Vertical Slice of the ATLAS Detector Control System

H.Boterenbrood[1], H.J. Burckhart[2], J.Cook[2], V. Filimonov[3], B. Hallgren[2], F.Varela[2a]

[1]NIKHEF, Amsterdam, The Netherlands, [2]CERN, Geneva, Switzerland, [3]PNPI, St.Petersburg, Russia,
[a]also University of Santiago de Compostela, Spain

Abstract

The ATLAS Detector Control System consists of two main components: a distributed supervisor system, running on PCs, called Back-End system, and the different Front-End systems. For the former the commercial Supervisory Control And Data Acquisition system PVSS-II has been selected. As one solution for the latter, a general purpose I/O concentrator called Embedded Local Monitor Board has been developed. This paper describes a full vertical slice of the detector control system, including the interplay between the Embedded Local Monitor Board and PVSS-II. Examples of typical control applications will be given as well.

I. Scope of DCS

The ATLAS Detector Control System (DCS) [1] must enable a coherent and safe operation of the ATLAS detector. It has also to provide interaction with the LHC accelerator and the external services such as cooling, ventilation, electricity distribution, and safety systems. Although the DCS will operate independently from the DAQ system, efficient bi-directional communication between both systems must be ensured. ATLAS consists of several subdetectors, which are operationally quite independent. DCS must be able to operate them in both stand-alone mode and in an integrated fashion as a homogenous experiment.

DCS is not responsible for safety, neither for personal nor for equipment. It also does not deal with the data of the physics events.

II. Architecture of DCS

The ATLAS detector is hierarchically organised, starting with the subdetectors (e.g. Transition Radiation Tracker, Tile Calorimeter, etc.), and on the further levels down following their respective subsystems (e.g. barrel and end-cap parts or High Voltage, Low Voltage, gas systems, etc.). This organisation has to be accommodated in the DCS architecture. The DCS equipment is geographically distributed in three different areas as shown in figure 1. The main control room is situated at the surface, in SCX1 and houses the supervisory stations for the operation of the detector. This equipment is connected via a LAN to the Local Control Stations (LCS) placed in the underground electronics room USA15, which is accessible during running of the experiment. The Front-End (FE) electronics in UX15 is exposed to radiation and a strong magnetic field. This equipment is distributed over the whole volume of the detector with cable distances up to 200 m. The communication with the equipment in USA15 is done via fieldbuses.

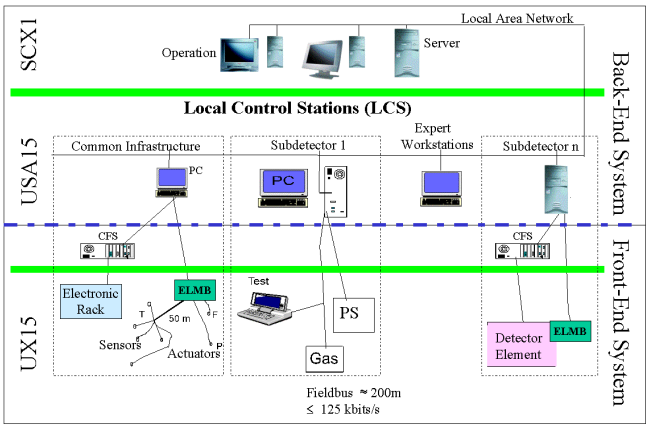

Figure 1: Architecture of ATLAS DCS

A. *Back-End System*

The highest level is the overall supervision as performed from the control room by the operator. Apart from the Human Interface it includes analysis and archiving of monitor data, 'automatic' execution of pre-defined procedures and corrective actions, and exchange of data with systems outside of DCS. The middle level consists of LCSs, which operate a sub-detector or a part of it quite independently. These two levels form the Back-End (BE) system. The commercial Supervisory Control And Data Acquisition (SCADA) package PVSS-II [2] has been chosen, in the framework of the Joint COntrols Project (JCOP) [3] at CERN, to implement the BE systems of the 4 LHC experiments. PVSS-II gathers information from the FE equipment and offers supervisory control functions such as data processing, execution of control procedures, alert handling, trending, archiving and web interface. It has a modular architecture based on functional units called managers, which perform these individual tasks. PVSS-II is a device-oriented product where devices are modelled by structures called data-points. Applications can be distributed over many stations on the network running on both Linux and WNT/2000. These features of modelling and distribution facilitate the mapping of the control system onto the different subdetectors. Due to the large number of channels to be handled in ATLAS, the event-driven architecture of the product was a crucial criterion during the

selection process. PVSS-II also provides a wide set of standards to interface hardware (OPC, fieldbus drivers) and software (ODBC, DDE, DLL, API).

