10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, PO1.093-8 (2005)

REPLACEMENT OF OUTDATED VME BOARDS AS STARTING POINT FOR CONTROL SYSTEM MODERNIZATION

L. Hechler, K. Höppner, P. Kainberger, U. Krause, G. Schwarz
GSI, Darmstadt, Germany

ABSTRACT

Control systems are operational for quite a long time, easily spanning more than a decade. Along the way facilities are extended and operations are more and more refined. The control system has to be modified and extended accordingly. This has to be done on the background of rapidly evolving technology. Hardware components and software products on which the system originally was based are no longer available while progress in the IT area eases implementing additionally requested features.

Preparation for continuous evolution was not a primary aspect in the design of the GSI control system. As a consequence, introduction of new hard- and software components were avoided whenever possible. The system depends on many components which dated from the beginning of operation. Replacing a no longer available VME board was taken as an occasion for a general renovation of the system. Rebuilding the middle layer of the control system allows much simpler integration of new components and prepares the system for future extensions.

PRESENT GSI CONTROL SYSTEM

Architecture

The control system was designed as a decentralized distributed system (Fig. 1), according to the nowadays well established standard model. The accelerator's equipment is connected by a field bus to front-end controllers which communicate by Ethernet with operation workstations. M68020 VME boards are used at the front-end level while the operations level uses OpenVMS Alphas.

In contrast to many other systems; the front end side is divided in two layers, device presentation computers (DPC) as access points for operations requests and equipment control (EC) computers to service the devices. Assigning the tasks to different processors eases ensuring precisely timed device control. Synchronized by signals from the timing system, only the ECs have to implement real-time functionality.

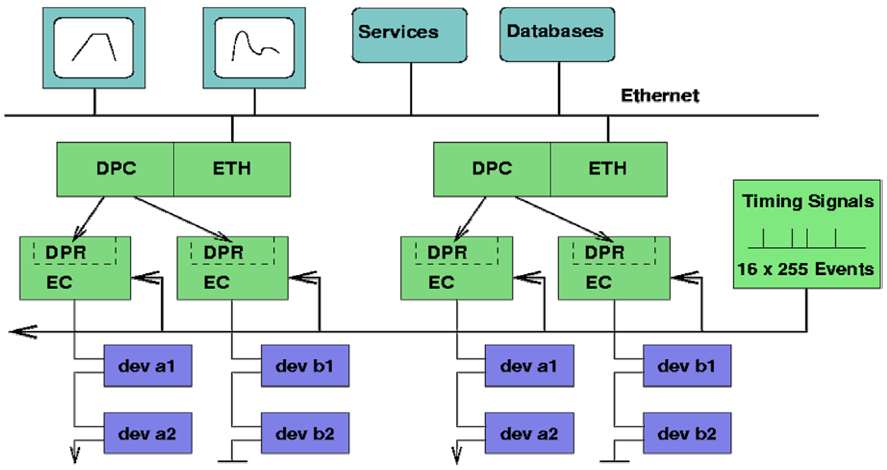

Figure 1: Hardware layout of GSI control system

One DPC serves up to nine ECs located in one VME crate. Special VME controller boards, which interact closely with the DPCs, establish the network connection. ECs and DPCs communicate by common memory on the ECs, and by VME interrupts from the ECs.

Software Structure

The GSI control system models the equipment of the accelerator as independent devices with a set of properties. Implemented on the DPCs as functions, properties handle specifics of the various device types. Grouping the USRs to so called "equipment models", the structure reflects well the object oriented paradigm. Equipment models correspond to classes, properties to methods, and devices to control system software object.

A system manager process on the DPCs handles all general tasks like administration of devices and the set of USRs on the particular VME node, and execution of commands from the operation level. A network board, implementing GSI's network protocol, connects the front-ends to the Ethernet.

The DPC layer does not interact with the accelerator's equipment directly but provides the EC layer with data to handle the accelerator's equipment. Similar to the DPCs, device specific handling is coded in functions, called Equipment Modules (EQMs).

Applications on the operation level communicate with the devices by one general interface. Access is qualified by the name of the device and the name of the property. Synchronous and asynchronous calls are supported as well as connected commands, in which a response can be sent from the VME-level either periodically or each time a specified timing event was received. Various data formats are supported by exchanging via void pointer and specifying type and count explicitly. Automatic conversion is provided between the data types supported by the control system.

Messages can be broadcasted by the VME level which can be received via a central alarm process on each operation level workstation. This is widely used to signal error conditions and all kind of status changes.

Status of the System

Since commissioning, a successively increased flexibility could be achieved in the operation of the GSI accelerators. Regularly up to five experiments are serviced in parallel, on a pulse to pulse basis, with three different ion species. Operation can be switched between experimental support, allowing free access to all parameters, and a rigid mode in which it is operated with verified and confirmed data. Used for medical cancer therapy, the latter mode automatically pauses the experimental operation when requested from the irradiation place.

