Wednesday, 28 November 2001

WEA - Evolution of a Control System, Maintenance, Upgrading, Re-Engineering: Challenges and Design

WEAI001
physics/0111210

COSMIC—THE SLAC CONTROL SYSTEM MIGRATION CHALLENGE

M. Clausen (DESY), Ron MacKenzie, Robert Sass, Hamid Shoaee, Greg White, Leeann Yasukawa
SLAC, Stanford, CA 94025, USA

Abstract

The current SLC control system was designed and constructed over 20 years ago. Many of the technologies on which it was based are obsolete and difficult to maintain. The VMS system that forms the core of the Control System is still robust but third party applications are almost non-existent and its long-term future is in doubt. The need for a Control System at SLAC that can support experiments for the foreseeable future is not in doubt. The present B-Factory or PEPII experiment is projected to run at least 10 years. An FEL laser of unprecedented intensity plus an ongoing series of fixed target experiments is also in our future. The Next Linear Collider or NLC may also be in our future although somewhat farther distant in time. The NLC has performance requirements an order of magnitude greater than anything we have built to date. In addition to large numbers of IOCs and process variables, Physicists would like to archive everything all the time. This makes the NLC Control System a bit like a detector system as well. The NLC Control System will also need the rich suite of accelerator applications that are available with the current SLC Control System plus many more that are now only a glimmer in the eyes of Accelerator Physicists. How can we migrate gradually away from the current SLC Control System towards a design that will scale to the NLC while keeping everything operating smoothly for the ongoing experiments?

1 ORACLE

Recent releases of the Oracle RDBMS provide expanded capabilities that make it reasonable to consider using it for configuration information, archive and perhaps even soft real-time data.

1.1 Configuration Information

Since its inception in the early 1980's, the SLC control system has been driven by a highly structured memory resident real-time database. While efficient, its rigid structure and file-based sources makes it difficult to maintain and extract relevant information. The goal of transforming the sources for this database into a relational form is to enable it to be part of a Control System Enterprise Database that is an integrated central repository for SLC accelerator device and control system data with links to other associated databases.

We have taken the concepts developed for the NLC Enterprise Database [1] and used them to create and load a relational model of the online SLC control system database. This database contains data and structure to allow querying and reporting on beamline devices, their associations and parameters. In the future this will be extended to allow generation of controls software configurations for EPICS and SLC devices, configuration and setup of applications and links to other databases such as accelerator maintenance, archive data, manufacturing history, cabling information, documentation etc. The database is implemented using Oracle 8i.

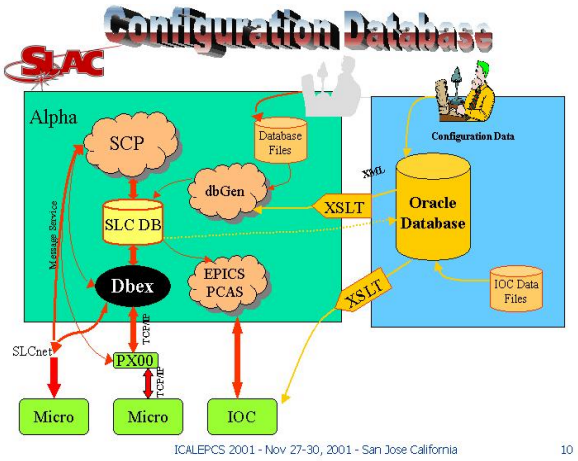


Figure 1: Proposed Configuration of SLC and EPICS Real-time Databases Using Oracle

Figure 1 above depicts the use of Oracle for maintenance of configuration as we envision it to be. We presently have a substantial subset of SLC data in Oracle and are using it for some query applications but are not yet generating the SLC database or loading EPICS IOCs with data generated from Oracle.

1.2 Archive Data

Presently, the SLC and EPICS archive systems are completely separate. They each have disadvantages and neither will scale to the Petabyte sizes eventually

needed by the NLC. Recent test results [2] for archiving into Oracle are very encouraging and present plans are to move forward rapidly on this part of the Cosmic development.

