and the value of the local Datapoint is updated when the value of the remote Datapoint is updated as well. This saved both computational power and memory as Datapoints are not copied all the time but only when their value changes. Two versions of these scripts were developed, one for side A and one for side B. Later, these scripts were combined into one script: `bfCopycat.ctl`.

This communication architecture was known before the wheels were installed and thus a simulation of this copy mechanism was built in advance. The algorithm implementing this project is called `bfSimulation.ctl` it is a simple design that updates in real time Datapoints in the MDT structure with pseudo random values. In principle, the algorithm uses the WinCC OA function `secureRandom()` which calculates a squence of secure integer pseudo random numbers in the range of `0` to `32767`.

The ADC values of the Hall sensors that represent temperature fields are updated at around 25.00 °C using the calculation

$$\texttt{ADC\_value} = 25000 \pm \frac{\texttt{secureRandom()}}{33}$$

where + is used for side A and − for side C. In a simpler way, the Hall sensor fields for magnetic field reading are updated using the calculation

$$\texttt{ADC\_value} = \texttt{secureRandom()}$$

The ADC values are then converted to magnetic field values using a function defined abstractly as a the linear equation

$$B = \frac{h}{h_0} + B_0$$

were h is the Hall unsigned ADC field, $h_0 = 5000$ is an abstract value and $B_0 = 0.4\,\mathrm{T}$ an abstract constant. This is to simulate the calibration constants of each sensor. All these abstractly defined constants are only a way to test the functionality of the monitoring panel. It will be made clear later in this section how these values will be replaced.

The final calibrated value of magnetic field, is measured by each coordinate's component as

$$\texttt{Bcal} = \sqrt{\texttt{Bx}^2 + \texttt{By}^2 + \texttt{Bz}^2}$$

Values that correspond to MDM information are hardcoded into matrix elements and fed into the Datapoints.

The last procedure before developing and testing the interface, is to calculate the average magnetic field value in each sector. This is rather useful because each torus of the endcap toroid magnet of ATLAS is directly adjacent to the large sectors. The script developed for this is `AvgCalculator.ctl` and it implements a logic of calculating the average value of a sector's magnetic field readings, only for the sensors that are in a good state. This is the simple calculation

$$\overline{B}_{\texttt{cal}} = \frac{1}{N} \sum_{i=1}^{N} B_{\texttt{cal,i}}$$

where N is the number of B-sensors that in a good reading state.

5.6.2 The Control System Development

The final panel that will be used for magnetic field monitoring in ATLAS has to implement many functionalities so that the operator can access all the information at once. The `stgBmon.xml` panel will not be part of the Finite State Machine of ATLAS DCS. The challenge therefore is to create one panel that can display with different instances different granularities for the operator to monitor.

The ATLAS DCS already features an expert panel which monitors the barrel toroid magnet magnetic field. This interface was designed by the MDT group and displays the readout from the B-sensors installed on the barrel MDT detectors. It can thus be identified that the most crucial implementations that the magnetic field monitoring should bring off for NSW are:

1. Display of a colormap of the magnetic field readout around NSW by the B-sensors
2. Display specific information about the sensors, the MDMs, the mapping and their connectivity
3. Design of a user-friendly interface for ATLAS operators to navigate and monitor all variables
4. Archive the monitored magnetic field data

In order to successfully display the data incoming to the WinCC OA application by the OPC UA server, first, some state checks need to be applied. Except from sending the 24-bit ADC unsigned value to the MDM, the B-sensors also provide state bits that inform the MDM of any activity. These states are shown in Table 5.3 below:

State	Value	Description
BSENSOR_OKAY	0	Healthy B-Sensor readout
BSENSOR_DEAD	1	B-Sensor non responsive
BSENSOR_BAD	2	B-Sensor bad readout
BSENSOR_EMG	3	B-Sensor emergency status
BSENSOR_TEMP	4	B-Sensor temperature problem
BSENSOR_MASK	5	B-Sensor masked
BSENSOR_UNK	6	Unknown error
BSENSOR_NOID	7	B-sensor no ID readout

Table 5.3: States of a B-sensor readout.

The above list of states is used in `stgBmon_constants.ctl`. This library was developed to set system constants, states and value ranges. Before updating the color in a connection on the interface, checks have to be made whether a B-sensor is `MASKED`, `BAD`, `DEAD`, `OKAY`, etc. The general principle, is to see if a sensor is in the `OKAY` state. Then, it's value is updated and the sensor position is displayed in green. If a sensor is in the `MASKED` state, it will be displayed in grey.

The data color scheme that will be displayed in the panel is currently abstract. All the value ranges are mapped with color codes inside the library `stgBmon_constants.ctl`. For the final experiment, this library can be edited with the real values. The library and interface were both built in such a way that the future developer will be able to easily maintain the code in `stgBmon_constants.ctl`. The intervention should be very small (changing the numerical values of the ranges) and all the states and mapping will propagate to the graphical objects in an automated manner.

The library currently implements 10 ranges plus the fatal and masked states. They are mapped in the following constants:

```
// B-Sensor color codes
const string BSENSOR_GREEN_1  = "{0, 250, 0}";
const string BSENSOR_GREEN_2  = "{25, 250, 0}";
const string BSENSOR_GREEN_3  = "{50, 250, 0}";
const string BSENSOR_GREEN_4  = "{75, 250, 0}";
const string BSENSOR_GREEN_5  = "{100, 250, 0}";
const string BSENSOR_GREEN_6  = "{125, 250, 0}";
const string BSENSOR_GREEN_7  = "{150, 250, 0}";
const string BSENSOR_GREEN_8  = "{175, 250, 0}";
const string BSENSOR_GREEN_9  = "{200, 250, 0}";
const string BSENSOR_YELLOW   = "{250, 250, 0}"; // Warning
const string BSENSOR_ORANGE   = "{250, 150, 0}"; // Error
const string BSENSOR_RED      = "{250, 0, 0}"; // Fatal/Dead/Bad
const string BSENSOR_GREY     = "{200, 200, 200}"; // Disconnected/Masked
```

```
// B-Sensor colormap ranges
// TO BE UPDATED WITH REALISTIC VALUES
const float BSENSOR_GOOD_1 = 3.25;
const float BSENSOR_GOOD_2 = 3.50;
const float BSENSOR_GOOD_3 = 3.75;
const float BSENSOR_GOOD_4 = 4.00;
const float BSENSOR_GOOD_5 = 4.25;
const float BSENSOR_GOOD_6 = 4.50;
const float BSENSOR_GOOD_7 = 4.75;
const float BSENSOR_GOOD_8 = 5.00;
const float BSENSOR_GOOD_9 = 5.25;
const float BSENSOR_WARNING = 5.75;
```

Since most of the above numbered implementations mentioned are heavily dependent on the mapping, significant part of the effort in the development was spent in the design of the interface. The main idea behind it, is to have different instances of the interface. The panel displays both wheels on top with their average magnetic field value. Then, on the bottom, the operator can switch between different views. The interface should at first display the colormap, as shown in Figure 5.23:

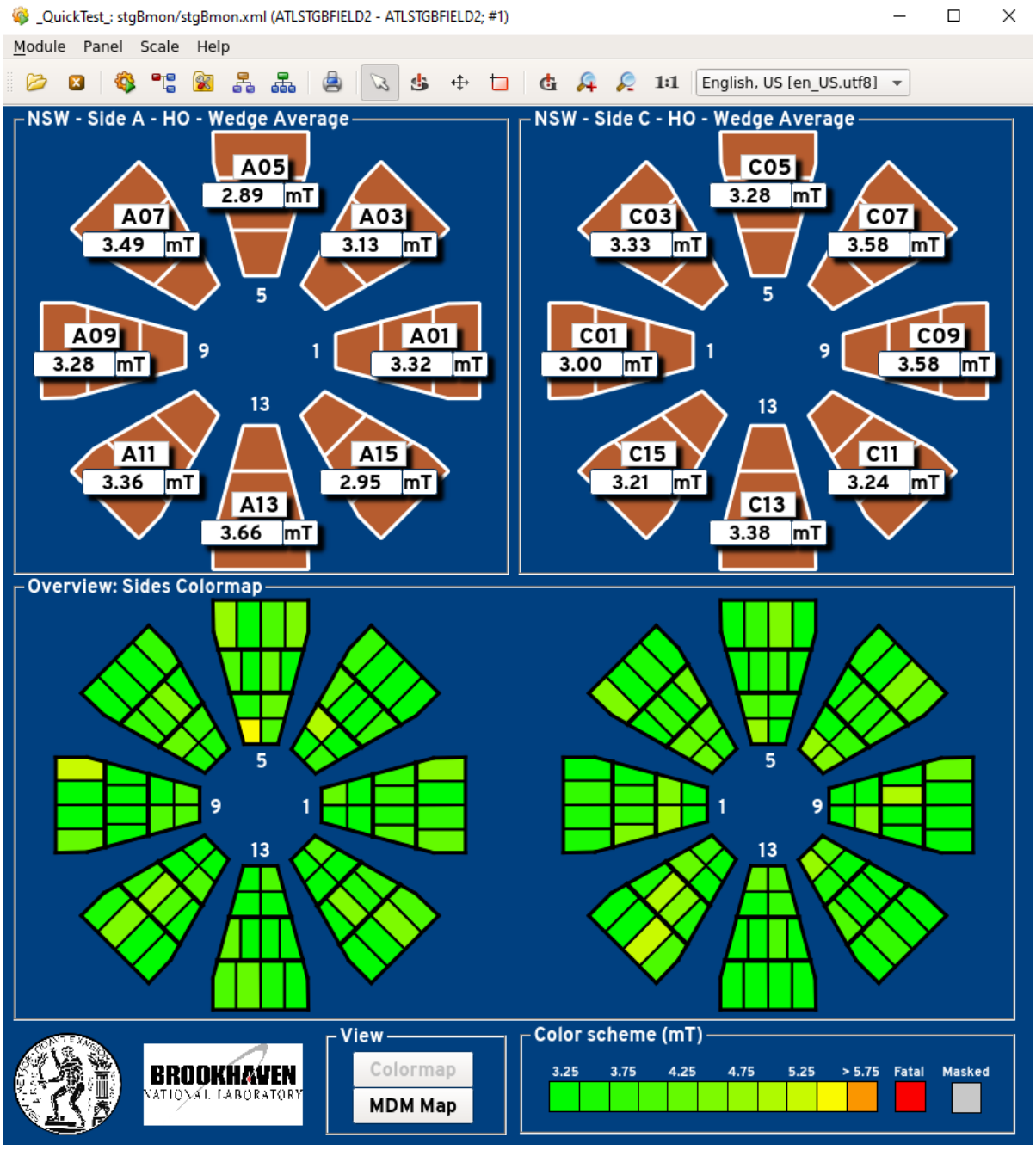


Figure 5.23: "Colormap view" of the `stgBmon.xml` panel. The displayed values are from the simulation. It represents sensor granularity.