The control system could cope with the growing demands from the accelerator's operation by adding various extensions. However, modifications of the core components are very delicate. The control system core was closely tailored to the original environment. Nearly all software, including network software implementing an in-house protocol, was built from scratch. Referencing internal data structures instead of using well defined interfaces lead to strong dependencies between the various components.

To reduce unforeseeable consequences, upgrades of the control system had to be restricted to components similar to the original ones. Alphas replaced the original VAXes on the operation level, and a newly developed board with M68020 processor replaced the first generation of ECs. The DPCs still use the original hardware, designed in the late eighties.

Replacement of the DPCs which are no longer available is indispensable now. Introducing a more recent board would be possible. But the rigid structures, and the limited capabilities of the underlying components, request too much effort to fulfil the growing wishes for extended functionality. New demands, like integrating the existing GSI facility as injector in the proposed FAIR facility [1], could hardly be fulfilled with the system in the present state. A general revision of the GSI control system is strongly advised.

Update Strategy

One could think of switching to a completely new control system to overcome the restrictions of the existing system. But the present operation of the GSI facilities could be achieved only after a long refinement. Switching devices between different beams on a pulse to pulse basis requested special handling in the system core, and optimized software in the USRs and the EQMs to handle the specifics of the various device types. Similar refinements would be needed in a new system too which would request far too much effort to accomplish in reasonable time.

Decision was made for rebuilding the middle layers of the control system instead. At the operation's side, the device access interface (user interface, UFC) provides a well defined interface to the application level. The system is cut at the device access interface on one end and at the connection between DPCs and ECs on the other end, and the middle layers are rebuilt. With limited explicit coding identical handling of the various device types is assured by integration the existing USRs.

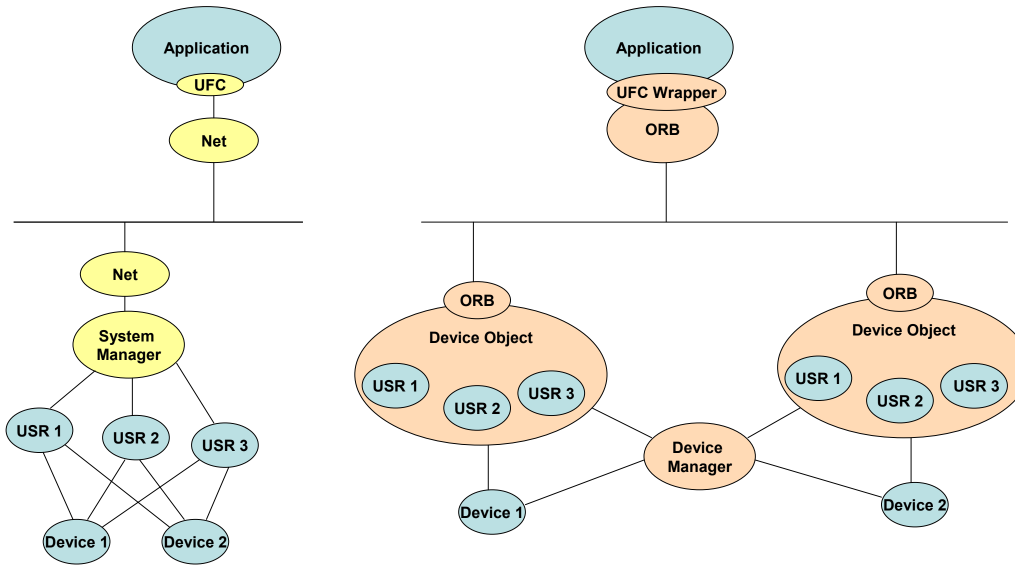

Figure 2: Replacing the Control System Middle Layer

Compared to the original implementation, much less coding is needed due to the broad spectrum of ready to use middleware which is nowadays available for general tasks. A TCP/IP based network communication, as an example, will use the onboard network interface, provided by modern boards, and soft- and hardware of the present network boards become obsolete. Products based on well established standards should be used to expect long term availability. Open source components are preferred not only for cost reasons but also to be more independent from management decisions of single companies.

SOFTWARE REBUILD

Outline

The renovated DPC software will be running under common Linux. Since all time critical activities are handled on the EC level, no real-time extensions are needed. Linux provides a very convenient environment for software development and is available for a broad range of platforms, also in the embedded field. Because of the close coupling between DPC and ECs, which cannot be overcome completely in the present renovation, the new DPC board has to use the same byte order as the ECs with their big endian M68020. A PowerPC VME board was selected which is available with preconfigured Yellowdog Linux.

