The control and data acquisition software for the gamma-ray spectroscopy ATCA sub-systems of the JET-EP2 enhancements

A. Neto\textsuperscript{a,}\textsuperscript{*}, J. Sousa\textsuperscript{a}, B. Carvalho\textsuperscript{a}, H. Fernandes\textsuperscript{a}, R.C. Pereira\textsuperscript{a}, A.M. Fernandes\textsuperscript{a}, C. Varandas\textsuperscript{a}, G. Gorini\textsuperscript{b}, M. Tardocchi\textsuperscript{b}, D. Gin\textsuperscript{c}, A. Shevelev\textsuperscript{c}, K. Kneupner\textsuperscript{d},

JET-EFDA Contributors\textsuperscript{1}

\textsuperscript{a}Associação EURATOM/IST, Centro de Fusão Nuclear, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
\textsuperscript{b}Istituto di Fisica del Plasma, EURATOM-ENEA-CNR Association, Milan, Italy
\textsuperscript{c}A.F. Ioffe Physico-Technical Institute of the Russian Academy of Sciences, St. Petersburg, Russian Federation
\textsuperscript{d}Institut für Energieforschung, Forschungszentrum Jülich, Jülich, Germany

Available online 24 October 2007

Abstract

A local control and data acquisition (CDAQ) sub-system for gamma-ray spectroscopy diagnostics is being developed for the JET-EP2 enhancements. The Hardware implementation is based on the Advanced Telecommunications Computing Architecture\textsuperscript{TM} (ATCA\textsuperscript{TM}) and will permit acquisition at a very high count rate (up to few MHz) with digital pulse processing (DPP) on a field programmable gate array (FPGA) performing pulse height analysis (PHA), pulse shape discrimination (PSD) and pile-up rejection (PUR) algorithms.

This paper presents the CDAQ software implementation, which is based on the FireSignal platform developed by the Euratom/IST Association. The FireSignal is based on a client/server modular approach, where both the server and the hardware clients are defined by a generic XML description. Its event-driven operation is dynamic thereby permitting continuous control and data acquisition and ‘plug-and-play’ of clients/hardware. The FireSignal is being extended to interface the diagnostic sub-systems to the JET CODAS HTTP data/event transport layer and to integrate the diagnostics specific requirements including the hardware device drivers, real-time spectroscopy processing and local control operation.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Data acquisition; JET; CODAS; CORBA; XML

1. Introduction

The recent upgrade in the JET gamma-ray spectroscopy diagnostic includes the replace of the current electronics by fast, high resolution transient recorders with digital pulse processing on a FPGAs, allowing PHA, PSD, peak time and pile-up detection algorithms to be performed in real-time and with a better dynamic range/pulse count rate performance. Each time a new diagnostic is to be integrated in a new data acquisition system all the inherent communication protocols must be studied and a new implementation must be written. JET already provides an abstraction layer based on the HTTP protocol \cite{1}, where any equipment is considered to be a “blackbox”—from the Control and Data Acquisition System (CODAS) point of view.

Before commissioning a system at JET, it is common that several test phases take place in different laboratories and only when the system is ready the integration with CODAS is performed. This paper presents an alternative solution where the diagnostic is not directly connected to CODAS, but integrated on an highly modular system named FireSignal \cite{2}, which in turn is connected to the JET acquisition system framework. This guarantees that once a system has been tested outside JET it can be easily connected to CODAS without further developments. Furthermore, it allows any system integrated within FireSignal to work on JET.

\textsuperscript{*}Corresponding author. Tel.: +351 21 8419096; fax: +351 21 841 7819. E-mail address: andre.neto@cfn.ist.utl.pt (A. Neto).

Fig. 1. In FireSignal all modules are connected through CORBA allowing them to run in different operating systems and to be written in different computer languages. Currently FireSignal is using Java, Python and C++ as main languages.

2. FireSignal

FireSignal is a data acquisition and control system software designed to control and operate physics experiments. It is based on a distributed architecture where each module is completely independent from the rest of the system and can even be programmed in different languages. This kind of abstraction is achieved with the use of CORBA [3] network infrastructure, which exposes a common interface to all the modules. The system is based on events, rather than shot numbers, where the data is tagged by time-stamps and relevant events, allowing for instance searching for data relative to a particular event or within a time-frame. Events are defined by a unique name identifier and a time-stamp.

The system can be divided in five different specialized modules, as illustrated in Fig. 1. FireSignal provides a skeleton with all the basic functionality. When specific implementation details are requested the system is extended and changes can be easily integrated. All the modules are connected through the Central Server which acts as a communication bridge. The Database Controller is responsible for storing and retrieving data. New data is handed over to the storage system which translates the requests from FireSignal into the specific storage language. Currently FireSignal provides “out of the box” Database Controller implementations for PostgreSQL™, MySQL™ and for a simple file system database. The Security Manager authenticates and authorizes operations from both users and hardware. The remaining modules are of great relevance for this work and are described in the following two subsections.

2.1. Node module

Hardware is driven by the node module. Each kind of device is described by an extensible markup language (XML) file and validated by a document type definition (DTD) file. The XML provides information about all the configuration limits and data types for each data acquisition channel, allowing FireSignal to automatically validate inputs and correctly handle acquired data. The node is responsible for managing events provided by the Central Server for which it can, for instance, trigger hardware or change acquisition rates. If the connected hardware generates events, the node is allowed to broadcast these events for the rest of the system. Before sending acquired data back to the server, the data is tagged with the correct time-stamps and associated with the events that triggered the acquisition.