It is not straightforward for the average reader but in the WinCC OA environment all shapes in the above image are graphical objects manipulated by scripts and functions developed for the desired actions. The operator can click on any sensor, sector or MDM and the information is propagated to the next panel. For the successful placement of these graphics and their

panel actions, the library `stgBmon_panel_graphics.ctl` was developed. It is composed of 16 functions that implement the adding and removal of families of objects with the correct variables connected to the WinCC OA Datapoints.

By clicking the view button "MDM map", the operator can see two lists of all the MDMs in each wheel. This view can be seen in Figure 5.24:

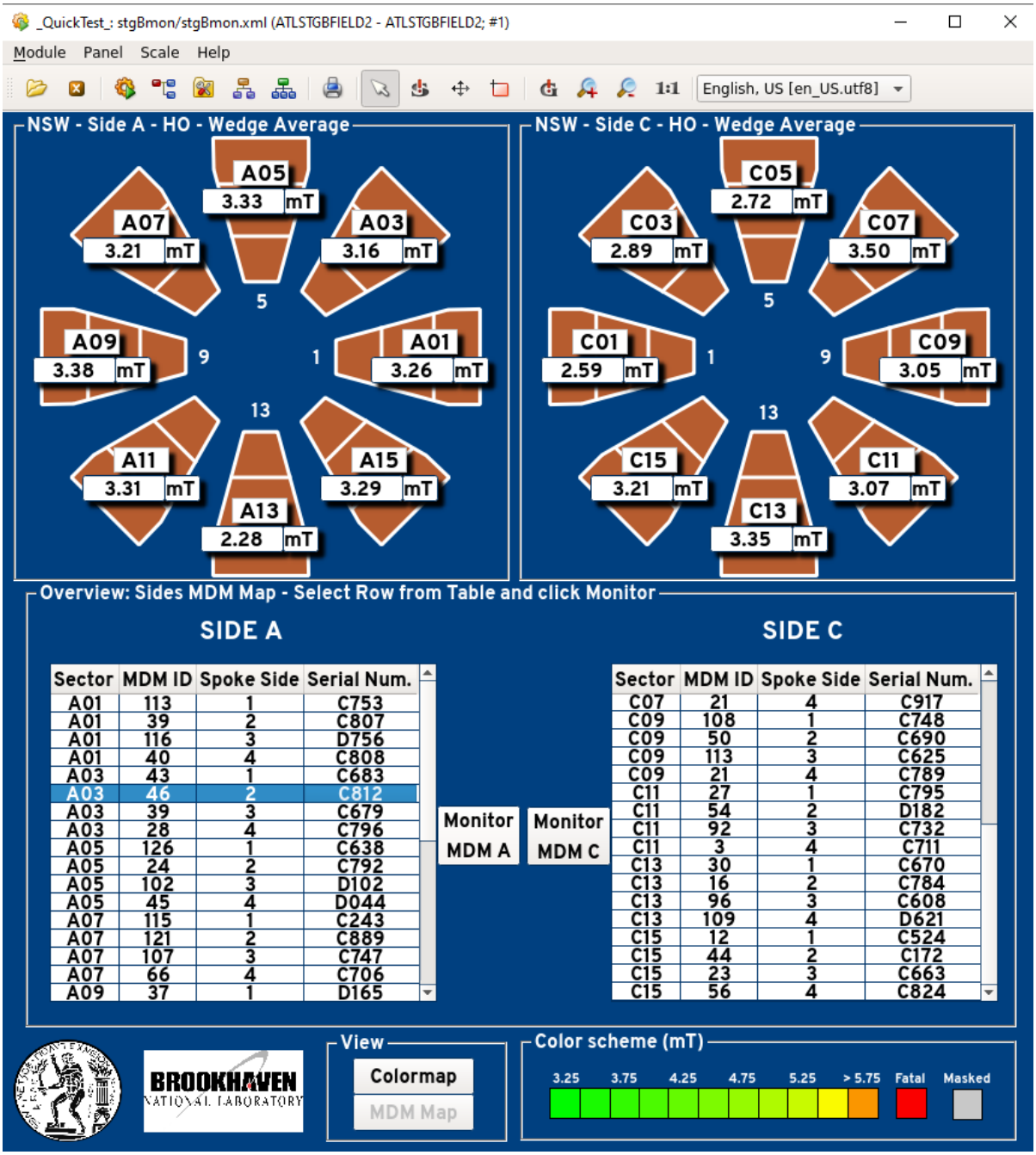


Figure 5.24: "MDM map view" of the `stgBmon.xml` panel. It represents MDM granularity.

The operator can now select an MDM and click "Monitor MDM A/C". This will prompt a new view, the MDM view. MDM view is available both through the MDM map and the sector view.

In case the operator wants to monitor a specific sector, clicking any sector on the top half of the panel will produce the sector view. In the sector view, the operator can monitor all sensors and MDMs installed on this sector. Figure 5.25 shows this instance of the panel:

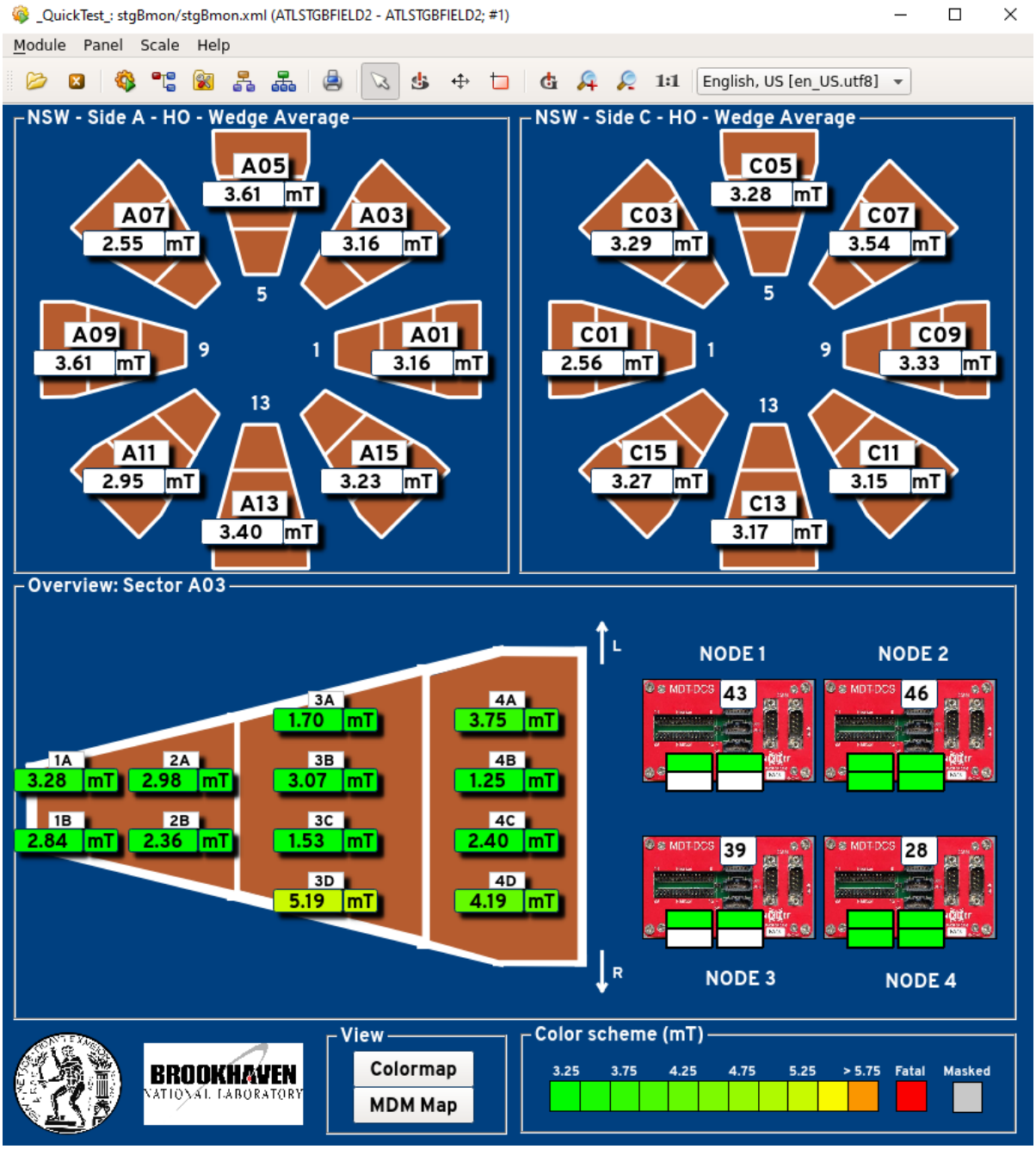


Figure 5.25: "Sector view" of the `stgBmon.xml` panel. It represents sector granularity.

Notice how the panel displays both mapping information and connectivity. The operators can thus see where each MDM is located on the wheel and which sensors are connected to it. Sector view is especially useful to watch incoming data from all sensors in one torus of the endcap magnet. The "Sector view" also displays connectivity on the MDMs. The four positions on each MDM correspond to the sensors connected, just like the scheme used in the commissioning site and the testing panel.

