Data acquisition and online monitoring software for CBM test beams

J Adamczewski-Musch, N Kurz, S Linev and P Zumbruch

GSI Helmholtzzentrum für Schwerionenforschung, 64291 Darmstadt, Germany

E-mail: j.adamczewski@gsi.de

Abstract. The Compressed Baryonic Matter (CBM) experiment is intended to run at the FAIR facility that is currently being built at GSI in Darmstadt, Germany. For testing of future CBM detector and readout electronics prototypes, several test beam campaigns have been performed at different locations, such as GSI, COSY, and CERN PS. The DAQ software has to treat various data inputs: standard VME modules on the MBS system, and different kinds of FPGA boards, read via USB, Ethernet, or optical links. The Data Acquisition Backbone Core framework (DABC) is able to combine such different data sources with event-builder processes running on regular Linux PCs. DABC can also retrieve the instrumental set up data from EPICS slow control systems and insert it into the event data stream for later analysis. Vice versa, the DIM based DABC control protocol has been integrated to the general CBM EPICS IOC by means of an EPICS-DIM interface. Hence the DAQ can be monitored and steered with a CSS based operator GUI. The CBM online monitoring analysis is based on the GSI Go4 framework which can directly connect to DABC online data via sockets, or process stored data from list-mode files. A Go4 sub-framework has been implemented to provide possibility of parallel development of analysis code for different sub-detectors groups. This allows to divide the Go4 components up into independent software packages that can run either standalone, or together at the beam-time in a full set up.

1. Introduction

The Compressed Baryonic Matter (CBM) collaboration is developing one of the major experiments of the future Facility for Antiproton and Ion Research (FAIR) [1] in Darmstadt, Germany. CBM will investigate high-density nuclear matter as produced by heavy ion collisions at 10-45 AGeV energy range [2]. Scientific topics are the in-medium modifications of hadrons in dense matter, the deconfinement phase transition at high baryon densities, and exotic states of matter such as condensates of strange particles.

The CBM detector will consist of several types of sensor components, such as:

- Silicon pixel detector (“micro vertex detector”, MVD)
- Silicon strip detector (“silicon tracking stations, STS)
- Ring Imaging Cherenkov (RICH) detector
- Transition Radiation Detector (TRD)
- Muon Chamber (MUCH) with Gas Electron Multiplier (GEM)
- Resistive Plate Chamber (RPC)
- Electromagnetic calorimeter (ECAL)

Each of these sub-detectors can be developed independently by specific working groups, optimizing the component performance, and regarding the overall constraints.

The future CBM data acquisition is planned with a "trigger-less" concept, because in worst case the first level trigger decision may need analysis of the full detector information. Since this is not possible with a classical frontend-triggered system, the DAQ will sample and transport all time-stamped frontend data to a computing farm, the first level event selector (FLES) [3]. This concept requires the investigation of special readout hardware and optimization of data transport and networking.

For these developments a number of CBM detector prototypes have been tested under beam irradiation in various facilities, such as GSI, the CERN Proton Synchrotron (PS), or the Cooler Synchrotron (COSY) in Jülich, Germany. First prototypes of trigger-less data acquisition hardware have also been applied, together with conventional trigger-based readouts.

Such heterogeneous test beam set-ups with different detector groups sharing the same infrastructure require a flexible software organization, both for data taking, and for online monitoring. Especially the reusability of existing code is an issue, as same hardware will be used in different test environments with similar structures and algorithms. For this purpose, two software frameworks have been applied at CBM beam tests since 2008: the Data Acquisition Backbone Core (DABC) for readout and event-building, and the Go4 analysis system for online monitoring.