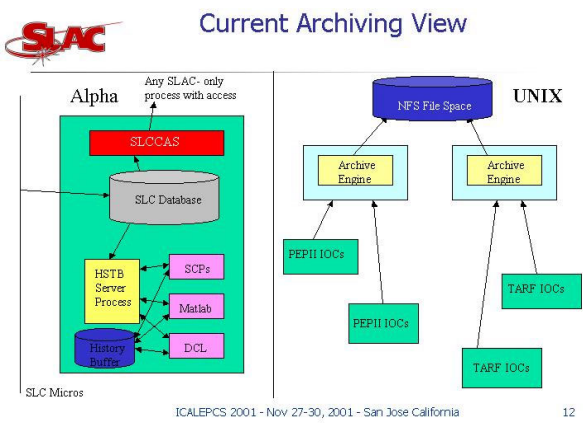


Figure 2: Current Archive Configuration in the SLC Control System

The SLC archive system has been running for many years and one can retrieve and analyze archive data over that entire span of time. Retrieval times are acceptable and data can be plotted using standard SLC Control System utilities. The data can also be exported to Matlab for detailed analysis and plotting. Despite these capabilities, it does have some serious disadvantages.

1. Data can only be retrieved and analyzed on the MCC VMS system.
2. There's a 6-minute update interval for most signals which is too coarse for many kinds of analysis.
3. Data is stored in an internal format and so must be explicitly exported by special software for use by other tools.

Figure 2 above shows an overview of the current SLC archive systems. You see how the two systems are completely separate without mutual access to data.

The EPICS archiver saves data using EPICS monitors and thus only when it changes or at some pre-defined maximum interval. It is supported by the EPICS collaboration and works with a number of different EPICS tools but it also has some serious disadvantages.

1. It uses a doubly linked set of files so all files must be online for it to work.
2. There are a limited number of tools for data access and maintenance.
3. Since the data are kept in binary files, it's difficult to use commercial tools to access/analyze the data.

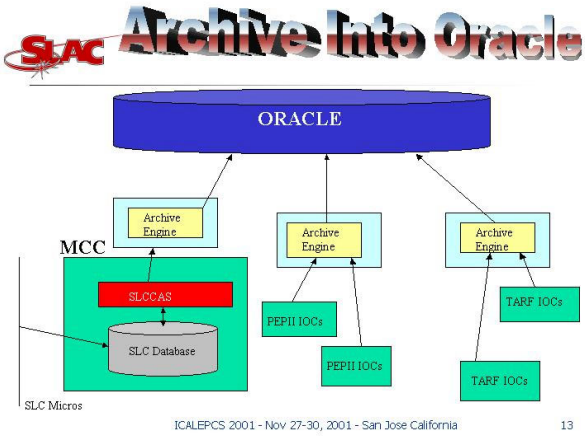


Figure 3: Proposed Archive Configuration for the SLC Control System

Figure 3 above depicts the proposed solution to the archive problem. The Portable Channel Access Server running on the MCC VMS system will interface to an Archive Engine that will be used to funnel data into Oracle. Other EPICS IOCs in the system will also use other instances of the Archive Engine so all SLC Control System data will be in a common Oracle archive. Once in Oracle, a wide variety of Oracle and third-party tools can be used to retrieve and analyze the data.

2 AIDA

Higher-level applications need to access a logically related set of data that is in different data stores and may require different processing. Accelerator Integrated Data Access (AIDA) is envisioned to be a distributed service that allows applications access to this wide variety of Control System data in a consistent way that is language and machine independent. It has the additional goal of providing an object-oriented layer for constructing applications on top of multiple existing conventional systems like EPICS or the SLC Control System.

2.1 Overview

The client interface to AIDA is via the CORBA Interface Definition Language or IDL. This interface is the "contract" between the clients and the AIDA services and sources. AIDA uses an Oracle database to map the names requested to services and sources. Initial discovery is made at runtime and subsequent

requests can the go directly to the data, bypassing the Oracle access.

Requests can be either synchronous or asynchronous. The CORBA Event service is presently used for monitors. The CORBA Notify service is layered on top of the Event Service and has additional attractive features like quality of service client-side and event filters.