The operator can now either re-select any other view from the "View" section on the bottom of the panel or another sector from the top of the panel. However, the "MDM view" is prompted when the operator clicks on an MDM (or selects an MDM from the "MDM map view"). The "MDM view" is shown below in Figure 5.26:

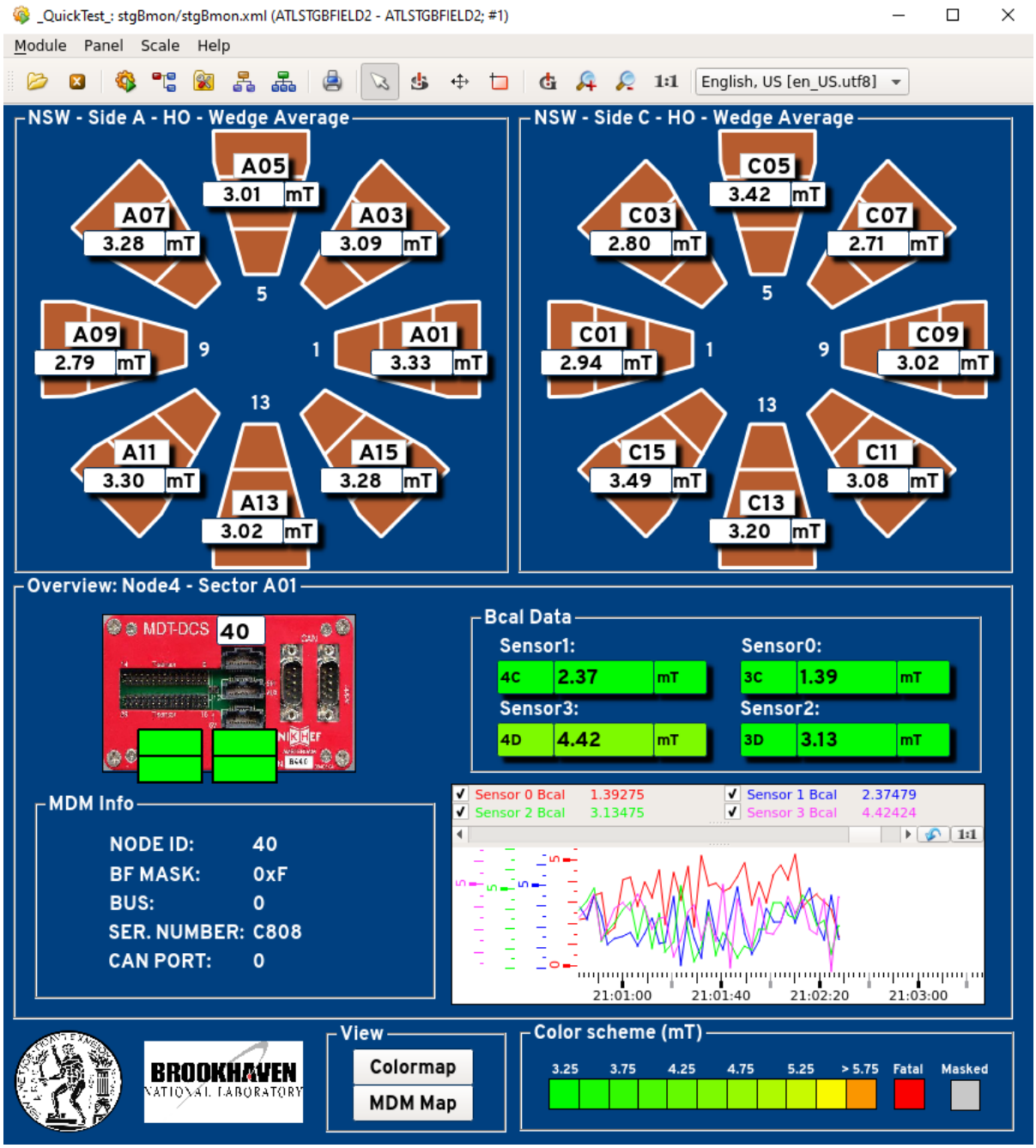


Figure 5.26: "MDM view" of the `stgBmon.xml` panel. It represents MDM granularity and a time graph.

The above interface is very similar to the testing panel but some extra information that concerns the mapping. First of all, the basic MDM Info is displayed. At the same time, the sensor label is also propagated at each sensor's textfield. This way the operator can tell where exactly on the sector this sensor is installed. Connectivity is also displayed on the left side of the panel at the MDM input positions. The panel also includes a graph that represents historical data of the sensors that are being monitored.

The final instance of the interface is that of a single sensor. The operator can select to monitor a specific sensor in three ways:

1. By clicking on any sensor in the "Colormap view".
2. By clicking on any sensor on the sector in the "Sector view".

3. By clicking on any data field of a sensor in the "MDM view".

The instance of the panel that prompts, is the following in Figure 5.27:

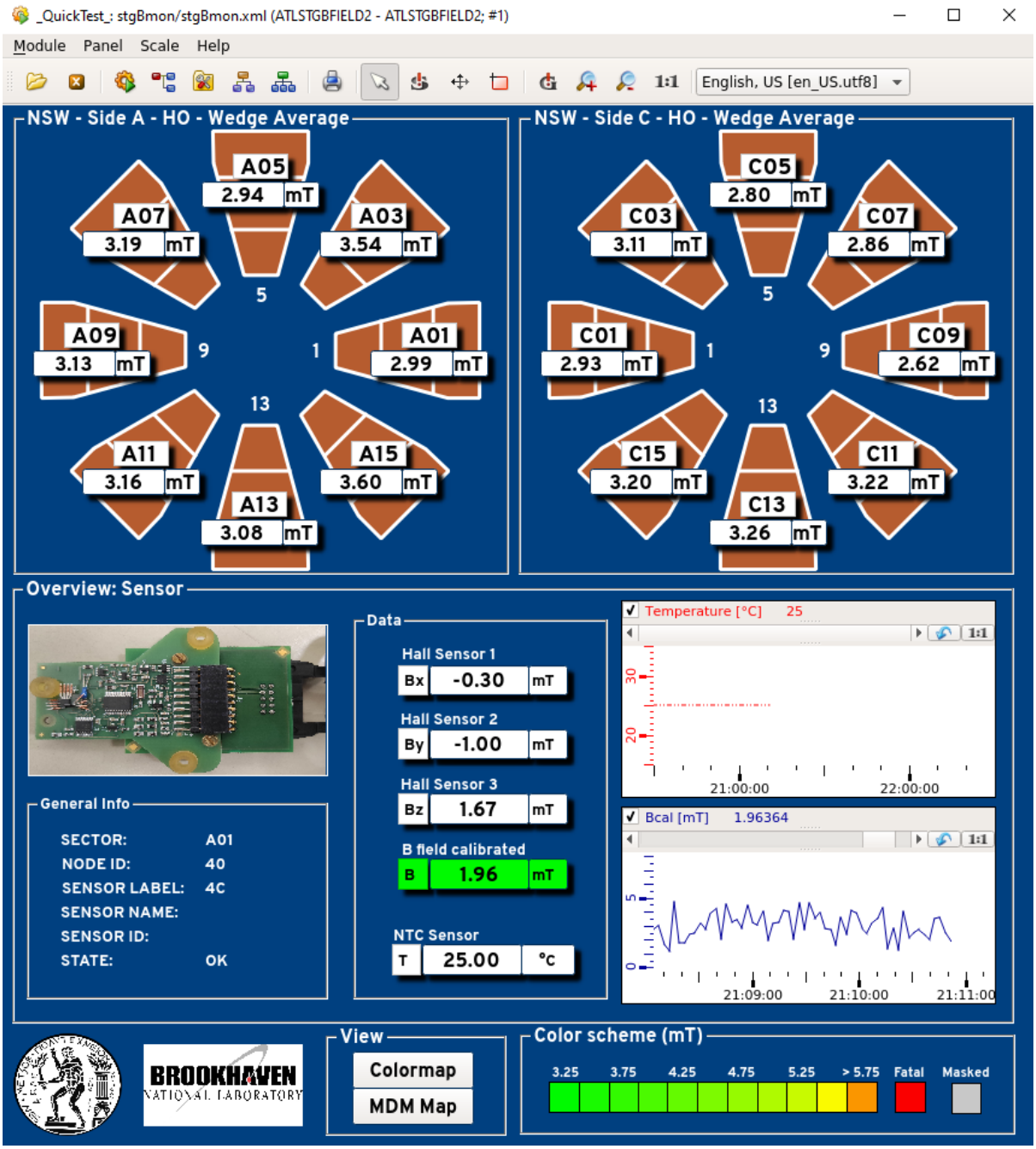


Figure 5.27: "Sensor view" of the `stgBmon.xml` panel. It represents sensor granularity together with information and all the value fields.

In this instance of the panel, the general info of the Sensor is displayed on the left. The sensor name variable is a code label that was introduced during commissioning, for example sensor `NSW#074`. The most important fields which here are temperature and `Bcal` are displayed in historical graphs.

All the values that are displayed in a graph so far are archived in the NSW specific Oracle database for the MDMs. WinCC OA features connection to a Database and thus scripts that automate this have been developed for this project.

The values that remain `0` or empty in the above panels are not documented for the commissioning and thus already exist in a database. At the same time, they are read by the OPC UA server on the MDT PCs. Conclusively, during the simulation they are not displayed in the panels. However, the correct datapoint is indeed connected to the displayed text and during the experiment these values will automatically be updated.

5.7 Maintenance and status of NSW magnetic field monitoring

All of the code, panels and graphical objects developed for this application is in GitLab in the repository **stg-bfield-nsw-dcs**. In order to install this is ATLAS experiment, the repository is connected to the NSW PC and everything is pulled from the repository to the local folder.

As it has already seen so far, the values displayed in the panel are all from the simulation. There are some necessary actions that need to take place for the final experiment to be ready:

- In `stgCopycat.ctl` script, the local project name has to be replaced with the hardcoded `ATLMUONSWMDM`. At the same time, the remote project has to also be hardcoded as `ATLMDTMDM4` or `ATLMDTMDM5`.
- The desired engineering unit has to be selected, if the magnetic field is not measured in mT. It can be changed at `objects\panels\stgBmon\BmonField_Sector.xml` and `BmonField_SectorAverage.xml`.
- NIKHEF experts have to bring off the Datapoint structure that is currently simulated, in order for the data to be successfully shown in `stgBmon.xml`. Also, there needs to be a matching of sensor ID and calibration constants through the database in order to obtain `Bcal` values.
- The value ranges of the color scheme will be significantly different in the actual experiment. They can be edited through the library `stgBmon_constants.ctl` and the panel will automate all changes.