B. *Front-End System*

The responsibility for the FE systems is with the sub-detector groups. In order to minimise development effort and to ease maintenance load, a general purpose I/O system, called Embedded Local Monitor Board (ELMB) has been developed, which is described in detail in another contribution to this workshop [4]. It comprises ADC and digital I/O functions, is radiation tolerant for use outside of the calorimeters of the LHC detectors and can operate in a strong magnetic field. Further functions such as DAC and interlock capability can be added. The readout is done via the fieldbus CAN [5], which is an industry standard with well-supported commercial hardware down to the chip level. Due to its very performent error detection and correction and its flexibility, CAN is particularly suited for distributed I/O as needed by the LHC detectors. CANopen is used as high-level communication protocol on the top of the physical and data link layers defined by CAN. It comprises features such as network management and supervision, a wide range of communication objects for different purposes (e.g. real-time data transfer, configuration) and special functions for network synchronisation, time stamping, error handling, etc.

C. *Connection FE-BE*

The interface PVSS-CANopen is based on the industry standard OPC (OLE for Process Control) [6]. OPC is a middle-ware based on the Microsoft DCOM (Distributed Compound Object Model) which comprises a set of interfaces designed to facilitate the integration of control equipment into Windows applications. OPC is supported by practically all SCADA products. This standard implements a multi-client/multi-server architecture where a server holds the process data or OPC items in the so-called address space and a client may read, write or subscribe to them using different data access mechanisms (synchronous, asynchronous, refresh, or subscribe). An OPC server may organise the items in groups on behalf of the client assigning some common properties (update rate, active, call-back, dead-band, etc.). Another important aspect of OPC is that it transmits data only on change, which results in a substantial reduction of the data traffic.

Several firms offer CANopen OPC servers, but those investigated are based on their own special hardware interface and they support only limited subsets of the CANopen protocol. Although these subsets fulfil most of the industrial requirements, they do not provide all functionality required in high energy physics. Therefore we have developed a CANopen OPC Server supporting the CANopen device profiles required. This package is organised in a part which acts like a driver for CANopen and is specific to the PCI-CAN interface card chosen, and a hardware-independent part which implements all OPC interfaces and main loops handling communication with external applications. This CANopen-OPC server imports from a configuration file all information needed to define its address space, the bus topology and the communication parameters.

III. IMPLEMENTATION OF VERTICAL SLICE

A full “vertical slice” of the ATLAS DCS has been implemented, which ranges from the I/O point (sensor or actuator) up to the operator interface comprising all elements described above, like ELMB, CAN, OPC Server and PVSS-II. The software architecture of the vertical slice is shown in figure 2.