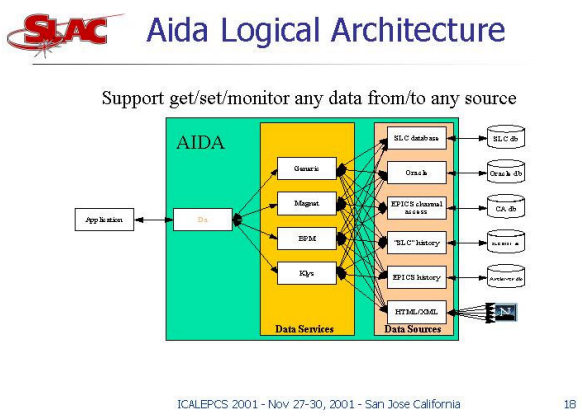


Figure 4: AIDA Architecture Overview

Figure 4 above shows the major parts of the AIDA architecture. The client's request is dispatched to the appropriate service which then gets the data from the appropriate source. The service may pre/post-process the data before storing/retrieving the requested data items. A generic service is provided for those data items that don't require additional processing.

2.2 Request Details

Figure 5 below follows a typical request through the AIDA system.

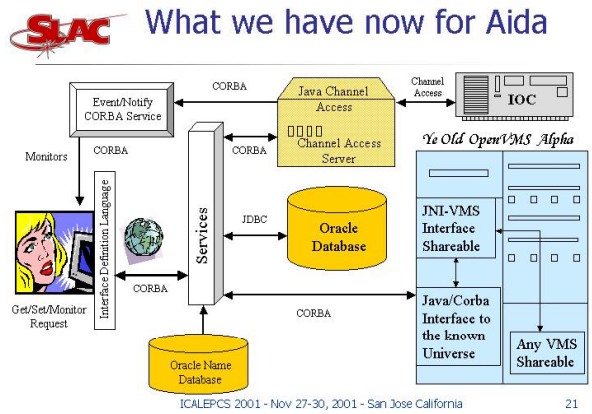


Figure 5: AIDA Request Example

The client interfaces with an AIDA object via the methods and structures specified in the IDL or Interface Definition Language. The method invocation on the remote CORBA object determines the service and sources of the data items requested by accessing the Oracle name database or its local cache if the item is still there from a previous request. The name database has information about the services and sources needed to process the request. Each of the data sources has one or more AIDA servers to access the local data store. Requested data is either returned directly to the client in the case of a synchronous read or asynchronously using the Event service if the request is for a Monitor.

3 JOIMINT

A final piece of the migration puzzle is a new display manager/user interface tool, Java Operator Interface and Management Information Tool (JoiMint). The existing SLC interface is a now antiquated touch panel that interfaces with the underlying applications. It may be old but it's very fast and thus operators are able to use it for effective control of the accelerator. The EPICS DM (DM2K EDM) can only read/write to data channels but the SLC interface has several additional features that will be required in any new user interface.

- It communicates with underlying applications.
- It can capture keystrokes/operations and save them to a file. This file can be edited with basic loop and conditional controls and later replayed.
- Separate local and system message windows.
- Dynamic loading and configuration of new objects

Matthias Clausen of DESY started preliminary work on this and there is an update to be presented at this conference [3].

4 SUMMARY

Figure 6 below shows how using Oracle, AIDA and JoiMint/Madam we can provide an environment for porting and implementing applications allowing us to gradually offload VMS while giving us a development environment that will hopefully be applicable to the NLC.

4.1 Archive to Oracle

Because early tests have been so promising, probably the first part of the migration will be the archive data. Merging both EPICS and SLC archive data in an Oracle database will give us good experience in managing large data stores. It will also provide an opportunity to explore the wide variety of Oracle and third-party tools for data extraction and analysis.

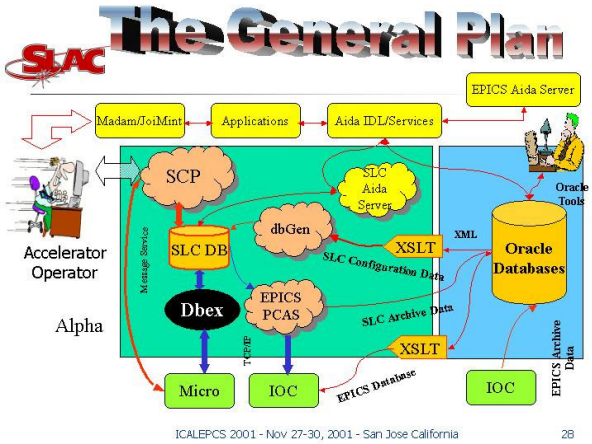


Figure 6 General Migration Plan

4.2 Configuration Management in Oracle

We have already done considerable work in modeling and implementing the SLC database in Oracle. Several other EPICS sites have used Oracle to store their EPICS database and we may use their implementation if it can be reasonably integrated with our existing design. The file system based storage of database information has served us well for many years so we'll need a high degree of confidence in the new system before switching over.