Appendix A

Datapoints for the Micromegas gas monitoring

This appendix is dedicated to the documentation of every parameter monitored by the Micromegas Gas System project. The reader must note that the Gas Control Group publishes via DIP many more datapoints than actually required by the monitoring project to operate. Some examples are CANbus states, profibusStates or generally, system states that are not monitored in ATLAS DCS. At the same time, all datapoints involved with the analysis module are also not monitored by the DCS. This appendix will only list the ones required or used by the monitoring project.

A.1 Datapoints in the Information Server

Data is published in the Information Server in three main Datapoint Types: `DipFlag`, `DipFloat` and `DipInteger`. The publishing follows the following conventions:

1. Each datapoint starts with the prefix `gas_MMG`.
2. After that follows the module, for example `_Analysis` or `_Mixer`
3. Then, follows the name of the physical component being monitored, i.e.: `_InPressure`
4. Finally, the name of the component tag doing the measurement is attached, i.e. `_PT9215`

The datapoint element that holds the actual value of the parameter is either `.active` for the `DipFlag` type or `.value` for the `DipFloat` and `DipInteger` types. Table A.1 shows a list of the `bool` (`DipFlag`) datapoints in the Information Server:

Variable Name	Description
<pre>_GasSystem_Status _GSY0505	Active status code of the entire gas system.
<pre>_Mixer_Interlock _ZSH9119	Interlock flag of the mixer module.

Table A.1: Datapoints of type `DipFlag` published in the Information Server.

Table A.2 shows a list of the `float` (`DipFloat`) datapoints used by the project:

Variable Name	Description
<pre>_Distribution_Rack<x> _Channel<y>_InFlow _FE<x>02Ch<y>	Input flow measurement of sector <y> gas channel in Rack <x>.
<pre>_Distribution_Rack<x> _Channel<y>_OutFlow _FE<x>06Ch<y>	Output flow measurement of sector <y> gas channel in Rack <x>.
<pre>_Distribution _Rack<x> _InPressure_PT<x>24	Input pressure of Rack <x>.
<pre>_Distribution _TotalInputFlow_TotFlowAS	Total input flow of the distribution module.
<pre>_Mixer _Line<i>InputPressure _PT1<i>03	Input pressure of line <i> of the mixer module.
<pre>_Mixer _Line<i>LFmfcFeedback _XFMFC1<i>06FIF	Low gas flow measured by the mass flow controller on the feedback valve of line <i> of the mixer module.
<pre>_Mixer _Line<i>Ratio _L<i>CompRatioAS	Composition ratio of the line <i> input gas.
<pre>_Mixer _OutPressure_PT1009	Output pressure of the mixer module.
<pre>_Mixer _TotalFlow_TotalFlowAS	Total gas flow from the mixer module.
<pre>_Mixer _Volume_TotalVolAS	Total volume of mixing buffer.

Table A.2: Datapoints of type `DipFloat` published in the Information Server.

Table A.3 shows a list of the `int` (`DipInteger`) datapoints used by the project:

Variable Name	Description
<pre>_Distribution_Rack<x> _State_Rack<x>StepWS	State code of Rack <x>.
<pre>_Distribution_State _ModStepWS	State code of the distribution module.
<pre>_GasSystem_State _GsStepWS	State code of the entire gas system.
<pre>_Mixer_State _StepperWS	State code of the mixer module.
<pre>_PlcCounter_AliveCounter	Rising counter of the Micromegas PLC. Indicates activity.

Table A.3: Datapoints of type `DipInteger` published in the Information Server.

The above state codes of each module are very important for defining the FSM states of each module in the Control System as well. They can be found in [113].

A.2 Datapoints in the monitoring project

The monitoring project uses the framework datapoint types `fwAtlasGas<module>`. Firstly, in `fwAtlasGasChannel` the datapoints are listed as `<pre>_Distribution_Rack<x>_Channel<i>`. The datapoint elements of these datapoints that are used are listed below in Table A.4:

Variable Name	Type	Description
`InFlow`	`FLOAT`	Gas input flow of the channel.
`OutFlow`	`FLOAT`	Gas output flow of the channel.
`quality.uncertain`	`BOOL`	DIP uncertain quality bit.
`quality.bad`	`BOOL`	DIP bad quality bit.
`quality.invalid`	`BOOL`	DIP invalid quality bit.
`UserDefined.GasLoss`	`FLOAT`	Gas loss of the detector: |`InFlow-OutFlow`|.

Table A.4: Datapoint Elements under `<pre>_Distribution_Rack<x>_Channel<i>` of the datapoint type `fwAtlasGasChannel`.

The datapoint type `fwAtlasGasDistribution` hosts the datapoint `<pre>_Distribution` with the datapoint elements listed in Table A.5:

Variable Name	Type	Description
`State`	`INT`	State code of the distribution module.
`TotalInputFlow`	`FLOAT`	Total input flow of the distribution module.
`quality.uncertain`	`BOOL`	DIP uncertain quality bit.
`quality.bad`	`BOOL`	DIP bad quality bit.
`quality.invalid`	`BOOL`	DIP invalid quality bit.

Table A.5: Datapoint Elements under `<pre>_Distribution_` of the datapoint type `fwAtlasGasDistribution`.

The datapoint type `fwAtlasGasMixer` hosts the datapoint `<pre>_Mixer` with the datapoint elements listed in Table A.6:

Variable Name	Type	Description
`State`	`INT`	State code of the mixer module.
`Interlock`	`BOOL`	Interlock flag of the mixer module.
`OutPressure`	`FLOAT`	Output pressure of the mixer module.
`TotalFlow`	`FLOAT`	Total gas flow from the mixer module.
`TotalVol`	`FLOAT`	Total volume of the mixing buffer.
`Line<i>InputPressure`	`FLOAT`	Input pressure of line `<i>`.
`Line<i>LFmfcFeedback`	`FLOAT`	Low gas flow measured by the mass flow controller on the feedback valve of line `<i>`.
`Line<i>Ratio`	`FLOAT`	Composition ratio of the line `<i>` input gas.
`quality.uncertain`	`BOOL`	DIP uncertain quality bit.
`quality.bad`	`BOOL`	DIP bad quality bit.
`quality.invalid`	`BOOL`	DIP invalid quality bit.

Table A.6: Datapoint Elements under `<pre>_Mixer_` of the datapoint type `fwAtlasGasMixer`.

The datapoint type `fwAtlasGasGasSystem` hosts the datapoint `<pre>_GasSystem` with the datapoint elements listed in Table A.7:

Variable Name	Type	Description
State	INT	State code of the Micromegas gas system.
Status	BOOL	Status code of the Micromegas gas system.
quality.uncertain	BOOL	DIP uncertain quality bit.
quality.bad	BOOL	DIP bad quality bit.
quality.invalid	BOOL	DIP invalid quality bit.

Table A.7: Datapoint Elements under `<pre>_GasSystem_` of the datapoint type `fwAtlasGasGasSystem`.

The datapoint type `fwAtlasGasPlcCounter` hosts the datapoint `<pre>_PlcCounter` with the datapoint elements listed in Table A.8:

Variable Name	Type	Description
AliveCounter	INT	Rising counter of the Micromegas PLC. Indicates activity.
quality.uncertain	BOOL	DIP uncertain quality bit.
quality.bad	BOOL	DIP bad quality bit.
quality.invalid	BOOL	DIP invalid quality bit.

Table A.8: Datapoint Elements under `<pre>_PlcCounter_` of the datapoint type `fwAtlasGasPlcCounter`.

The datapoint type `fwAtlasGasRack` hosts the datapoints `<pre>_Distribution_Rack<x>` with the datapoint elements listed in Table A.9:

Variable Name	Type	Description
State	INT	State code of the distribution module.
InPressure	FLOAT	Input pressure of the gas rack.
quality.uncertain	BOOL	DIP uncertain quality bit.
quality.bad	BOOL	DIP bad quality bit.
quality.invalid	BOOL	DIP invalid quality bit.

Table A.9: Datapoint Elements under `<pre>_Distribution_Rack<x>` of the datapoint type `fwAtlasGasRack`.

Finally, Table A.10 lists two datapoint elements from the datapoints `GasProjectMonitor` in `fwAtlasGasProjMonitor`. They are updated by `fwAtlasGas_copyGasData.ctl` script:

Variable Name	Type	Description
AliveChecking.GCSAlive	BOOL	Flag that checks if Gas Control System is reachable.
CopyScript.Running	BOOL	Flag that checks if `fwAtlasGas_copyGasData.ctl` is running.

Table A.10: Datapoint Elements under `GasProjectMonitor` of the datapoint type `fwAtlasGasProjMonitor`.

Appendix B

The states and statuses of MMG Gas FSM

In this appendix, all states and statuses of the Micromegas Gas Finite State Machine hierarchy will be listed, together with their actions and logic.

States are defined by the GEDI interface "Device Editor and Navigator" where the developer can design Device Unit Types and develop their states. These types are provided for ATLAS Gas Systems by the framework. Table B.1 shows the states and actions of the gas mixer device unit:

State	Actions
UNKNOWN	REFRESH, MASK_BAD, UNMASK_BAD
STOPPED	REFRESH, MASK_BAD, UNMASK_BAD
STARTING	REFRESH, MASK_BAD, UNMASK_BAD
RUNNING	REFRESH, MASK_BAD, UNMASK_BAD
INTERLOCK	REFRESH, MASK_BAD, UNMASK_BAD
OTHER	REFRESH, MASK_BAD, UNMASK_BAD
FILL	REFRESH, MASK_BAD, UNMASK_BAD
PURGE	REFRESH, MASK_BAD, UNMASK_BAD
DIRECT	REFRESH, MASK_BAD, UNMASK_BAD

Table B.1: States and actions of `fwAtlasGasMixerDU` FSM node.

Then, Table B.2 shows the states and actions of both the distribution and the rack device units, since they both share the same states and actions:

State	Actions
UNKNOWN	REFRESH, MASK_BAD, UNMASK_BAD
STOPPED	REFRESH, MASK_BAD, UNMASK_BAD
STARTING	REFRESH, MASK_BAD, UNMASK_BAD
PURGE	REFRESH, MASK_BAD, UNMASK_BAD
NOT_READY	REFRESH, MASK_BAD, UNMASK_BAD
RUN_READY	REFRESH, MASK_BAD, UNMASK_BAD
OTHER	REFRESH, MASK_BAD, UNMASK_BAD

Table B.2: States and actions of `fwAtlasGasDistribution` and `fwAtlasGasRackDU` FSM nodes.