2. Data Acquisition with DABC

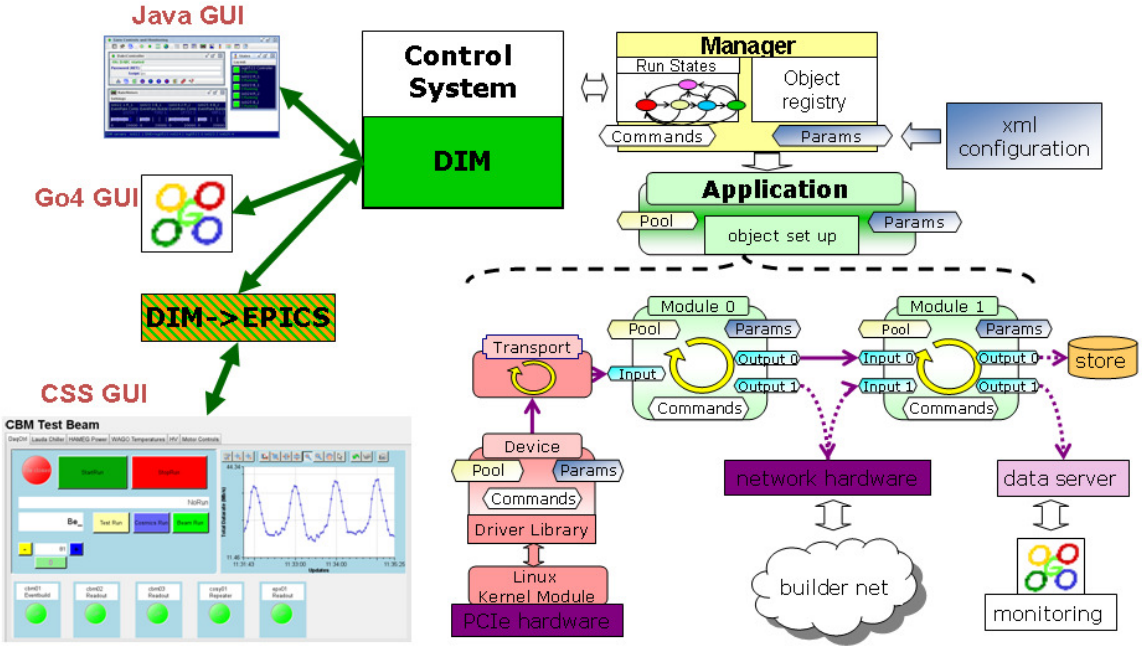

Figure 1. Schematic software object diagram of a DABC node. The user Application is a plug-in that defines the functional objects of the data acquisition, such as i/o devices, worker modules, and transport connections between them. A central Manager provides interfaces to configuration and control systems. The DIM protocol is implemented for monitoring and slow control. It may serve various GUIs. For beam tests, it was integrated to EPICS CSS GUI via a DIM/EPICS interface.

2.1. General features

The Data acquisition Backbone Core (DABC) is a software framework for experiment data acquisition and processing [4][5]. It is based on the C++ programming language and is currently running on Linux operating systems. By default, it supports Ethernet (IP sockets) and InfiniBand (verbs) networks. To

integrate legacy data acquisition hardware, DABC also implements data formats and connections of the established DAQ framework MBS [6]. User-defined code for experiment specific hardware readout and data formats can be added to the framework libraries by means of a plug-in architecture [7].

Figure 1 shows software objects and their relationship in a typical DABC node. A central *Manager* registers all objects and provides interfaces to the configuration and control systems. A user specific *Application* plug-in defines the *Module* and *Device* objects and their data connections. Any external i/o with e.g. custom readout hardware, an event building network, the data storage, or an online monitoring server is implemented as *Transport* object with an associated *Device*. The actual data processing is carried out within *Module* objects. *Modules* and *Transports* can run in separate threads, or in shared threads of the DABC runtime environment. The configuration of all these functional entities is set via *Parameter* containers, initialized on startup from an XML file. Additionally, *Command* objects may be exchanged between the DABC entities even throughout different host nodes, allowing to invoke user defined actions at run time.

2.2. Integration to Detector Control system

The abstract DABC control system interface allows for setting the internal run state of each node, and for monitoring - and optionally modifying - any kind of parameter values, e.g. data rates, buffer fill levels, or output file names. Current DABC v1.9 distribution delivers the tcp/ip based DIM protocol [8] with specific "record" structures as a default implementation. A generic JAVA based GUI client is also provided that can connect to such DIM server nodes [9]. Moreover, the GUI of the Go4 analysis framework can monitor states and rates of DABC DIM nodes [10]. Figure 1 illustrates the different DABC control connections.