4.3 AIDA

We plan to finish prototyping and start detailed design early next year. We hope to have a production quality version ready for initial application development later on in the year. CORBA capabilities are rapidly evolving. If you procrastinate long enough there will be a specification and eventually a product to do what you want. Current CORBA specifications already encompass real-time capabilities, failover and redundancy, messaging and embedded CORBA. As part of the telecommunications applications there is also a CORBA message logging service. After the core of AIDA is robustly functional, we'll need to continuously monitor new CORBA capabilities and incorporate them or replace existing implementations as warranted.

4.4 JoiMint

This development is presently being undertaken by DESY based on requirements and initial prototyping work done at SLAC. An intelligent display capability is crucial to the Cosmic plan.

5 THE FUTURE

Even after the major applications have been ported and enhanced to the new environment, we have not provided for migrating the running database and the interfaces to our Intel Micros. Perhaps in the end these will just stay on a VMS system with the bulk of the applications running on the outside. These micros present several unique challenges.

- They run the iRMX operating system, which is now almost as obscure as VMS.
- Much of the code is tightly coupled to the Intel architecture in its handling of segments and pointers.
- All communication is done through the SLC home-built message system. We would either need to port support for it to other platforms or consider changing this to a CORBA message service.

None of the above options are particularly attractive and involve touching a lot of very old code that is at the heart of controlling the running of the accelerator. In the end, if VMS remains a supported operating system, we may retain the core of the database and communications on VMS while exporting the bulk of the applications and displays to a more distributed and modern environment.

REFERENCES

[1] T. Lahey et al. An Integrated Enterprise Accelerator Database for the SLC Control System. These proceedings.

[2] LeeAnn Yasukawa et al. Archiving into Oracle. EPICS Collaboration Meeting December 3-4, 2001, San Jose CA.

[3] Matthias Clausen et al. An XML Driven Graphical User Interface and Application Management Toolkit. These proceedings.

WEAT002
physics/0111060

RE-ENGINEERING OF THE GSI CONTROL SYSTEM

U. Krause, V. RW Schaa, GSI, Darmstadt, Germany

Abstract

After more than 12 years of operation without substantial revision a modernization of the control system at GSI is overdue. A strategy to adapt the system to future needs is outlined. The system has to support a specific environment of which the main features are described. More flexibility than in the current system can be achieved while still using many parts of the actual system.

1 INTRODUCTION

The actual GSI control system started operation in 1989. Many extensions and refinements have been developed since but no substantial revision could be made. As a result the system is outdated in many aspects. Only one environment is supported, one fieldbus, one type of device controller, one operating system for the applications. Hardware is no longer available, (e.g. the controller boards from 1990), and support for Pascal as a programming language for the device control software ended.

Most components were developed to inhouse standards. Interfaces between components are too complex. Therefore, exchange of single components is costly. A general revision is overdue. The modernization has to consider the characteristics of the GSI environment.

2 CONTROL ENVIRONMENT

2.1 GSI Accelerator Operation

GSI operates three accelerators for all kind of ions from hydrogen to uranium: The linear accelerator Unilac, the synchrotron SIS and the storage ring ESR. The linac and synchrotron are operated in a pulse to pulse time sharing mode with a repetition rate of 50 Hz of the linac and 0.1 to 0.5 Hz of the synchrotron. Switching to different ion species, energies, and experimental targets is done with this rate. Three independent ion sources serve in parallel, typically five experiments at Unilac, SIS and ESR. The average duration of an experiment is one week.

In addition to this flexible experimental operation, a rigid mode for heavy ion cancer therapy is provided [1]: Any carbon beam from a fixed set of 254 energies, 15 intensities, and 7 spot sizes will be delivered to the irradiation place by request.

2.2 Device Handling

Pulse to pulse switching demands a more complex device handling than simple schemes like ‘set reference’, and ‘read actual value’. This can be illustrated with magnets in Unilac cycles for low-charged ions. One controller has to service up to 12 magnets in 20 ms cycles. Preset time has to be maximized to reach high currents, and stable current duration has to be limited to reduce thermal load.