Finally, in the lowest part of the hierarchy, the states and actions of the channel device units are shown in Table B.3:

State	Actions
UNKNOWN	REFRESH, MASK_BAD, UNMASK_BAD
OPEN	REFRESH, MASK_BAD, UNMASK_BAD, DECLARE_CLOSED
CLOSED	REFRESH, MASK_BAD, UNMASK_BAD, DECLARE_OPEN

Table B.3: States and actions of `fwAtlasGasChannelDU` FSM node.

The states of the Device Units in the FSM tree influence the states of Logical Units, their parent node. The rack Logical Unit states can be seen in Table B.4:

State	Actions
UNKNOWN	REFRESH
STARTING	REFRESH
READY	REFRESH
ON_PART	REFRESH
OFF	REFRESH
SHUTDOWN	REFRESH
NOT_READY	REFRESH

Table B.4: States and actions of `ATLAS_GAS_RACK` Logical Unit FSM node.

Next, Table B.5 lists the states of the distribution module:

State	Actions
UNKNOWN	REFRESH
STARTING	REFRESH
READY	REFRESH
SHUTDOWN	REFRESH
NOT_READY	REFRESH

Table B.5: States and actions of `ATLAS_GAS_DISTRIBUTION` Logical Unit FSM node.

Finally, the gas system states can be seen in Table B.6:

State	Actions
UNKNOWN	REFRESH
TRANSITION	REFRESH
READY	REFRESH
SHUTDOWN	REFRESH
NOT_READY	REFRESH

Table B.6: States and actions of `ATLAS_GAS_SYSTEM` Logical Unit FSM node.

The states of the Logical Units above are passive and by that it is meant that they change only by calculating the states of their children. The only available action for these modules is

the REFRESH action. Rack states are influenced by the rack distribution module device unit and the channel children. At the same time, the distribution does not need to know the states of the channels under each rack. It can calculate it's states from the Logical Unit of each rack and the distribution device unit.

The statuses in the FSM tree are defined and designed as "states" in the developing environment "Device Editor and Navigator" but are handled differently by the libraries. Statuses are Logical Units and the FSM tree utilizes two: the ATLAS_DU_STATUS for Device Units and ATLAS_STATUS for everything else. The statuses of Device Units are shown in Table B.7:

Status	Actions
UNINITIALIZED	GOTO_OK, GOTO_WARNING, GOTO_ERROR, GOTO_FATAL
OK	GOTO_WARNING, GOTO_ERROR, GOTO_FATAL, GOTO_UNINITIALIZED
WARNING	GOTO_OK, GOTO_ERROR, GOTO_FATAL, GOTO_UNINITIALIZED
ERROR	GOTO_OK, GOTO_WARNING, GOTO_FATAL, GOTO_UNINITIALIZED
FATAL	GOTO_OK, GOTO_WARNING, GOTO_ERROR, GOTO_UNINITIALIZED

Table B.7: Statuses of ATLAS_DU_STATUS Logical Unit.

The statuses of modules other than the Device Units have the statuses of Table B.7 but no actions. They remain passive just like the states of the same Logical Unit. Again, their statuses are calculated from the statuses of their children.

B.1 The SMI++ language

The calculation of statuses and states in the FSM tree is brought off with scripts developed in the SMI++ language. This framework was developed based on the State Manager concept of the DELPHI experiment at CERN in collaboration with the CERN Computing Division [114]. The purpose of this framework is to help developers to design Finite State Machine transitions and cope with the complexity of system states in distributed machines.

Below can be seen a section of the script used to define the state NOT_READY in the Logical Unit ATLAS_GAS_RACK:

```
<other states>
state: NOT_READY
!color: FwStateOKNotPhysics
    when(($ANY $FwCHILDREN in_state UNKNOWN) or
      ($ANY $FwCHILDREN in_state DEAD)) move_to UNKNOWN
    when($ANY $FwCHILDREN in_state STARTING) move_to STARTING
    when(($ALL $fwAtlasGasRackDU in_state RUN_READY) and
      ($ALL $fwAtlasGasChannelDU in_state OPEN)) move_to READY
<other transitions>
<other states>
```

Some syntax rules of the SMI++ scripts can be found in [114] but it is not important to explain here as the keywords used above are self explanatory: In order for the Rack module to move from the state NOT_READY to the state READY, all gas channel children must be in the state OPEN and the rack module must be in the state RUN_READY.

SMI++ provides complete automation on the large control systems of LHC experiments. The error-recovery mechanism used by this framework is bottom up, which means that every object reacts in events asynchronously and in a decentralized way. This design is thus inherently scalable and does not produce a bottleneck from central examination of all states in a system.

Appendix C

Physics of Hall Effect Sensors

Let us assume a metal plate of width w, thickness d and length L with the x axis parallel to its length as shown in Figure C.1. Let us also assume a magnetic field $\vec{B}$ pointing towards the z axis. If current I is flowing through the metal plate towards x direction, the Hall effect can be understood by basic principles of electromagnetism.

The Lorentz force exerted on a charged particle is:

$$\vec{F} = q_0\vec{E} + q_0\vec{v} \times \vec{B} \tag{C.1}$$

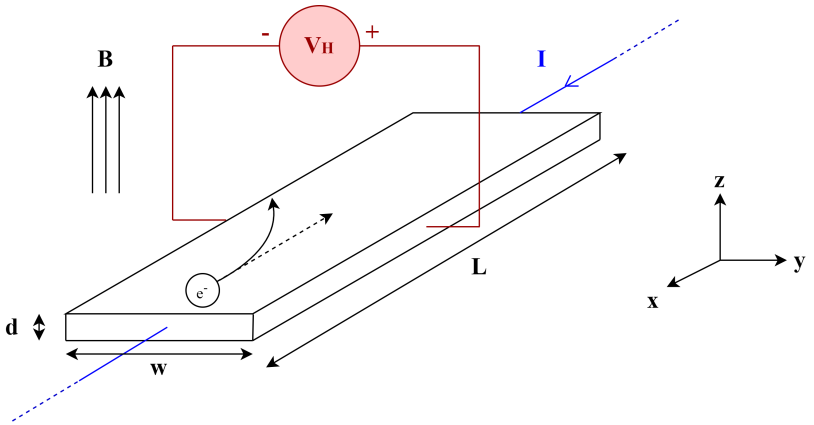


Figure C.1: Representation of the Hall Voltage in a metal plate with current flowing in the x direction.

In this example, $\vec{E} = (E_x, 0, 0)$, $\vec{v} = (v_x, 0, 0)$ and $\vec{B} = (0, 0, B_z)$ and thus C.1 becomes:

$$\vec{F} = q_0 E_x \hat{x} - q_0 v_x B_z \hat{y} \tag{C.2}$$

Equation C.2 shows two effects happening at the same time: charge carriers are forced both to the x direction by V_x and to the y direction by $\vec{B}$. Although the magnetic field forces them to the one side of the plate, this process is self-limiting, because the excess concentration of charges to one side and consequent depletion on the other gives rise to an electric field E_H accros the transducer. This field causes the carriers to try to redistribute themselves evenly and simultaneously gives rise to a voltage V_H that can be measured across the plate. Thus at the y direction after equilibrium:

$$q_0 E_H - q_0 v_x B_z = 0 \tag{C.3}$$

and therefore

$$E_H = v_x B_z \tag{C.4}$$

Integrating the Hall electric field over w, assuming it is uniform, gives us the Hall Voltage:

$$V_H = w v_x B_z \tag{C.5}$$

If n is the charge carrier density and wd the cross-sectional area, the current can be computed as:

$$I = ndw(-v_x)(-e) \tag{C.6}$$

Plugging C.6 into C.5 produces:

$$V_H = \frac{IB_z}{nde} \tag{C.7}$$

For the complete measurement, a module with three plates in x, y and z direction can therefore measure all components of a given magnetic field in space. [115]

Appendix D

Digital calculations for the B-sensor

D.1 B-sensor bit mask

The MDT-DCS Module has two inputs for B-sensor and up four B-sensor can be connected (in pairs). Lets assume a four bit word that represents the positions occupied by B-sensors. We apply the following rules:

- The first bit represents input 0 (sensor 0).
- The second bit represents input 1 (sensor 1).
- Bit three represents the pair of the sensor connected in input 0 (sensor 2).
- Bit four similarly represents the pair of the sensor connected in input 1 (sensor 3).

For four sensors connected in two inputs, the permutations allowed are $2^4 = 16$, so 16 bit masks in total. The first mask with an MDM with no sensors connected is 0000. Each next mask is described by adding one bit. Every possible permutation is displayed in Table D.1:

Bit form	Hex form	Description
0000	`0x0`	No sensors connected
0001	`0x1`	1 sensor at input 0
0010	`0x2`	1 sensor at input 1
0011	`0x3`	2 sensors, 1 at input 0 and 1 at input 1
0100	`0x4`	*Not possible*
0101	`0x5`	2 sensors at input 0
0110	`0x6`	*Not possible*
0111	`0x7`	3 sensors, 2 at input 0 and 1 at input 1
1000	`0x8`	*Not possible*
1001	`0x9`	*Not possible*
1010	`0xA`	2 sensors at input 1
1011	`0xB`	3 sensors, 1 at input 0 and 2 at input 1
1100	`0xC`	*Not possible*
1101	`0xD`	*Not possible*
1110	`0xE`	*Not possible*
1111	`0xF`	4 sensors, 2 at input 0 and 2 at input 1

Table D.1: B-sensor masks in bit and hex forms

The *Not possible* masks are situations where a B-sensor is connected in the 'pair' position of the IDC cable of an MDM input. Lets assume sensor 0 and sensor 2 of an IDC cable connected in input 0 of an MDM.

- If sensor 0 is disconnected and pins 7 and 8 are indeed swapped to read sensor 2 as well, this will result in a short circuit in the SPI interface and sensor 2 will not be read.
- If sensor 0 is disconnected and pins 7 and 8 are not swapped, this case degenerates to sensor 2 becoming sensor 0.