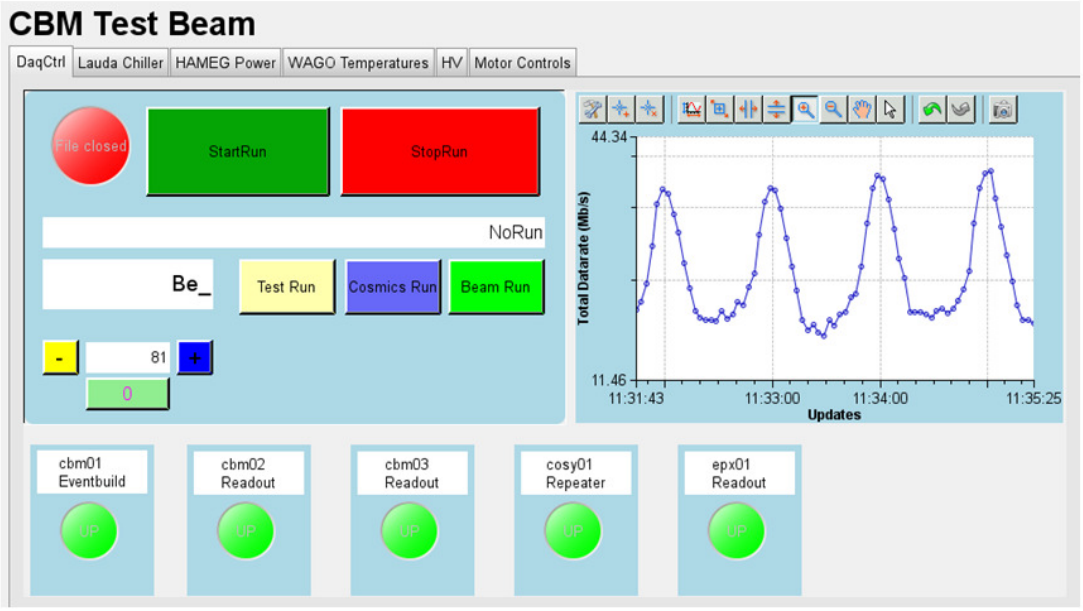

Figure 2. The CSS/EPICS GUI as used for DABC monitoring and run control at COSY 2012 beam test (see section 4.2.).

Because CBM chose EPICS [11] as common detector control system, it is advantageous to integrate DABC controls to this framework. For this purpose a separate DIM-EPICS interface application has been developed. Here the DABC records, as exported via DIM, are converted to EPICS process variables. The interface application can be run as a component of the main EPICS IOC,

together with the other detector control system modules of the beam test set up. Like all EPICS process variables, the exported DABC parameters can hence be controlled by a regular EPICS GUI.

For different beam test situations, dedicated DABC GUIs have been developed within the EPICS Control System Studio (CSS) environment [12] that offers powerful GUI designer tools. Since other beam test detector controls also use CSS system, the DABC GUI has been integrated with other subsystems to a combined control window. Figure 2 shows an example of the CSS designed DABC GUI used for DAQ control at COSY in January 2012.

2.3. Acquisition of EPICS slow control data

In test beam measurements the setup during data taking is intentionally changed very often, and the interpretation of the acquired detector data may depend on the experimental conditions. A direct access to such values in the DAQ data stream can be very useful for a conditional analysis depending on certain settings. Moreover, it would be possible to perform a fast scan of detector properties with the control system and record simultaneously the measured data in the same file. As the CBM test set-up is managed by EPICS, data readout of EPICS process variables has been developed as a plug-in for DABC [13].

This DABC readout plug-in is based on the existing Easy Channel Access (ezca) extension library. It can be configured by an XML file which contains the names of the process variables to be fetched from an EPICS IOC. One special process variable defines a “flag” which is polled by DABC with a configurable repetition time. Only if this flag variable shows a specific value, e.g. 0, the complete set of the defined records is acquired from the IOC. Thus the IOC can decide when EPICS data is recorded to the DAQ stream. Alternatively, all process variables may be fetched with a predefined time interval to ensure a continuous trending.

Since the EPICS readout has been implemented, it was applied at all test beams to monitor the detector set up. Another use case: EPICS changes some measurement settings automatically by a sequencer, and the combined EPICS and detector data is used for analysis of such scans. This is planned for STS laser irradiation tests as described in section 4.3.

3. Online analysis with Go4

3.1. The Go4 framework

The GSI Object Oriented Online Offline framework (Go4) [10] is a GSI standard tool for online and offline analysis, based on C++ and the ROOT framework [14]. Go4 features a plug-in architecture for user-defined subclasses, implementing data structures (“event element”), data sources (“event source”), and algorithms (“event processor”). Such plug-ins are compiled into a user analysis library which is run embedded to the Go4 framework.

The same user analysis code can run either from the command line in a single-threaded batch mode, or controlled by a generic Go4 GUI that offers a multi-threaded, interactive live display with full ROOT and Qt graphics (figure 3). Moreover, the same analysis can switch data input between a recorded file and an online monitoring data server on the fly, thus direct comparison of online and offline data is possible without restarting. As standard GSI data protocol, the “list-mode” file of the DAQ system MBS [6] and various MBS online data servers are supported by Go4. Because DABC system also provides these data source formats, Go4 can directly connect to DABC produced data.

The Go4 framework allows splitting subsequent stages of the analysis into “analysis steps”, with each step producing intermediate data which is used by the following step (figure 4). So the first analysis step would unpack the raw data from DAQ. The second analysis step would map such DAQ channels into a detector display and perform a specific analysis. An advanced analysis, like detector correlations or a simple beam “tracking”, can be implemented in a third analysis step. Each analysis step is defined by an “event processor” class which produces an “event element” structure as resulting output. A further offline analysis with ROOT macros could be based on such Go4 output events which are optionally stored into ROOT *TTree*s.