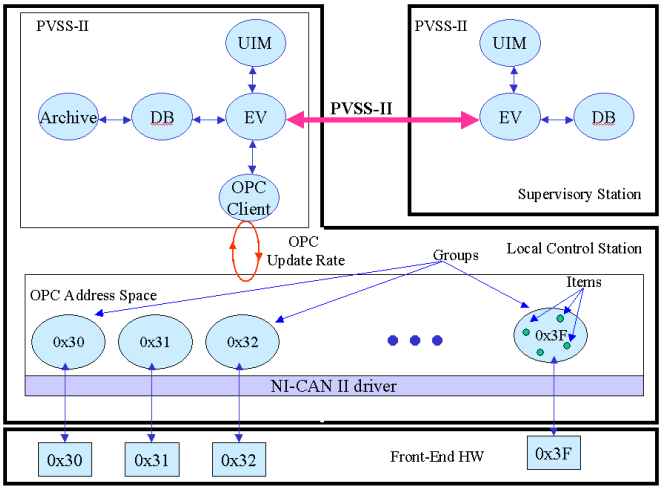

Figure 2: Software Architecture

The system topology in terms of CANbus, ELMBs and sensors is modelled in the PVSS-II database using data-points. These data-points are connected to the items in the CANopen-OPC server address space. Setting a data-point in PVSS-II will trigger the OPC server to send the appropriate CANopen message to the bus. In turn, when an ELMB sends a CANopen message to the bus, the OPC server will decode it, set the respective item in its address space and hence transmit the information to a data-point in PVSS-II. The SCADA application carries out the predefined calculations to convert the raw data to physical units, possibly trigger control procedures, and trend and archive the data. The vertical slice also comprises PVSS-II panels to manage the configuration, settings and status of the bus.

This vertical slice has been the basis for several control applications of ATLAS subdetectors like the alignment systems of the Muon Spectrometer, the cooling system of the Pixel subdetector, the temperature monitoring system of the Liquid Argon subdetector and the calibration of the Tile Calorimeter at a test beam. As an example the latter will be discussed in the next paragraph.

A subset of the Tile Calorimeter modules needs to be calibrated with particles in a test beam. The task of DCS is to monitor and control the three different subsystems, the high voltage, the low voltage and the cooling system. A total of seven ELMBs were connected to the CAN bus. For the low voltage system, the standard functionality of the vertical slice

was easily extended in order to drive analogue output channels by means of off-board DAC chips. This application also interfaced to the CERN SPS accelerator in order to retrieve the beam information for the H8 beam line. Data coming from all subsystems were archived in the PVSS-II historical database and then passed to the Data Acquisition system for event data by means of the DCS-DAQ Communication software (DDC) [7].

Figure 3 shows the PVSS graphical user interface of this application. The states of the devices integrating the DCS are colour-coded and the current readings of the operational parameters are also shown in the panel. Dedicated panels for each subsystem and graphical interfaces to the historical database and for alert handling are also provided. The system has proven to work very reliably.

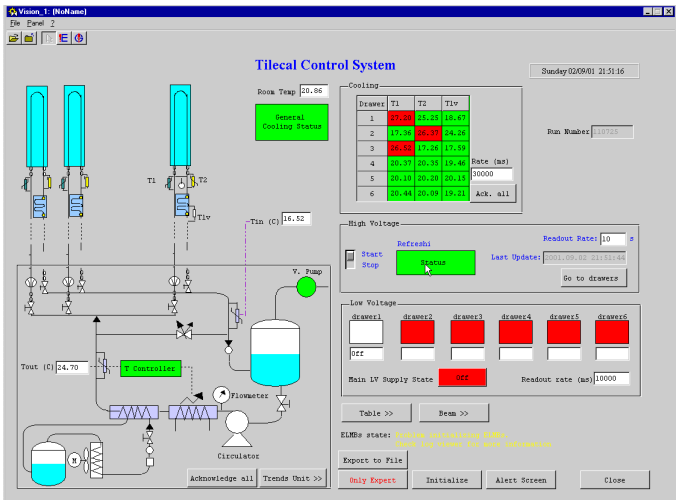

Figure 3: Control Panel for Tile Calorimeter Calibration

IV. CAN BRANCH TEST

Several thousand ELMB nodes will be used in ATLAS, the largest sub-detector comprising alone 1200 nodes. When organising them in CANbuses, conflicting requirements like performance, cost, and operational aspects have to be taken into account. For example, a higher number of ELMBs per branch – the maximum number possible is 127 – reduces the cost, but also reduces performance and increases the operational risk, i.e. in case of failure a bigger fraction of the detector may become in-operational. Additionally, several CAN messages having different priorities may be transferred at the same time. This calls for an efficient design of the bus to minimise the collisions of the frames. The priority is defined by the so-called Communication Object Identifier (COB-ID) in CANopen, which is built from the node identifier and the type of message.