Setting the B-sensor mask at the *Not possible* situations is of course permitted by the firmware but no actual data acquisition will take place for those cases. Any other type of message for the bit mask will result in data taking from the approprate sensors, for example for four sensors:

Host → MDT-DCS module

COB-ID	Data Byte 0
2800h	0xF

D.2 ADC signed data conversion

In order to explain the 24-bit signed conversion of each of the H-sensor's readout, let's assume a TPDO message from an H-sensor that looks like below:

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-4
480h + *0x6F*	0x02	0x29	0x1C0300

The data seen above, translates as follows:

- 0x6F = 111 is the Node ID of the CANbus
- Data Byte 0 displays the channel, and thus 0x02 is channel 2 which is H3 of B-sensor 0
- Data Byte 1 displays the ADC-config and thus 0x29 = 00101001
- Data Bytes 2-4 are the ADC-value in hex which is 0x1C0300 = 1835776

To further elaborate the ADC-config lets look at the bit word 00101001:

BIT	7	6	5	4	3	2	1	0
Meaning	Error	W2	W1	W0	G2	G1	G0	U/B
Value	0	0	1	0	1	0	0	1

This is an example where the data is Bipolar, Word Rate Conversion is 010 and Gain is 100. The conversion word rate is translated as follows from [105]:

Bit word	000	001	010	011	100	101	110	111
Word Rate (Hz)	15.0	30.0	61.6	84.5	101.1	1.88	3.76	7.51

Similarly, gain can be extracted from the bit word using the values from [105]:

Bit word	000	001	010	011	100	101
Gain (V)	0.100	0.055	0.025	1	5	2.5

To calculate the conversion of the signed ADC-value, the 24-bit ADC value has to be extended to 32-bit signed value that the computer reads. If the 23$^{\text{rd}}$ bit is 1, that means the value is actually negative. In this example:

$$\texttt{0x1C0300} \rightarrow \texttt{0x001C0300} = \texttt{00000000}\mathbf{0}\texttt{011100000001100000000} = \texttt{1835776}$$

In this example, the number is positive. Another example would be `0xFFFFFF` which is extended as `0xFFFFFFFF` and thus it represents the number `-1`:

$$\texttt{0x00FFFFFF} \rightarrow \texttt{0xFFFFFFFF} = \texttt{11111111}\mathbf{1}\texttt{11111111111111111111111} = -1$$

Appendix E

Map of the MDT-DCS Modules in New Small Wheel

It is known by now to the reader that the total amount of MDT-DCS Modules used by the New Small Wheel are 64. In addition to implementing the readout of B-sensors, the MDM is also providing readout of the T-sensors for New Small Wheel. However, T-sensors are required in all four detectors in a combined NSW sector, including both small and large sectors.

For this particular reason, referencing of each MDM can be a bit tricky, since the modules are planted in groups of four behind each large sector. In the end, the convention used, is to refer to the sector number and detector technology. The complete map of all modules was created and is shown in Figures E.1 and E.2 as a way to simplify display of information:

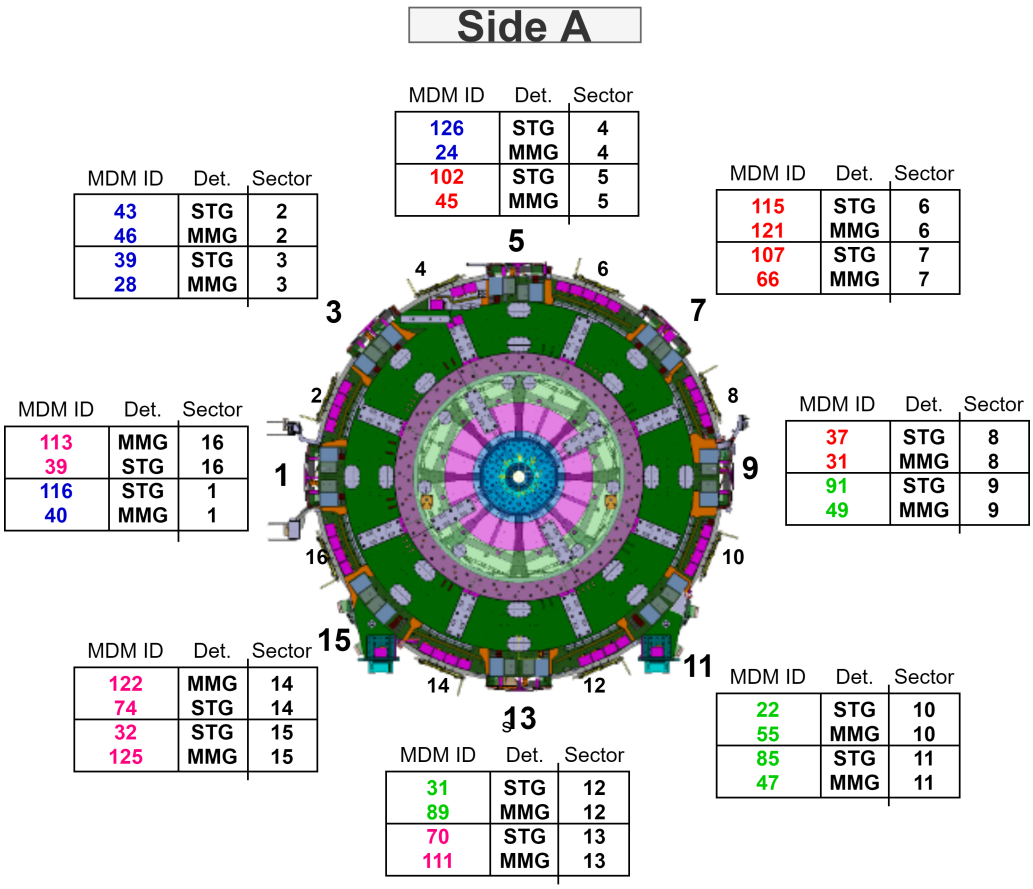


Figure E.1

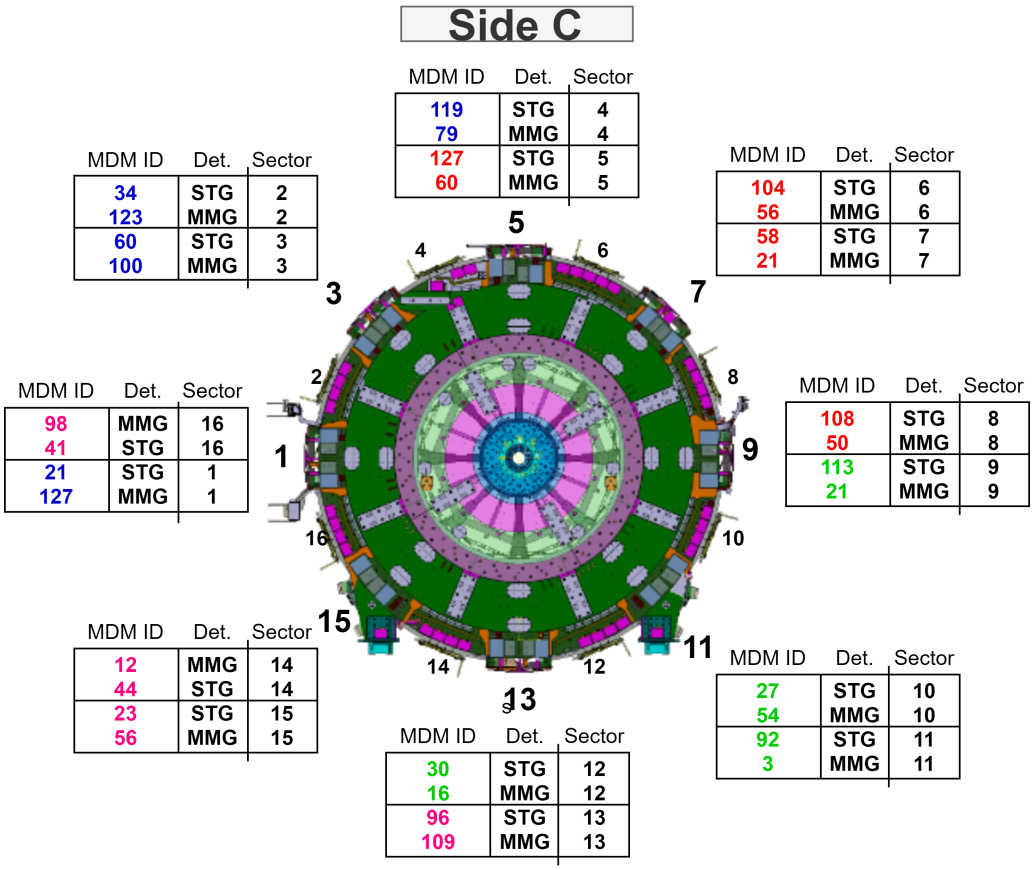


Figure E.2

The above pictures show tables on each large sector representing every MDM installed behind it. They are placed in the table in their side on spoke order. The ID, detector technology and serving sector are noted as well. Finally, different colors on the MDM ID denote different CANbus daisy chains.

Appendix F

Datapoints for the magnetic field monitoring

This appendix is dedicated to the documentation of every parameter monitored by the WinCC OA panel `stgBmon.xml`, the main panel for magnetic field monitoring. Table F.1 shows the top structure of the Datapoints:

Variable Name	Type	Description
`NSWMDM_BField`	`DPT`	Contains the sector structure for magnetic field monitoring.
`EIZ4<side><sector>`	`DP`	Datapoint that includes data for a specific sector.
`BcalAverage`	`FLOAT`	Average value of `Bcal` on a sector.
`node<num>`	`DP_STRUCT`	Node number (1 through 4) which corresponds to side on spoke.

Table F.1: Datapoint structure of each NSW Large Sector.

Each node also contains other Datapoint Elements. They are shown in Table F.2:

Variable Name	Type	Description
`id`	`INT`	MDT-DCS Module Node ID.
`canbus`	`STRING`	The CANbus this node belongs.
`busid`	`STRING`	The ID of the CANbus this node belongs.
`state`	`INT`	MDT-DCS Module integer bit state.
`serial`	`DP_STRUCT`	Contains serial number of the node.
`serialnr`	`STRING`	The serial number of the node.
`bf`	`DP_STRUCT`	Contains all the information about the B-sensors.

Table F.2: Datapoint Elements under the `node DP_STRUCT`.