Because of such features, the CBM collaboration has applied Go4 since 2008 at several detector test beam campaigns. Most recently the Go4-based CBM online monitoring software has been newly organized to match the ever increasing complexity of such tests.

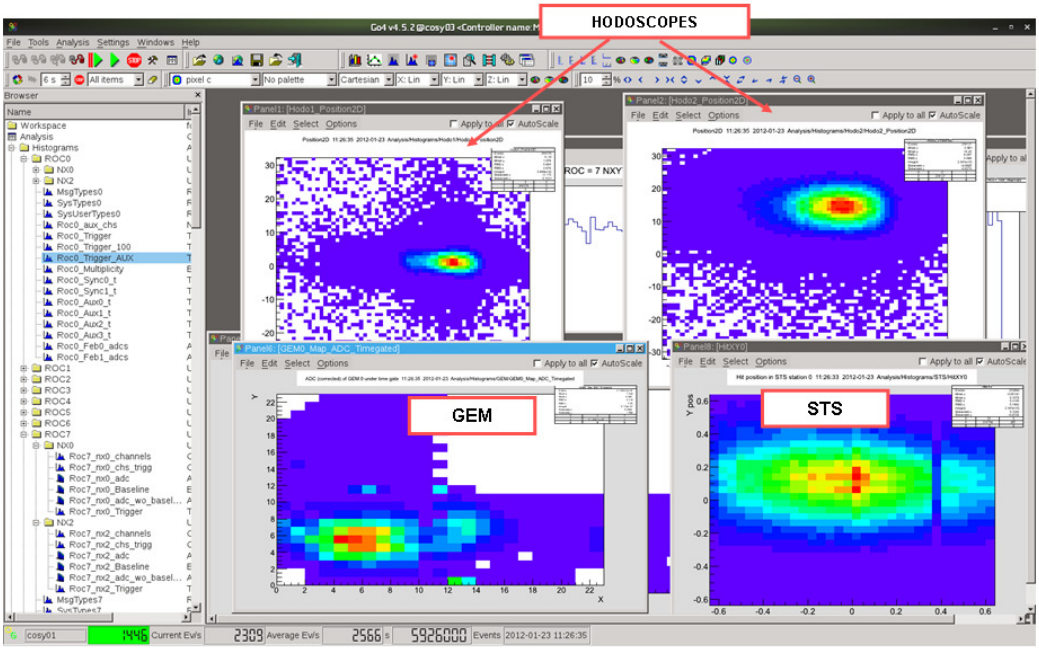

Figure 3. Go4 GUI with live display of accumulated hit maps from different detector prototypes at COSY 2012 test beam (see section 4.2.).

3.2. The CBM test beam framework and generic libraries

The Go4 analysis for a specific test beam should best reuse existing analysis code of the standard CBM readout components connected to the DAQ system DABC, like the SysCore readout controller (ROC) [15], the Susibo/SPADIC frontend [16], miscellaneeous VME-bus modules read out by an MBS crate, or the EPICS process variables readout as described in section 2.3. These parts are kept as Go4 user classes in separate shared libraries which are part of a "CBM test beam framework" that has been developed on top of Go4. DABC will pack the data of all these different frontends into a generic "MBS event" container structure, so the first Go4 analysis step can read them using standard MBS data sources. Besides the reusability of such libraries, they provide also a standalone Go4 analysis with all generic monitoring features required for a simple lab set up.

The actual CBM test beam analysis will usually not modify these libraries, but just use their classes which are available by default. Runtime configuration of the library objects for various test situations is done via Go4 parameters. These can be set to new values by scripts at analysis initialization, or can be saved and recovered with a ROOT file ("Go4 auto save file").

To cover the test beam use case of various readout systems for different test detectors, each of the Go4 analysis steps needs to be divided up into structurally parallel processing entities and their resulting data structures. Hence the CBM test beam framework introduces a "composite event processor" class that can register and run a number of regular Go4 "event processors" together within each analysis step. Additionally, the resulting data structures of these software processors are aggregated as a common "Go4 composite event", reflecting the hierarchy of the DAQ and detector set up. Figure 4 illustrates the structure of a typical implementation.