Object oriented techniques are used throughout the new software components. Quite naturally, the accelerator's devices then are represented by software objects on the DPCs. Remote access to these devices will be based on the CORBA standard because it fits perfectly to the communication needs in accelerator control. Selection was for omniORB on Linux platforms while for OpenVMS a commercial ORB had to be purchased.

Special attention has to be given to future upgrades. At least parts of the today's control system will be in use for a long time. To achieve a maximum of flexibility for future usage, a good modularization is mandatory. Encapsulation of implementation details and access over well defined interfaces only will be major design criteria.

Device Access Interface

Devices on the new DPCs will be equipped with a new CORBA access interface. Access via the present application's interface however must be still possible to allow seamless integration. This restricts the basic outline of the new interface. Selection of the property has to be by name, and data exchange must support all types of data used in the control system. Synchronous as well as asynchronous access has to be implemented, including "connected commands" in which a response is sent periodically or on selected timing system events.

The CORBA interface could, in principle, be accessed directly within an application. However, to make the application's code independent from the CORBA data exchange, an adapter to separate the transport's specifics is provided. As long as the adapter interface remains unchanged, the underlying remote access may be modified or even completely replaced by switching to another access mechanism.

Data of various types, and varying count, can be exchanged with the front-ends. Since it turned out to be very useful to mix integers and floats in one call, the existing control system provided rudimentary support for mixed data types. To retain the present level of flexibility, data exchange is handled by a container. It allows insertion of any one of the supported data types in any order. When unpacking the container on the receiving side, the type of each element can be determined to extract it accordingly. Conversion to other data types can be requested too.

Several CORBA data representations have been tested to find an efficient exchange of the container's content. The preferred mechanism, covering a sufficiently broad spectrum, is to use sequences of basic types in unions, which on their part then can be arranged in a sequence. If at least about 20 data of identical type follow each other, overhead against a pure sequence is negligible.

INTEGRATION IN THE EXISTING ENVIRONMENT

Device Specific Code

Reusing the existing device specific code on the DPCs is fundamental to provide the same device functionality as in the old system. The existing implementations, the USRs, have to be integrated in the device objects. Each USR is converted to a class, derived from one Usr base class, in which the old function body is transformed to a method read(), or write(), respectively.

Each device object stores a list of the USRs which are defined for the specific device, together with the names of the properties they represent. When a property is called, the corresponding USR is searched and called with the data to be exchanged. A more detailed description can be found in [2].

Command reception and evaluation, and calling the requested USR from the device's lists are implemented by general classes which are inherited by the device class. Coding is needed only to transform the existing USRs to new classes and to register them. The main modifications in the USRs affect data exchange. While in the old implementation the data section of the received network packet was directly referenced via a void pointer, the new implementation decouples the USR classes from the transport implementation. The same data container as in the operation application interface is used.

Connection to ECs

DPCs and ECs communicate via VME bus by common memory on the ECs. Each device owns its specific area, used for data exchange between USRs and their counterpart on the ECs. A separate region is assigned to the system software. The memory layout is defined as a single complex structure. Communication is implemented by referencing specific elements of the structure.

To make the new DPC software independent from the memory layout, access to the system's part of the common memory is encapsulated in new EC classes. Modification of the memory layout, or using a different communication mechanism, will only affect these EC classes but not the main part of the DPC implementation.

Directly referencing the common memory is still widely used in the USRs. In the present stage, coupling could be reduced by not referencing the common memory as a whole but using pointer to the device's data. It is planed, and partly done already, to make data exchange transparent by new communication classes. The required conversions of the USRs would need too much effort to be spent at short time. Identical data layout, both from EC and DPC, still has to be assured. This is achieved by using the same GNU GCC compiler for both boards, and explicitly requesting natural data alignment in the M68020 compiler.

Operation Level Device Access Interface

Programming on the operation level is based on OpenVMS. Not many ORBs are available for this nowadays somehow exotic operating system. When work started, none of the well known Open Source ORBs were available for OpenVMS. A commercial ORB was purchased from 2AB Inc.

Keeping the old device access interface for applications on the operation level is indispensable when switching to the new CORBA based device access. Only then the huge amount of application software to operate the facility can be used without modifications. The old interface therefore has to be provided as a wrapper around the new one. This access interface is implemented as a set of procedures which are linked to each application.

While synchronous requests can be mapped rather straight forward, handling of asynchronous requests need special provisions. In the old interface responses to asynchronous requests are handled in an interrupt routine immediately upon reception. Data are stored internally and have to be read explicitly by the application. The CORBA device interface handles asynchronous requests via callback object which is called from the front-end DPC when the request has been processed.