Each node contains under `bf` all the B-sensor information in another structure. It is shown in Table F.3:

Variable Name	Type	Description
bfMask	UINT	Bit mask of the B-sensors.
state	UINT	Overall state of the B-sensors, calculated by each individual sensor
data	DP_STRUCT	Contains all unsigned ADC values and the ADC config.
adc<i>	UINT	The i ADC field read by the MDM.
conf<i>	BIT32	The i conf field corresponding to adc<i>.
sensor<num>	DP_STRUCT	STRUCT containing all information about the sensor in position num.

Table F.3: Datapoint Elements under bf DP_STRUCT.

Each sensor<num> STRUCT contains all the data and information about the sensor in position num. It is displayed in Table F.4:

Variable Name	Type	Description
label	STRING	The sensor position on sector.
name	STRING	The NSW ID given at production.
id	STRING	The 8 byte sensor ID (combination of idhigh and idlow).
idhigh	INT	First 4 bytes of the sensor ID (subindex 01 in OD).
idlow	INT	Last 4 bytes of the sensor ID (subindex 02 in OD).
h<1, 2, 3>	INT	Signed integer conversion of the respective ADC field.
temp	FLOAT	The temperature sensor reading in °C.
B<x, y, z>	FLOAT	Magnetic Field value in the x, y, z direction.
Bcal	FLOAT	Calibrated value of the magnetic field.
state	INT	B-sensor readout state.

Table F.4: Datapoint Elements under sensor<num> DP_STRUCT.

Bibliography

[1] CERN (2021), ***CERN Annual report 2021*** [DOI: 10.17181/AnnualReport2021]

[2] Counsil of Member States, ***Convention of the establishment of a European Organization for Nuclear Research*** [WEB: CERN Counsil website] fcr

[3] CERN, ***The birth of the Web*** [WEB: CERN website]

[4] Haidt Dieter (2004), ***The discovery of neutral currents*** [DOI: 10.1140/epjc/s2004-01763-y]

[5] R. Cashmore, L. Maiani, Jean-Pierre Revol (2003), ***Prestigious Discoveries at CERN*** [DOI: 10.1007/978-3-662-12779-7]

[6] Salvatore Mele (2015), ***The Measurement of the Number of Light Neutrino Species at LEP*** [DOI: 10.1142/9789814644150_0004]

[7] CERN, ***Antimatter*** [WEB: CERN website]

[8] G. Baur, G.Boero, W. Eyrich, G.Schepers, R. S. Simon et al (1996), ***Production of antihydrogen*** [DOI: 10.1016/0370-2693(96)00005-6]

[9] NA48 Collaboration (1999), ***A new measurement of direct CP violation in two pion decays of the neutral kaon*** [DOI: 10.1016/S0370-2693(99)01030-8]

[10] CERN (2000), ***New State of Matter created at CERN*** [WEB: CERN website]

[11] Eugenie Samuel Reich (2010), ***Antimatter held for questioning*** [DOI: 10.1038/468355a]

[12] The ALPHA Collaboration (2011), ***Confinement of antihydrogen for 1000 seconds*** [DOI: 10.1038/news.2011.349]

[13] ATLAS Collaboration (2012), ***Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at LHC*** [DOI: 10.1016/j.physletb.2012.08.020]

[14] CMS Collaboration (2012), ***Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*** [DOI: 10.1016/j.physletb.2012.08.021]

[15] CERN, ***The Large Hadron Collider*** [WEB: CERN website]

[16] CERN, ***Pulling together: Superconducting electromagnets*** [WEB: CERN website]

[17] CERN, ***Accelerating: Radiofrequency cavities*** [WEB: CERN website]

[18] G. Apollinari, I. Bejar Alonso, O. Bruning, P. Fessia, M. Lamont, L. Rossi, L. Tavian (2017), ***High-Luminosity Large Hadron Collider (HL-LHC) Technical Design Report*** [DOI: 10.23731/CYRM-2020-0010]

[19] CERN, ***High-Luminosity LHC*** [WEB: CERN website]

[20] Glen Elert, ***The Standard Model*** [WEB: The Physics Hypertextbook website]

[21] Encyclopedia Britannica, ***quark*** [WEB: Britannica website]

[22] Encyclopedia Britannica, ***gluon*** [WEB: Britannica website]

[23] F. Englert, R. Brout (1964), ***Broken Symmetry and the Mass of Gauge Vector Mesons*** [DOI: 10.1103/PhysRevLett.13.321]

[24] Peter W. Higgs (1964), ***Broken Symmetries and the Masses of Gauge Bosons*** [DOI: 10.1103/PhysRevLett.13.508]

[25] G. S. Guralnik, C. R. Hagen, T. W. B. Kibble (1964), ***Gobal Conservation Laws and Massless Particles*** [DOI: 10.1103/PhysRevLett.13.585]

[26] W. Herr, B. Muratori, ***Concept of luminosity - CERN Accelerator School, Intermediate Accelerator Physics*** [DOI: 10.5170/CERN-2006-002]

[27] CMS Collaboration, ***What do we mean by 'Cross Section' in particle physics?***, [WEB: CERN website]

[28] The University of New South Wales, ***The Planck scale: relativity meets quantum mechanics meets gravity*** [WEB: New South Wales University website]

[29] B. Bajc, P. Fileviez Perez, G. Senjanovic (2002), ***Proton decay in minimal supersymmetric SU(5)*** [arXiv: hep-ph/0204311]

[30] CERN, ***Supersymmetry*** [WEB: CERN website]

[31] A. Salam, ***Weak and Electromagnetic interactions*** [DOI: 10.1142/9789812795915_0034]

[32] I. N. Levine (1991), ***Quantum Chemistry*** [Chapter 7: Parity]

[33] D. V. Schroeder, M. E. Peskin (1997), ***An introduction to Qunatum Field Theory*** [Chapter 3: Discrete Symmetries of the Dirac Theory]

[34] R.D. Peccei, H. Quinn (1977), ***CP Conservation in the Presence of Pseudoparticles*** [DOI: 10.1103/PhysRevLett.38.1440]

[35] V. Trimble (1987), ***Existence and Nature of Dark Matter in the Universe*** [DOI: 10.1146/annurev.aa.25.090187.002233]

[36] CERN, ***The matter-antimatter asymmetry problem*** [WEB: CERN website]

[37] Encyclopedia Britannica, ***muon*** [WEB: Britannica website]

[38] The ATLAS Collaboration (2008), ***The ATLAS Experiment at the CERN Large Hadron Collider*** [IOP: 2008 JINST 3 S08003]

[39] The ALICE Collaboration (2008), ***The ALICE experiment at the CERN LHC*** [IOP: 2008 JINST 3 S08002]

[40] The LHCb Collaboration (2008), ***The LHCb Detector at the LHC*** [IOP: 2008 JINST 3 S08005]

[41] The CMS Collaboration (2008), ***The CMS experiment at the CERN LHC*** [IOP: 2008 JINST 3 S08004]

[42] CERN, ***The ATLAS Detector*** [WEB: CERN website]

[43] Herman H. J. ten Kate (2008), ***ATLAS Magnet System Nearing Completion*** [IOP: 2008 JINST 3 S08004]

[44] ATLAS Experiment, ***Trigger and Data Acquisition*** [WEB: CERN website]

[45] Cheuk-Yin Wong (1994), ***Introduction to High-Energy Heavy-Ion Collisions*** [Chapter 2: Pseudorapidity Variable]

[46] ATLAS Experiment, ***Muon Spectrometer*** [WEB: CERN website]

[47] K. Einsweiler, L. Pontecorvo, ***Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*** [DOI: 10.17181/CERN.2LBB.4IAL]

[48] D. Pudzha (2020), ***Small-strip thin gap chambers for the muon spectrometer upgrade of the ATLAS experiment*** [IOP: 2020 JINST 15 C09064]

[49] M. Abbrescia, V. Peskov, P. Fonte (2018), ***Resistive Plate Chambers in High Energy Physics Experiments*** [DOI: 10.1002/9783527698691.ch51]

[50] ATLAS Muon Group, M. Livan (1996), ***Monitored Drift Tubes in ATLAS*** [DOI: 10.1016/S0168-9002(96)00851-0]

[51] T. Argyropoulos, K. A. Assamagan, B. H. Benedict, V. Chernyatin et al. (2008), ***Cathode strip chambers in ATLAS: Installation, commissioning and in situ performance*** [DOI: 10.1109/NSSMIC.2008.4774958]

[52] G. Charpak, J. Derre, Y. Giomataris, Ph Rebourgeard (2002) ***Micromegas, a multipurpose gaseous detector*** [DOI: 10.1016/S0168-9002(01)01713-2]

[53] ATLAS Experiment, ***Magnet System*** [WEB: CERN website]

[54] Z. Charifoulline (2006) ***Residual Resistivity Ratio (RRR) Measurements of LHC Superconducting NbTi Cable Strands*** [DOI: 10.1109/TASC.2006.873322]

[55] V. Parma (2015) ***Cryostat Design*** [arXiv: 1501.07154 [physics.acc-ph]]

[56] ATLAS Collaboration (2013), ***ATLAS New Small Wheel Technical Design Report*** [REP. NUMBER: CERN-LHCC-2013-006; ATLAS-TDR-020]

[57] ATLAS Collaboration (2013), *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*, [REP. NUMBER: CERN-LHCC-2013-018; ATLAS-TDR-023]

[58] V. Hedberg (2001), ***The Shielding Project*** [PDF: atlas_shield.pdf]

[59] High Energy Physics Group - Brandeis University, ***New Small Wheel Alignment System*** [WEB: Brandeis University website]

[60] G. Charpak, J. Derre, Y. Giomataris, Ph. Rebourgeard (2002), ***Micromegas, a multipurpose gaseous detector*** [DOI: 10.1016/S0168-9002(01)01713-2]

[61] L. Pezzotti (2020), ***Irradiation and gas studies of Micromegas production chambers for the ATLAS New Small Wheel*** [DOI: 10.22323/1.390.0766]

[62] Glenn F. Knoll ***Radiation Detection and Measurement*** [Chapter 5: Charge Mobility]

[63] Glenn F. Knoll ***Radiation Detection and Measurement*** [Chapter 6: Avalanche Formation]