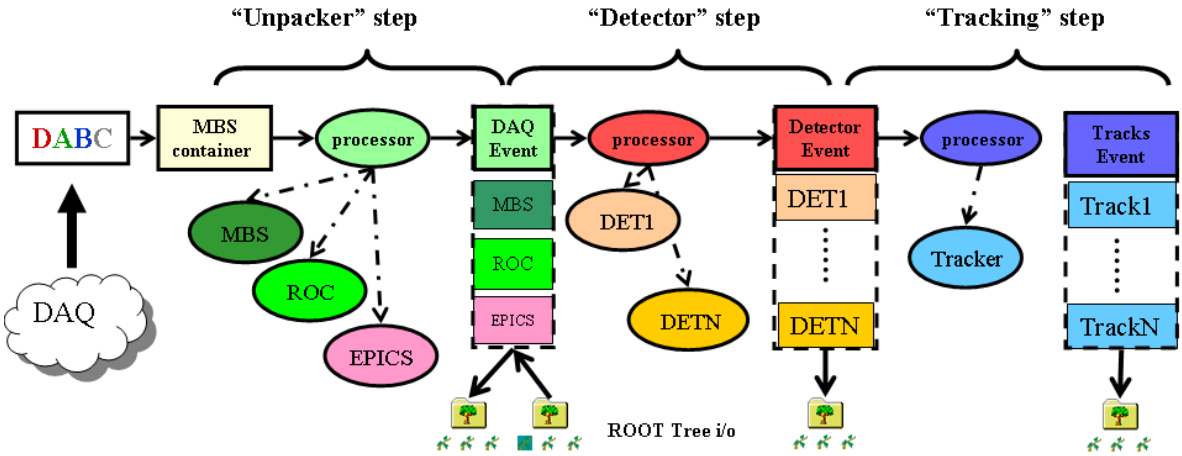

Figure 4. Go4 analysis software organization for typical CBM test beams. Subsequent data generations are separated by "analysis steps", each producing a resulting data structure. Each analysis step may run together code for independent frontend or detector components. The final step may show correlations, or trajectory fits of different components. The event structure of the legacy Multi Branch System (MBS) framework is used as generic data container interface for all DABC generated inputs.

3.3. Test-specific implementations

Although the code of the first analysis step is mostly generic, the further analysis steps are still too specific for each test beam to be reusable. Thus the Go4 analysis for each CBM test beam is organized in separate source code directories with a class name prefix of the top classes indicating the campaign (e.g. "CernOct11"). Depending on the particular set up, the composite event and processor classes of each analysis step will define different components. For the first analysis step these are mostly taken from the CBM test beam framework libraries, for the other steps the classes are implemented within the test beam specific library.

Within each test beam directory, the source code is also organized in a modular way: each detector subgroup can develop and run their individual code as standalone Go4 analysis. The combination of several or all subgroup codes can run as advanced or full analysis. Another benefit of such partitioning into different source code packages is the clear responsibility of each working group for their code.

4. Application examples

4.1. CERN PS October 2011

In October 2011, a two week test of TRD and RICH detectors was done at CERN PS beam line T9 with 1-10 GeV/c secondary electron and pion beams. Figure 5 shows the data acquisition set up. A total of 8 different detector systems is read out via three different frontend systems. Here conventionally beam-triggered readout systems, such as MBS [6] and the Susibo/SPADIC [16] , have to be combined with free running systems, such as the ROC [15] that continuously acquires time stamped messages. This is achieved by inserting special trigger synchronization messages into the ROC data stream via LVDS signal cable from a VME-bus FPGA module (VULOM) in the triggered MBS crate. These markers are then used by DABC event building application to pack corresponding data into the same MBS container.

Here two DABC event-builders are connected to the frontends via tcp/ip ethernet (MBS crates), udp/ip ethernet (ROC), and USB 2.0 cables (Susibo/SPADIC). The two DABC event-builder nodes (PC1, PC2) are combining at first half of the frontends each. Another DABC node (PC3) is dedicated to fetch the EPICS slow control data in frequent intervals. Finally, one of the event building DABC

nodes (PC1) combines the data of all three DABC processes, writes the data file and serves the online monitoring clients. Due to the low beam intensities, the average file acquisition data rates are typically about 500 kByte/s.

Online monitoring and initial offline analysis are implemented with the Go4 based CBM test beam framework. In the first analysis step, generic software modules are used, such as for the ROC/nXYTER readout, for the SPADIC readout, and for EPICS slow control variables. The unpacking of two MBS crates' data is treated with specific classes. The components of the second analysis step are developed by various institutes. It has different processor modules for beam monitoring scintillators, a fiber hodoscope, a RICH prototype, and several TRD stations of different prototypes. The results of such subdetector analyses are combined in the second analysis step and used for particle identification.