Broadcasting a medium level set value at the start of a cycle gives slow devices time to reach full current. To avoid overheating of fast devices, the dedicated set value is delayed for 4 ms. After the end of beam, all magnets have to be set to a zero value. The duration of stable currents is too short to read actual values directly. So ‘sample and holds’ are triggered for read-out. The controller cannot read these values until the following cycle after reference values have been set.

2.3 Device Realization

In most cases, the devices connected to the control system are rather complex. Some components are implemented as distinct parts like the extraction kicker with 28 modules. On the other hand, some hardware components host several devices, e.g. profile grids. Up to 16 grids are distributed on 8 channels in one measuring device. Multiplexing restricts measurement to one channel per cycle. Nevertheless, the variety of different implementations has to be presented in a comprehensive way to the operation crew. Grids have to appear as independent devices and the kicker modules have to be combined to one kicker.

3 ACTUAL CONTROL SYSTEM

3.1 Scheme of the control system

The hardware outline of the actual control system [2] is given in Figure 1. The diagram also reflects the logical view since each module is rigidly connected to one of the hardware levels.

All devices are linked via field bus (modified MIL 1553) to distributed equipment controllers (EC). Synchronized operation is achieved by triggers from programmable central timing units, one for each accelerator. Supervisory controllers (SC) handle interaction with the operation level. One SC serves up to nine ECs.

Communication between EC and SC is done by dual ported RAM on the EC. SCs and workstations communicate

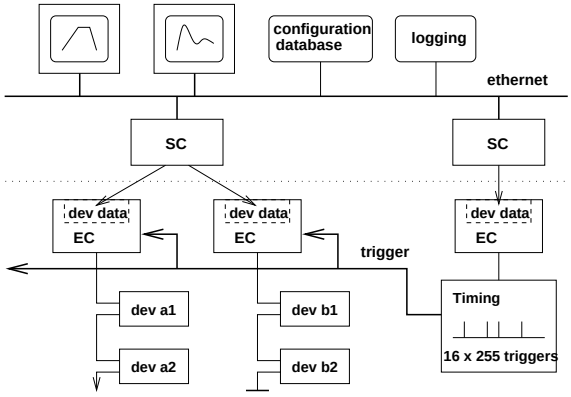


Figure 1: Schematic view of the actual control system

via ethernet by an in-house protocol comparable to UDP/IP. VME boards with 68020 processors are used for ECs and SCs. Operation level workstations run OpenVMS.

3.2 Real-Time Control

The real-time level is the most substantial part of the GSI control system. Device interactions are triggered by signals from central timing units. Up to 255 different triggers allow flexible adaption to the accelerating process.

ECs run autonomously under control of the timing unit after device data have been supplied. Up to 16 different sets of data, called virtual accelerators, may be configured in parallel on the ECs. This enables pulse to pulse switching between as many different beams. Beams for cancer therapy are handled analogously [1].

The sequence of virtual accelerators is determined online by the timing units: Beam is produced only on request by the experimental area.

Execution of commands from the operation level is provided. Device interrupts and polling services for survey of the devices are supported.

3.3 Device Representation

The devices are represented in an object oriented manner as independent units even though the control system was developed in a procedural way. Unique device names, the so-called nomenclatures, facilitate addressing. Every property is modelled by an action, coded as a procedure on the SC, with data to be exchanged, e.g. sending a reference or reading an actual value. Properties are identified by name and described in a formalized way by type and count of corresponding data. Based on this description, one single interface allows access to every property of every device.

To keep applications well structured, nomenclatures represent independent objects with relevance for the process of acceleration. These logical devices are constituted on the real-time level: Every nomenclature must have a corresponding entry on the EC. Mapping of the connected hardware to the operations view is demanded.

4 UPGRADES

4.1 Strategy

Replacement of the control system by a different one would require a lot of effort. Existing devices and interfaces must be supported, the achieved quality of accelerator operation has to be provided, and existing control hardware has to be used further in order to reduce expenses.

Actually 2750 devices are controlled by 256 ECs in 41 VME crates. Device specific adaptions can be subsumed in 61 classes, each requiring its own handling. Implementation of device adaptions requires 145,000 lines of code (LOC) compared to 26,000 LOC for common system software. The effort needed to reimplement peculiarities could be seen when a functional prototype of the Unilac timing unit was rebuild. Although it provided less functionality than is implemented today, and work was done within the same environment, it took about one person year.