FireSignal provides generic nodes in three different languages: C++, Python and Java. These already implement all the CORBA communication, XML parsing and generation, data validation and a series of helper functions to assist on data tagging, acquisition rates to time-stamp conversions, among others. For each different type of device a connector between the generic node module and the hardware driver must be developed.

2.2. Client module

This module provides a way for end-users to interact with the system. Authorized users are allowed to dynamically configure and change hardware connections and status, send software events and visualise acquired data. Usually the client is associated to a graphical user-interface (GUI) but it can also work as a software connection to another acquisition system. FireSignal provides a complete multi-user GUI to interact with the system, using the XML files provided by the different nodes to automatically build configuration interfaces that know how to validate and render the several inputs, and how to display the acquired data. Multiple users can be connected at the same time and share the configuration of the apparatus,
status of the experiment, launch new discharges and follow experiments.

In the case of a connection to another control and data acquisition system (CDAQ) the native CDAQ behaves as a client to FireSignal, where the calls from both systems are translated using a software connector previously developed. As referred to above, FireSignal is a multi-client system, allowing the GUI to co-exist with the connector to the native acquisition system. This can be particularly useful for debugging and following the experiment locally.

2.3. ATCA interface to FireSignal

The ATCA crate will host a FireSignal system, running on Linux, composed of a generic C++ node, the server and a file system database. The node is responsible for controlling and configuring the ATCA digitizer. The JET control and data acquisition system is a FireSignal’s client.

The digitizer board will be described in an XML file where it exposes all the available configuration capabilities, the number of channels and the type of data produced. Fig. 2 shows a reduced set of the XML file used for the configuration of the ATCA digitizer.

A native Linux driver, connected to the generic FireSignal C++ node module, will be used to control the acquisition boards, translate the configurations sent by the server and retrieve the acquired data.

During the commission phase the file system database will be used to store locally the data acquired by the nodes. This feature is already available.

3. JET diagnostic control

The HTTP Finite State Machine (JetFsm) describes the different stages of a JET pulse. JetFsm is described in Fig. 3.

The interface for every diagnostic to the JET central control and storage system is a HTTP-based protocol, called the “blackbox” protocol. For this the diagnostic provides a HTTP server, where the central software system (named GAP) acts as a client. This software prepares a diagnostic for a pulse and collects the data after the pulse finishes. From the central control system viewpoint the gamma-ray spectroscopy diagnostic is special due to the large amount of data involved. The collection of 4 GB of data in between pulses (~20 min) cannot be guaranteed due to high workload for data collection already existing after each pulse. Therefore, the data from this system is declared as a late pulse file (LPF). The data collection for such files does not happen necessarily before the next pulse, being often collected only overnight. As a guideline the diagnostic has to able to store data from 2 days.

Fig. 4. The JetConnector component will be used to connect to FireSignal with JET. This component is responsible of driving FireSignal through events generated by state changes in the HTTP server.
4. Connection between FireSignal and JET

The ATCA system is responsible for the real-time data collection of the gamma-ray spectroscopy diagnostic. A controller named JetConnector will be developed to connect FireSignal to the JetFsm (see Fig. 4). The JetConnector uses the JetFsm internally for the communication with the HTTP server. Users of that control component are informed about upcoming pulses, the corresponding pulse number and parameter settings.

The JetConnector implementation will also manage data directories which are used by the diagnostic for storage and free the disk space after data collection.

Each JetFsm state change is translated into one FireSignal event. The most relevant JetFsm states are: init, used for hardware configuration; abort, which causes FireSignal to stop the acquisition; wait_pulse where all the nodes are put in running state and start waiting for data. Every time there is a new configuration, the JetConnector warns FireSignal which then fetches the new parameter values and sends it to the nodes to program the hardware accordingly.

The ATCA crate will be connected to the JET Composite Timing and Trigger Signal (CTTS) which will deliver absolute time and will be used by the nodes to tag the acquired data.

When a new shot sequence starts the Central Server changes the hardware and node status to running and waits for a trigger from the CTTS. As soon as new data is available in the hardware, it is broadcasted to the Central Server, delivered to the Database Controller and saved into the directories previously created by the JetConnector. If a pulse sequence is aborted this information is immediately passed to all nodes, allowing them to stop and if needed restart the hardware.

5. Conclusions

FireSignal provides an easy way of integrating new diagnostics at JET. When a particular piece of equipment is known to work with FireSignal, it is guaranteed that it will also work inside the JET CODAS. Since FireSignal is free and open-source, providers are allowed to develop and test all the needed software outside JET.

This new concept will be used for the GRS, a particular project where a mixture of different technologies and operating systems is used, and where a large amount of data is expected.

The FireSignal server is interfaced through the HTTP JET interface running on Linux. FireSignal is already running on some Linux servers around the world [2] as a complete data acquisition solution. This is the first project where it is layered on other control and data acquisition system.

Acknowledgments

This work has been carried out within the framework of the Contract of Association between the European Atomic Energy Community and “Instituto Superior Técnico”.

References