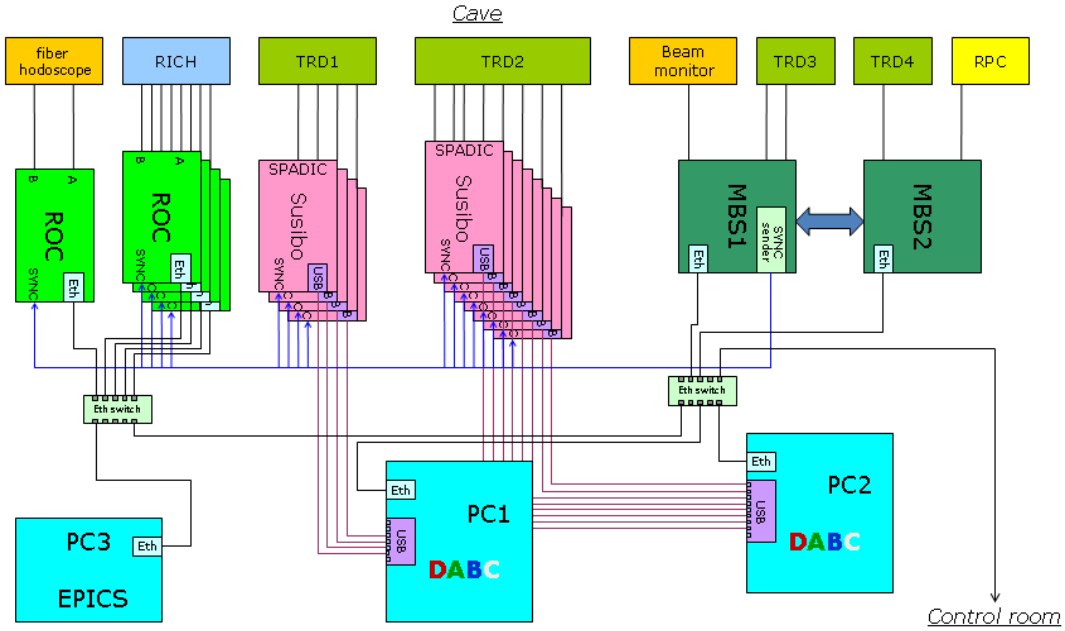

Figure 5. DAQ set up at CERN/PS test beam for CBM in October 2012. See text for details.

4.2. COSY January 2012

In January 2012, a test of STS and GEM/MUCH detectors was done with a 2 GeV/c proton beam at COSY (cooler synchrotron) facility in Jülich, Germany. Figure 6 shows the data acquisition set up which is much more homogenous than at CERN. There is only one type of readout system, such as the ROC [15], for all three detector types (STS, GEM, fiber hodoscope). Three DABC nodes are connected to the ROCs via optical fibres, intermediate data combiner boards (DCB), and PCIe receiver boards (AVnet). These connections use a proprietary CBMnet protocol [17]. All readout is purely "trigger-less", i.e. there is only a time-stamped data stream. Clock synchronization between the frontends is inherently done by the optical protocol. For initial reset of the time counters, deterministic latency messages are send from a DABC time master (PC3) node to a "clock master" DCB which distributes the synchronization to the other DCBs. The other two DABC nodes (PC1, PC2) read the data via optical fibre and forward them via Gbit ethernet sockets to the event-building DABC node (PC4) that stores it to local listmode files. Additionally, the DABC event-builder merges the EPICS slow control data which is retrieved by another dedicated DABC application on PC3 and is also read via ethernet. For performance reasons, the Go4 online monitoring server is running on a separate DABC node (PC5) which gets data stream samples from the event building PC4. Because of the

trigger-less readout mode and high beam intensities, average file acquisition data rates are limited by the DABC event-builder host speed. These rates have typically been adjusted to 10 MByte/s (out of beam spill) and 50 MByte/s (in beam spill).