Fortunately the architecture of the existing control system is still up to date. Stepwise migration to a system similar to the actual one but providing greater flexibility is managable. This will result in an up-to-date system and will allow re-use of most spent investments.

4.2 Outline of the Future Control System

The outline of the proposed future control system is shown in Figure 2. It is still a three level approach: Device control engines, device representation by logical devices, and the application level. Different from Figure 1, it is a logical view. Components can be installed on every hardware level of the control system.

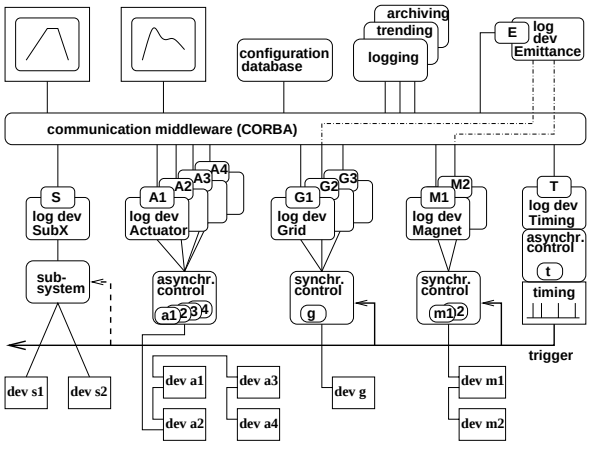


Figure 2: Schematic view of the future control system

4.3 Synchronized Device Control

Strong point in the existing control system is the real-time device control. It proved to be very well adapted to the needs of flexible pulse to pulse operation of the GSI accelerators. Therefore, the same mechanism will be used in the

future. This allows utilization of the existing generation and delivery of triggers for device synchronization.

Actions to service devices autonomously, under control of the timing system, are realized in synchronous device control engines. The actual EC software can serve as a first version of these engines. Fortunately, most of the effort to implement device specific adaptions was on the EC, both in size and complexity.

4.4 Extensions for the Device Control Level

In many cases a synchronization with the accelerating cycle is not needed, e.g. for slow mechanical actuators. An asynchronous device control engine will be provided by reducing a synchronous engine to polling and service of device interrupts.

Real-time systems are difficult to survey and to debug. To keep the control engines simple, any mapping between the hardware and the process view should be avoided here, e.g. the electronics for profile grids should be handled as one unit. Consequent reduction to the kernel functionality often allows description of device characteristics by simple tables. These devices can then be handled by common software.

In the current EC software, the hardware dependent parts of the code have to be identified to make the control engines portable. Implementation will be possible on any computer in the control system, on dedicated device controllers and on multi purpose computers. Nevertheless, if good real-time performance is needed computers with a hard real-time system will be used.

4.5 Device Modelling Level

An object oriented approach suggests representation of connected hardware as objects, called logical devices. Device functionality in the sense of the actual control system then corresponds to methods of the logical device.

Dealing with various device specific methods can be confusing. Therefore every logical device will have a common method as interface. Variable data formats can be exchanged by using types like CORBA's `any`. In IDL notation this method may look in principle like

```
void access(in string property, inout any data)
    raises (ControlException);
```

Logical devices are suitable locations to map the accelerator hardware to a process oriented structure. Different abstraction levels can be build by cascading, e.g. magnets and profile grids can be combined to emittance measurement devices. Any unit can be integrated as a logical device at this level: Complete subsystems like SCADA systems, devices with an OPC interface, or even software objects like databases.

A scheme for the logical devices has to be developed which is more general than in the actual system. Effort for implementation of device specific adaptions can be reduced by integrating code from the existing system. Procedures corresponding to properties can be transformed to objects' methods. Header and exit part of the procedures have to be replaced but the body can be kept with slight modifications. This allows a fast integration of the existing implementations but of course can only be a first step. Modern software development techniques allow much more elegant solutions compared to realisations in the actual system.

4.6 Networking

Communication with logical devices, distributed objects, is straightforward. Common middleware, like CORBA based systems, supports all needs of accelerator operation. Naming services, or alternatively explicit handling of object references, allow addressing of logical devices by name.