A separate thread should be started to provide access to the callback object independent from the application's current activities. However, the thread support of the OpenVMS ORB in use may conflict with activities in the application which are not under control of the access interface. Therefore it is planned to handle all asynchronous activities in a subprocess which communicates by mailboxes with the interface in the application. Unfortunately, this introduces another level of complexity in the device access. However, usage of the old access interface is expected to decrease on the long term. New applications should be built on the new adapter on the CORBA interface.

Alarm System

Integration of the new DPC components in the alarm system has just started. Alarm messages have to be broadcasted to the existing alarm handlers. UDP multicasts are the fully adequate IP-equivalent to the presently used transport over the GSI network. A first prototype, in which received messages were transferred via a gateway to the alarm handlers, quickly demonstrated feasibility. In the final implementation, most notably, proper conversion of the byte order has to be added. While presently alarms messages are always send in little endian order, messages in the new UDP channel will be extended by an indicator of the byte order used, to allow swapping by the receiver when needed.

Access Devices Interpreter

Besides a comfortable operating environment, a flexible way to access devices is needed during commissioning and for troubleshooting. It must be possible to call any device property interactively, and to rapidly write scripts for more complex activities.

At GSI an interpreter for the Nodal language is used, running under OpenVMS. Its well structured design would allow porting to other operating systems. But then a huge package of software, written in the language Modula-2, would have to be processed. Anyway, the Nodal language with its line numbers, and with limited support for function calls, is no longer an adequate programming environment. A modern replacement is desired.

The Python interpreter was configured to handle CORBA calls by installing omniORBpy. Python provides state of the art object oriented programming and, being an interpreter, allows interactive access as well as scripting.

EXPERIENCE

Progress was slow at start of the project because developers were not familiar with many new components like Linux, CORBA, and also the PowerPC. After passing the learning phase, the selected environment allowed to focus on the GSI specifics rather than dealing with IT basics. No cumbersome adaptation of the operating system was needed as with the old boards. The new DPC board with its preconfigured Linux only had to be configured and was operational in short time. While in the old GSI implementation a significant part of the code handled the communication between operation level and front-ends, off the shelf CORBA, and the IP protocol, cover communication needs with much more comfort and flexibility. C++'s standard template library reduces implementation effort further. The relief was a prerequisite to start the renovation. Complete in-house development, as in the existing the control system, never would have been possible.

Python, with its integration of the CORBA standard, showed as the most striking benefit of nowadays ready to use components. Within a few days it was possible for a Python newbie to write a wrapper to encapsulate the CORBA calls which provides a device access interface as simple and easy to handle as the device access interface in Nodal.

Development could concentrate on the specifics of the GSI installation. The new software has to reproduce the established access interfaces, and has to integrate the device specific adaptations, the USRs, which could be achieved rather straightforward. A first version of the front-end components, including the adapter to encapsulate the device access, is operational. The OpenVMS device access interface showed to be more difficult than expected. Compared to UNIX and Windows, fewer products are available for OpenVMS, and often they provide less flexibility. Effort for integration of the CORBA communication in the existing access interface was underestimated. While synchronous device access fits well, asynchronous commands request laborious and awkward substitution.

SUMMARY AND OUTLOOK

Continuous adaptation to technical progress is of great importance for control systems which, during their long usage, have to be refined more and more. If delayed too long, one may suddenly be in a dead end situation when the system has come to its limits. A general revision is then unavoidable. With today's availability of powerful middleware it should be possible to cut the Gordian knot by rebuilding core parts of the system with reasonable effort.

To avoid similar situations in the future, attention must be given to flexibility. Clear separation of modules, and encapsulation of the underlying components, is a key aspect. With proper modularisation upgrades should be restricted to limited areas and will not affect the system as a whole.

When the present work will be finished, a major step on the way to modernize the GSI control system will be achieved. Renovation has to continue, like decoupling the DPC and the EC layers to allow other platforms than VME only, and extending the programs for the accelerator's operation to Linux and Windows. However, the fundament will be solid then. More important than refining the GSI system will be to migrate it towards other control systems. The core which is now under construction provides options to establish connections to other control systems. Components of these systems then can be integrated into the GSI environment. The goal is to pick the best of various systems and to combine it to a powerful controls platform. Only a combined effort can master the challenges from complex future facilities like FAIR. The GSI system will be prepared to proceed on the way. The flexibility gained for future developments more then compensates the effort for rebuilding the middle layers which had been, as often, underestimated.

REFERENCES

[1] P. Schütt et al, “Operational Modes and Control Aspects of FAIR”, ICALEPCS’2005, Geneva, Switzerland, October 2005.

[2] K. Höppner, L. Hechler, P. Kainberger, U. Krause, “Embedded Linux and CORBA in the GSI Control System”, PCaPAC 2005, Hayama, Japan, March 2005.