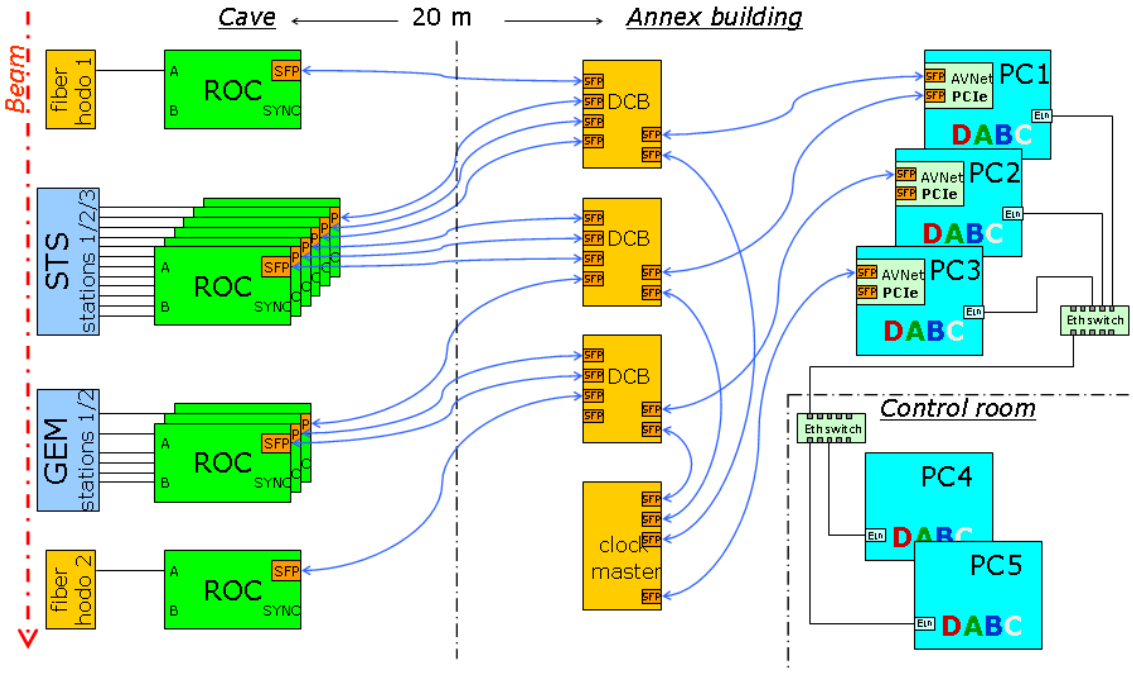

Figure 6. DAQ set up at COSY test beam in January 2012. See text for details.

Again the Go4 based CBM beam time framework is applied for online monitoring. In the first analysis step the generic ROC/nXYTER and EPICS monitors are used. Here a simple event selection within the message data stream is done: Go4 imposes a window condition of the ROC data message time differences with respect to a beam-induced marker message. This marker is derived from the first hodoscope detector's sum signal that has been fed to some of the ROCs and inserted to the data stream. The event selection time windows of all ROC streams can be adjusted interactively on Go4 GUI, regarding the histograms of the message time differences.

The second analysis step has generic components (event processor and event structure) for the two fiber hodoscopes, and special components for the STS, and the GEM prototypes. For the first time a third analysis step has been implemented, providing detector correlations and a simple linear fit of the single "beam tracks" through the hits of the two hodoscopes, three STS and one GEM planes.

4.3. STS laser table project

An example for the readout of EPICS process variables together with the other DAQ frontends is illustrated in figure 7. These measurements are under preparation for summer 2012. A silicon strip detector (STS) prototype is locally irradiated by a movable laser, the position of which is steered by an EPICS sequencer. The silicon strip signals are continuously sampled and read out by a self triggered SysCore2 readout controller (ROC) [15]. The combination of STS signals with a laser position is provided by DABC which inserts a special data marker ("system message") whenever the EPICS sequencer indicates begin or end of a laser position. The subsequent Go4 analysis software can evaluate such markers and can select all messages belonging to each laser scan point as a separate "Event". Currently these DABC readout mechanisms and the appropriate Go4 analysis have been developed and tested by means of ROC data simulators and a dummy EPICS sequencer. So the software is mostly ready for the experiments to be performed.

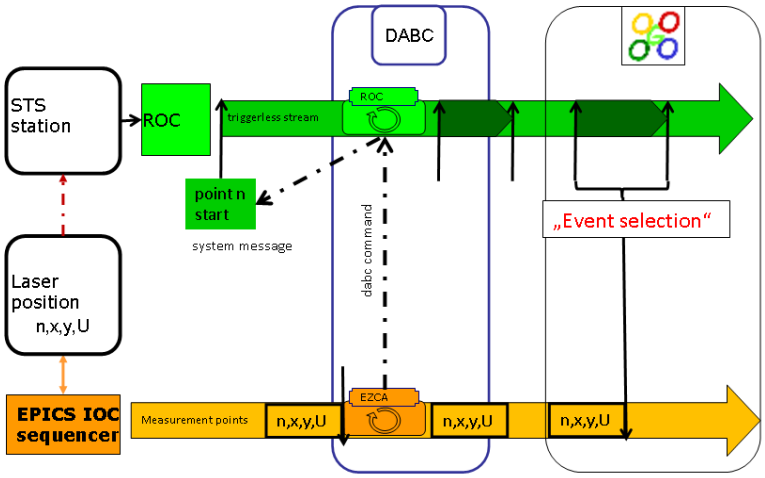

Figure 7. Application of EPICS readout for silicon strip station prototypes (STS) irradiated by a movable laser. The laser position is controlled by an EPICS sequencer. The strip responses are sampled by nXYTER/SysCore2 Readout Controllers (ROC) as trigger-less stream of time-stamped messages. Each laser scan point is combined with corresponding messages in Go4 analysis. For synchronization DABC inserts time markers into ROC data stream when EPICS signals a begin or end of scan point.