4.7 Application Level

No detailed investigations have been made yet. OpenVMS and Unix are based on similar concepts. This suggests the use of Linux as future basis for the application level.

In a transitional period, both operation systems have to be supported in parallel. To allow further usage of existing applications, the current interface for device access has to be provided in the future.

4.8 Preparatory Work

The device control software was written in Pascal. Actual software development systems are now based on C++. To enable future usage of existing software, conversion of the code to C has started [3].

Substantial changes of the actual system can easily impact the ongoing accelerator operation. To limit the implications of modifications, the modularization has to be enhanced. In a re-engineering process each module in the control system has to be provided with structured interfaces to reduce coupling. Only after this, will it be possible to replace existing components or port components to other platforms with acceptable effort.

5 CONCLUSION

The paper outlines a strategy for a rejuvenation of the existing control system. It shows the possibility to enhance flexibility and capacity of the system and nevertheless to integrate many parts of the existing system. The modernized control system will be suitable for the proposed new accelerator facilities too.

6 REFERENCES

[1] U. Krause, R. Steiner, "Adaption of a Synchrotron Control System for Heavy Ion Tumor Therapy", Proceedings of ICALEPCS '95, Chicago, USA, 1995.

[2] U. Krause, V. Schaa, R. Steiner, "The GSI Control System", Proceedings of ICALEPCS '91, Tsukuba, Japan, 1991.

[3] L. Hechler, "Converting Equipment Control Software from Pascal to C/C++", these proceedings.

THE EVOLUTION OF THE DAFNE CONTROL SYSTEM: A HISTORY OF LIBERATION FROM HARDWARE

G. Di Pirro, G. Mazzitelli. I. Sfiligoi, A. Stecchi, INFN-LNF, Frascati (RM), P.O. Box 13, 00044 ITALY

Abstract

During the DAFNE [1] commissioning and run operations the Control System [2] has been continuously evolving in order to fulfill the user requirements and the needs of a complete accelerator management. The original structure of distributed CPUs relaying to a central shared memory proved to be scalable and suitable for adding functionality 'on the fly'.

After 5 years of operation, the system had reached some intrinsic limits so that we focused on an upgrade plan mainly regarding the user level and the capability of the system to connect to external resources.

The re-engineering was done thinking to the maximum software reuse and the original choice of a commercial software environment for all the control applications demonstrated to be valid allowing to redesign the user level with no worries for the porting.

1 DESIGN STATEMENTS

When we started the Control System development we had a large amount of devices (Tab. 1) and a few people involved in the job.

Table 1. Device List

Device	Quantity	Interface Type
Magnet Power Supplies	425	Serial
Vacuum Pumps	157	Serial
Beam Position Monitors	123	MUX,DVM
Vacuometers	43	Serial
Fluorescent Flags	23	I/O
Ion Clearing Electrodes	39	ADC,DAC,I/O
Beam Loss Monitors	31	Scaler
Kickers	10	ADC,DAC,I/O, GPIB
Video Multiplexers	10	Serial
Scrapers	12	Stepper Motor Control
Programmable Delays	7	GPIB
RF Cavities	3	ADC,DAC,I/O
Beam Current Monitors	3	DVM

In order to optimize the development time we decided to use commercial technologies as much as possible. A commercial product is characterized by a broad distribution, which means a lot of feedback from the users and, consequently, deep debugging. Furthermore the wider the distribution of a product, the more reliable is its support from the producer.

Another criterion was to provide "easy development and maintenance".

We decided to use:

- LabVIEW [3] as the development environment for all the software;
- industrial VME bus to house the front-end hardware.

2 SYSTEM GENERAL DESCRIPTION

We distributed VME crates all around the machine in order to have the front-end hardware close to the devices to be controlled. At that time, LabVIEW was available only for Macintoshes so we decided to develop a customized processor based on the Macintosh LCIII mother board as a VME controller. The result was a fully operative Macintosh 68030 computer, joint to an interface, performing the VME System Controller functions.

The distributed CPUs make up the system 3rd level where the applications dedicated to device handling and control reside.

All the CPUs run asynchronously and write into their own VME memory the result of the control tasks for all the devices of which they are in charge. The data refresh time ranges from a few Hz to 50 Hz depending on the number of devices, the interface type and the complexity of the process.