[64] M. Dris, T. Alexopoulos (2014), ***Signal Formation in Various Detectors*** [arXiv: 1406.3217 [hep-ex]]

[65] P. Gkountoumis (2016), ***Level-1 Data Driver Card of the ATLAS new small wheel upgrade compatible with the phase II 1MHz readout scheme*** [DOI: 10.1109/MOCAST.2016.7495115]

[66] S. Tang, D. Matakias, H. Chen, K. Chen, M.A. Pleier, V. Polychronakos, L. Yao (2021), ***Production and tests of the ADDC for the Micromegas detector of the ATLAS New Small Wheel*** [IOP: 2021 JINST 16 P03021]

[67] G. Iakovidis (2019), ***VMM3a, an ASIC for tracking detectors*** [DOI: 10.1088/1742-6596/1498/1/012051]

[68] B.G. Taylor (1998), ***TTC distribution for LHC detectors*** [DOI: 10.1109/23.682644]

[69] L. Guan (2016), ***Trigger algorithms and electronics for the ATLAS muon New Small Wheel upgrade*** [DOI: 10.1088/1748-0221/11/01/C01083]

[70] W. Wu (2019), ***FELIX: the New Detector Interface for the ATLAS experiment*** [DOI: 10.1109/TNS.2019.2913617]

[71] A. Abusleme, C. Bélanger-Champagne, A. Bellerive, Y. Benhammou, J. Botte, H. Cohen, M. Davies, Y. Du, L. Gauthier, T. Koffas, S. Kuleshov, B. Lefebvre, C. Li, N. Lupu, G. Mikenberg, D. Mori, J. P. Ochoa-Ricoux, E. Perez Codinag, S. Rettie, A. Robichaud-Véronneau, R. Rojas, M. Shoa, V. Smakhtin, B. Stelzer, O. Stelzer-Chilton, A. Toro, H. Torres, P.Ulloa, B. Vachon, G. Vasquez, A.Vdovin, S. Viel, P. Walker, S. Weber, C. Zhuk (2016), ***Performance of a full-size small-strip thin gap chamber prototype for the ATLAS new small wheel muon upgrade*** [DOI: 10.1016/j.nima.2016.01.087]

[72] R. Keyes, K.A. Johnson, L. Pepin, F. Leger, C. Qin, S. Webster, A. Robichaud-Veronneau, C. Belanger-Champagne, B. Lefebvre, S.H. Robertson (2017), ***Development and characterisation of a gas system and its associated slow-control system for an ATLAS small-strip thin gap chamber testing facility*** [IOP: 2017 JINST 12 P04027]

[73] P. Miao, F. Li, L. Guan, I. Ravinovich, S. Zhou, N.J. Zhang, Z.L. Zhang, X.X. Wang, G. Jin (2020), ***The development of the Front-End Boards for the small-strip Thin Gap Chambers detector system of the ATLAS Muon New Small Wheel upgrade*** [IOP: 2020 JINST 15 P11024]

[74] P. Miao et al (2020), ***The development of the Front-End Boards for the small-strip Thin Gap Chambers detector system of the ATLAS Muon New Small Wheel upgrade*** [IOP: 2020 JINST 15 P11024]

[75] P. Gkountoumis (2019), ***LEVEL-1 DATA DRIVER CARD - A high bandwidth radiation tolerant aggregator board for detectors*** [DOI: 10.22323/1.322.0036]

[76] D. Gerbaudo (2013), ***Update ATLAS L1 Muon Trigger with sTGC: Design and Performance*** [DOI: 10.22323/1.180.0094]

[77] X. Hu (2016), ***Design and Test of a Signal Packet Router Prototype for the ATLAS NSW sTGC Detector*** [REP. NUMBER: ATL-MUON-SLIDE-2016-010]

[78] R. M. Coliban, S. Popa, T. Tulbure, D. Nicula, M. Ivanovici, S. Martoiu, L. Levinson, J. Vermeulen (2016), ***The Read Out Controller for the ATLAS New Small Wheel*** [IOP: 2016 JINST 11 C02069]

[79] P. Leitao, S. Feger, D. Porret, S. Baron, K. Wyllie, M. Barros Marin, D. Figueiredo, R. Francisco, J. C. Da Silva, T. Grassi, ***Test bench development for the radiation Hard GBTX ASIC*** [IOP: 2015 JINST 10 C01038]

[80] J. M. Mendez, S. Baron, A. Caratelli, P. V. Leitao (2018), ***New slow-control FPGA IP for GBT based system and status update of the GBT-FPGA project*** [DOI: 10.22323/1.313.0083]

[81] F. Trantou (2022), ***New physics selection in the LHC ATLAS experiment with the upgraded muon detector New Small Wheel*** [DOI: 10.26240/heal.ntua.23130]

[82] T. Vafeiadis (2020), ***Integration and commissioning of ATLAS New Small Wheel Micromegas detectors with electronics at CERN*** [PDF: ICHEP_TheoVafeiadis_f.pdf]

[83] R. Ichimiya, H. Kurashige, T. Maeno, M. Ikeno, Osamu Sasaki (2002), ***Sector logic implementation for the ATLAS endcap level-1 muon trigger*** [DOI: 10.5170/CERN-2002-003.236]

[84] W. Buttinger (2012), ***The ATLAS Level-1 Trigger System*** [DOI: 10.1088/1742-6596/396/1/012010]

[85] CERN, ***ATLAS*** [WEB: CERN website]

[86] SIEMENS, ***SIMATIC WinCC OA Open Architecture Portal*** [WEB: WinCC OA website]

[87] SIEMENS, ***SIMATIC WinCC Open Architecture V3.18*** [PDF: technical-product-description-wincc-oa-v3-18-en.pdf]

[88] SIEMENS, ***SIMATIC WinCC Open Architecture Version 3.18*** [WEB: WinCC OA website]

[89] JCOP FRAMEWORK, ***About JCOP*** [WEB: JCOP Framework website]

[90] O. Holme, M. G. Berges, P. Golonka, S. Schmeling (2005), ***The JCOP Framework*** [REP. NUMBER: CERN-OPEN-2005-027]

[91] JCOP FRAMEWORK, ***JCOP Framework Project Proposal*** [WEB: JCOP Framework website]

[92] A. B. Poy, S. Schlenker (2016), ***ATLAS DCS - FSM Integration Guidelines*** [REP. NUMBER ATL-DQ-ON-0010]

[93] ATLAS Central DCS, ***DCS FSM*** [WEB: CERN Twiki website]

[94] R. Guida, M. Capeans, F. Hahn, S, Haider, B. Mandelli (2013), ***The gas systems for the LHC experiments*** [DOI: 10.1109/NSSMIC.2013.6829415]

[95] OPC FOUNDATION (2008), ***Unified Architecture***, OPC Foundation website]

[96] R. Guida et al. (2012), ***Development of a common gas analysis approach for the gas systems of all the experiments at the CERN Large Hadron Collider*** [DOI: 10.1109/NSSMIC.2012.6551290]

[97] Walter Fabinski (1989), ***Multicomponent infrared gas analyzer*** [PATENT: US5055688A]

[98] T. Alexopoulos, S. Maltezos et al. (2015), ***Gas System for the ATLAS NSW Micromegas Detectors: Design Aspects and Advanced Validation Methods for their QA/QC***, [DOI: 10.12681/hnps.1839]

[99] S. Karentzos (2019), ***Research and Development of the Micromegas Detector for the New Small Wheel upgrade in the ATLAS Experiment*** [DOI: 10.26240/heal.ntua.18630]

[100] B. Copy, E. Mandilara, I. Prieto Barreiro, F. Varela Rodriguez (2018), ***Monitoring of CERN's Data Interchange Protocol (DIP) system*** [DOI: 10.18429/JACoW-ICALEPCS2017-THPHA162]

[101] R. Barillere, F. Mico-Montava, S. Pavis, J. Rochez (2003), ***LHC GCS: A homogeneous approach for the control of the LHC experiments gas systems*** [REP. NUMBER: PAL-PUB-2005-002, pages 57-59]

[102] V. A. Mitsou (2003), ***The ATLAS Transition Radiation Tracker*** [arXiv: hep-ex/0311058]

[103] H. Broterenbrood, B. Hallgren (2003), ***The Embedded Local Monitor Board (ELMB) in the LHC Front-end I/O Control System*** [DOI: 10.5170/CERN-2001-005.325]

[104] H. Broterenbrood (2004), ***CANopen Application Software for the ELMB128*** [PDF: ELMBio.pdf]

[105] H. Broterenbrood (2011), ***MDT-DCS CANopen module*** [PDF: MDT-DCS-CANnode.pdf]

[106] R. Hart, H. Boterenbrood, G. Bobbink, Amsterdam, S. Zimmermann (2009), ***The ATLAS MDT Control System*** [ICALEPCS: TUP079]

[107] H. Broterenbrood (2000), ***CANopen high-level protocol for CAN-bus*** [PDF: CANopen30.pdf]

[108] CAN-in-Automation (CiA) e.V. (2002), ***CANopen, Application Layer and Communication Profile*** [PDF: 301_canopen.pdf]

[109] P. Bishop, N. Huntley (2010), ***Insulation displacemtn connector*** [PATENT: US20120003850A1]

[110] Frederick O. R. Miesterfeld, John M. McCambridge, Ronald E. Fassnacht, Jerry M. Nasiadka (1986), ***Method for serial peripheral interface (SPI) in a serial data bus*** [PATENT: US4739324A]

[111] SYSTEC Electronic, ***sysWORXX USB-CANmodule2*** [WEB: SYSTEC Electronic website

[112] J. Bortfeldt (2018), ***Construction and Test of Full-Size Micromegas Modules for the ATLAS New Small Wheel Upgrade***, [DOI: 10.1051/epjconf/201817401003]

[113] R. Barillere, G. Thomas (2014), ***Specification of the control interface between GCS and DCS***, [INT. NOTE: EDMS-0000000 REV-1.28 VAL-FINAL]

[114] B. Franek, C. Gaspar (2010), ***SMI++ Object Oriented Framework used for Automation and Error Recovery in the LHC Experiments***, [DOI: 10.1088/1742-6596/219/2/022031]

[115] Edward Ramsden (2006), ***Hall-Effect Sensors Theory and Applications*** [Chapter 1: Hall-Effect Physics]