5. Conclusions

Data acquisition and online monitoring for current CBM detector and electronics prototype tests have been handled with two software frameworks: DABC for DAQ, and Go4 for online analysis. Both frameworks proved to be versatile enough to match the requirements of various set-ups. The EPICS system has been used to control both detector and DAQ settings in a combined GUI. Moreover, DABC can read out EPICS process variables and can insert them into the DAQ data stream. This allows analyzing detector signals with respect to the slow control set up.

The Go4 analysis code for CBM test beams has been organized in a modular way, defining a sub-framework with CBM specific libraries, and splitting up the user code for different test beams and working groups. This turned out to improve the code reusability and maintenance, and has helped a lot for the concurrent development of independent analysis parts.

Because future detector tests will have increasing complexity, online analysis requires advanced features to treat detector geometry, hit evaluation, and correlation between different detectors - up to track finding and fitting capabilities. The structure of Go4 framework can handle such requirements, as has been successfully demonstrated at COSY January 2012 campaign by means of a third analysis step. For more complex set ups, it might be useful to integrate here some existing analysis code which has been developed for CBM detector simulation and offline analysis with the FairRoot framework [18]. However, the intended use-case of the Go4 online analysis is limited to quality monitoring of detector and electronics tests. For a more advanced analysis, it is rather planned to develop and run FairRoot code independently on the same DABC-acquired data.

The upcoming next iteration of CBM readout hardware will also mean a remanufacturing of existing data acquisition and online analysis software. All common code for the formatting and simple “event selection” analysis of CBM specific detector front-ends should be put in a set of CBM-DAQ libraries. These should be independent of any other software framework, but are to be used by all participating systems, such as the DABC data acquisition, Go4 monitoring analysis, FairRoot physics analysis, and the future first level event selection algorithms to be run on the FLES compute farms.

References

[1] The FAIR project home page, http://www.fair-center.de

[2] Friman B, Höhne C, Knoll J, Leupold S, Randrup J, Rapp R and Senger P (eds.) 2011 *The CBM Physics Book*, (*Lecture Notes in Physics* Vol. 814) (Berlin Springer)

[3] De Cuveland J and Lindenstruth V 2011 A First-level Event Selector for the CBM Experiment at FAIR *J. Phys.: Conf. Ser.* **331** 022006

[4] Adamczewski-Musch J, Essel H G, Kurz N and Linev S 2010 Data Flow Engine in DAQ Backbone DABC *IEEE Trans. Nuclear Science* **vol.57 no.2** 614-617.

[5] Adamczewski-Musch J, Essel H G, Kurz N and Linev S 2010 Data Acquisition Backbone Core DABC Release v1.0 *J. Phys.: Conf. Ser.* **219** 022007.

[6] Essel H G, Hoffmann J, Kurz N and Ott W 2000 The general purpose data acquisition system MBS *IEEE Trans. Nuclear Science* **vol.47 no.2** 337-339.

[7] Adamczewski-Musch J, Essel H G and Linev S 2011 The DABC framework interface to readout hardware *IEEE Trans. Nuclear Science* **vol.58 no.4** 1728-1732

[8] Gaspar C, *Distribution Information Management system DIM*, http://dim.web.cern.ch/dim

[9] Essel H G, Adamczewski-Musch J and Linev S, 2010 A DIM Based Communication Protocol to Build Generic Control Clients, *Proc. 17th IEEE-NPSS Realtime Conf.* (Lisbon) IEEE Xplore RTC.2010.5750463

[10] Adamczewski-Musch J, Essel H G and Linev S 2011 Online Object Monitoring With Go4.V4.4 *IEEE Trans. Nuclear Science* **vol.58, no.4** 1477-1481

[11] The Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/

[12] The Control System Studio, http://cs-studio.sourceforge.net

[13] Adamczewski-Musch J, Kolb B W and Linev S 2010 DABC data acquisition input for slow control variables *GSI scientific report* **2010** 45.

[14] The ROOT System, http://root.cern.ch

[15] Abel N, Manz S and Kebschull U 2009 Design and Implementation of an Universal Read Out Controller *GSI scientific report* **2009** 323

[16] Armbruster T, Fischer P and Peric I 2010 CBM TRD Readout with the SPADIC Amplifier / Digitizer Chip *GSI scientific report* **2010** 32

[17] Lemke F, Slogsnat D, Burkhardt N and Bruening U 2009 A Unified Interconnection Network with Precise Time Synchronization for the CBM DAQ-System, *Proc.16th IEEE-NPSS Realtime Conf.* (Beijing) IEEE Xplore RTC.2009.5321841

[18] Bertini D, Al-Turany M, Koenig I and Uhlig F 2008 The FAIR simulation and analysis framework *J. Phys.: Conf. Ser.* **119** 032011