A 200m long CAN branch with 16 ELMBs has been set up in order to measure its performance in terms of data volume and readout speed, and to identify possible limiting elements in the readout chain. The set-up used is shown in figure 4. The ELMBs were powered via the bus using a 9 V power supply. The total current consumption was about 0.4 A.. The total number of channels and their transmission types are given in table 1.

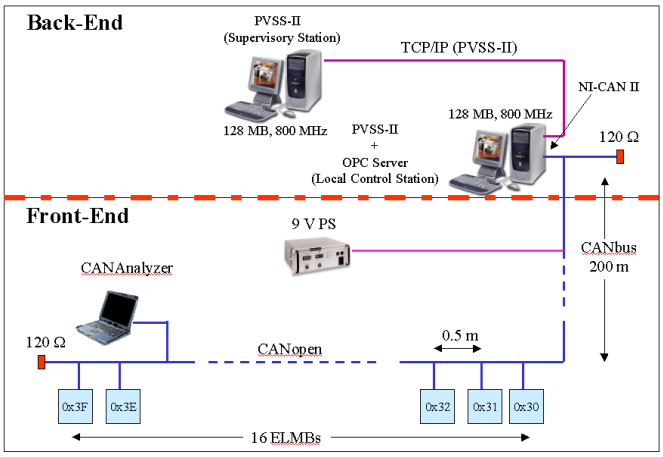

Figure 4: Set-up of CAN branch test

COB-ID	Type	Channels	Mode
0x180 + NodeId	Analogue Input	1024	Sync
0x200 + NodeId	Digital Input	128	Async + Sync
0x280 + NodeId	Digital Output	256	Asyn

Table 1: I/O points of the CAN branch test

Due to the large number of channels, the analogue inputs of the ELMBs were not connected to the sensors. A special version of the ELMB software was used to generate random ADC data ensuring new values at each reading and therefore maximising the traffic through the OPC server. The digital output and input ports were interconnected to check the transmission of CAN messages with different priorities on the bus, i.e. output lines can be set while inputs are being read. Figure 5 shows the bus activity after a CANopen SYNC command is sent to the bus. All ELMBs try to reply to this message at the same time causing collisions of the frames on the bus. The CAN collision arbitration mechanism handles them according to the priority of the messages. In this figure, the Bus Period is defined as the time taken for all synchronous messages to be received from all nodes after the SYNC command has been sent to the bus. δ defines the time between consecutive CAN frames on the bus and is a function of the bus speed, which is limited by the CANbus length (typically 0.7ms at 125kbaud). The time between successive analogue channels from a single ELMB, which is dependent upon the ADC conversion rate, is given by Δ. The OPC Server generates the SYNC command at predefined time intervals and this defines the readout rate.

The SCADA application was distributed over two PCs running WinNT (128 MB of RAM and 800 MHz clock frequency). The hardware was connected to a PC acting as Local Control Station (LCS), where the OPC server and control procedures where running. All values were archived to the PVSS-II historical database. The second PC, acting as operator station, was used to access the database of the LCS

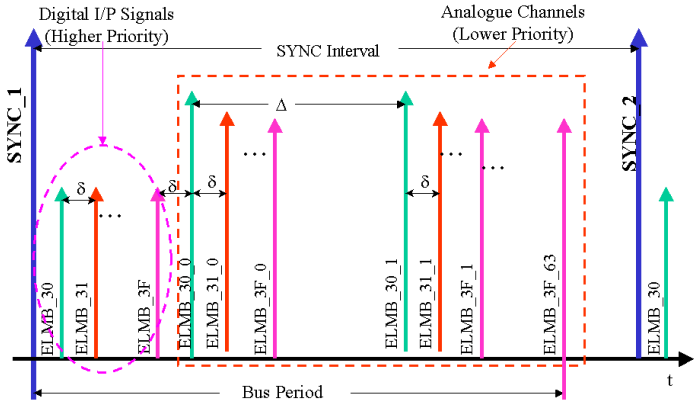

Figure 5: Bus activity after a SYNC

and to perform data analysis and visualisation. The communication between the two systems was internally handled by PVSS-II. A third PC, running as a CAN analyser, was used to log all CAN frames on the bus to a file for later comparison with the values stored in the SCADA database. This CAN analyser is a powerful diagnostic tool. It allows for debugging of the bus enabling visualisation of the traffic and sending of messages directly to the nodes.

The test was performed for different settings of the ADC conversion rate and of the update rate of the OPC server. This parameter defines the polling rate of the internal cache of the OPC server for new data to be transferred to the OPC client. The readout rate was also varied from values much greater than the bus period down to a value close to it. The CPU behaviour was monitored under these sets of conditions.

We have observed excellent performance of the ATLAS DCS vertical slice at low conversion rates (1.88 and 3.71 Hz). All messages transmitted to the bus have been logged in the PVSS historical database. This result is independent of the SYNC interval as long as this parameter is kept above the bus period. However, some ATLAS applications call for a faster readout. Results at 15.1 and 32.5 Hz show a good behaviour when the SYNC interval is higher than the bus period. Performance deteriorates when the SYNC interval tends to the bus period. Crosscheck with the CAN analyser files showed that many messages were not in the PVSS-II database. Two major problems were identified: overflows in the read buffer of the NI-CAN interface card, and the PVSS-II archiving taking close to 100% of the CPU time while the avalanche of analogue channels is on the bus. It was also found that these results are very sensitive to the OPC update rate. The faster the update takes place, the more CPU time is required limiting its availability for other processes like the PVSS archiving. This suggests to split the PVSS application in such a manner that only the OPC interface runs on the LCS while all archiving is handled higher up in the hierarchy shown in figure 4. However, further tests must be performed to address the limitation of each of the individual elements quantitatively.

V. Conclusions and Outlook

PVSS-II has been found to be a good solution to implement the BE system of the ATLAS DCS. It is device oriented and allows for system distribution to aid the direct mapping of the DCS hierarchy. The ELMB I/O module has been shown to fulfil the requirements of the majority of the sub-detectors. Both PVSS-II and the ELMB are well accepted by the ATLAS sub-detector groups. The vertical slice comprises of these two components interconnected via the CANopen OPC Server. Many applications have been developed using this vertical slice and they have shown that it offers high flexibility, good balance of the tasks, reliability and robustness. A full branch test has been performed with the aim of estimating its performance. Good results were obtained for low ADC conversion rates. Tests at higher ADC conversion rates allowed the identification of several problems, such as the read buffer size of the PCI-CAN card causing overflows of CAN messages. For this and other reasons, such as cost and architecture, this interface card will be replaced. CPU usage increases to unacceptable levels with high data flow when the OPC Server and archiving are both run on a single processor. The vertical slice tests have helped to better define the load distribution amongst different PVSS-II systems.

Further tests are required to define the CAN topology to be used in ATLAS. The main issues to be addressed are; the system granularity in terms of number of buses per PC, the number of ELMBs per bus (between 16 and 32 seems to fulfil most requirements) and powering. In addition, bus behaviour needs to be investigated further, e.g. the ELMB may only send data on change. In response to a sync, a status message would be sent giving a bit flag for each channel of the ELMB indicating whether an error had occurred. If values exceed pre-defined acceptable limits, then this could also be signaled by the ELMB. Bus supervision and automatic recovery must also be investigated. It must be possible to reset individual nodes, reset the bus or perform power cycling, depending upon the severity of any error encountered.

VI. References

[1] H.J. Burckhart, “Detector Control System”, Fourth Workshop on Electronics for LHC Experiments, Rome (Italy), September 1998, p. 19-23.
[2] PVSS-II, http://www.pvss.com/
[3] JCOP, http://itcowww.cern.ch/jcop/
[4] B.Hallgren et al., “The Embedded Local Monitor Board (ELMB) in the LHC Front-End I/O Control System”, contribution to this conference.
[5] CAN in Automation (CiA), D-91058 Erlangen (Germany). http://www.can-cia.de/
[6] OLE for Proccess Control, http://www.opcfoundation.org/
[7] H.J. Burckhart et al., “Communication between Trigger/DAQ and DCS”, International Conference on Computing in High Energy and Nuclear Physics, Beijing (China) September 2001, p. 109-112.