From a data point of view, the system 3rd level consists of several local memory pages where all the machine objects are represented with descriptive records continuously updated at their own rate. All 3rd level VME crates are connected to a central cluster of VMEs through point-to-point optical links to constitute a central common addressing space called 2nd level.

The end result is a virtually central memory where the machine RTDB (Real Time Database) resides.

The transaction from the 2nd up to the 3rd level is transparent to the user, that can fetch any descriptive

record from a remote memory page with a simple VME read cycle. These read actions are performed from the system 1st level where the consoles and the user applications reside.

3 SYSTEM IMPLEMENTATION

In the first system version we used Macintoshes 68040 as consoles. The connection between the operator's consoles and the VME was done by dedicated VMV [4] interfaces.

The system (see Fig. 1) main peculiarities were:

- "true" memory mapping of the central virtual memory into the consoles internal address space;
- uniform hardware and OS (Macintosh at any level).

The DAFNE commissioning started and continued until December 1999 with this setup.

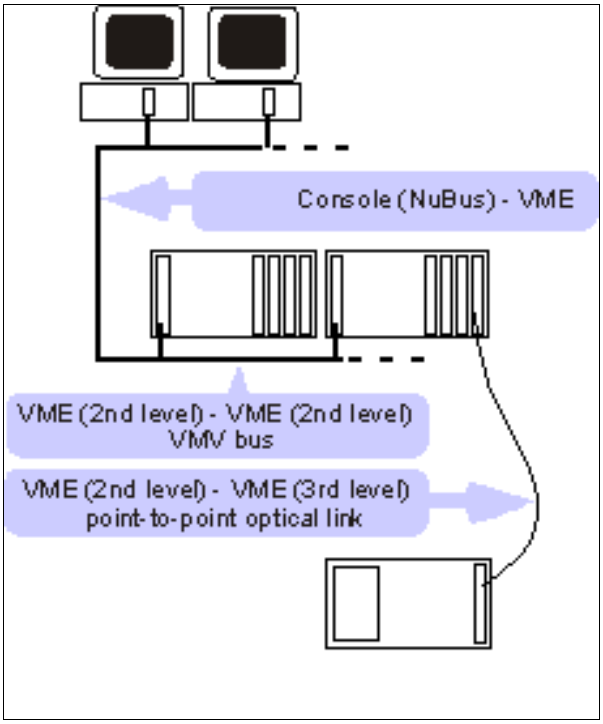


Fig. 1 Original System Implementation

4 SYSTEM UPGRADE

After 5 years of operation the system general structure based on distributed CPUs and a central shared memory demonstrated to be valid with no limitations from the hardware. It allowed easy and fast data gathering and correlation.

Also, the distributed processors were shown to be suitable for the front-end tasks, hence we focused on the 1st level for the upgrade project.

The operator level had reached some intrinsic limits:

- the 68040 μP was no longer able to stand the load of always-heavier requirements ;
- the Macintosh NuBus platform was dismissed.

We set the following targets:

- to improve 1st level reliability and performance;
- to get rid of the consoles connection bus and therefore the limitation on the number of consoles;
- gain remote access on the consoles;
- to have Internet media and services fully available.

An obvious issue was to reuse, as much as possible, the software already developed and this imposed the requirement to adopt computers able to run LabVIEW.

We chose Sparc VME embedded computers by FORCE [5] with Solaris operating system instead of Macintosh consoles. The first benefit of using VME embedded processors was to get rid of all bus-to-bus interfaces and cables needed to access the RTDB.

First, we re-wrote the VME read/write basic routines and then we encapsulated them into conventional LabVIEW graphic nodes. After this, porting all the user applications in LabVIEW for Solaris went on smoothly and required just a little cosmetic make-up and a few minor adjustments.

We estimated to load up to 5 user sessions on each Sparc CPU. We installed four diskless FORCE computers and a Sun Enterprise 250 as server over a switched 100 Mbps Ethernet network.

The interaction with the user applications running on the VME processors is done by mean of *SunRay* [6] lightweight terminals. These terminals are centrally managed by, and draw their computing resources from the *SunRay* server software that runs on the Enterprise 250.

The system performance, concerning the graphic presentations and the window management, greatly improved and also console hangs due to low memory disappeared.

This architecture (Fig. 2) is fully scalable: it is possible to increase the number of VME embedded processors and hence the power dedicated to user applications and to add "on the fly" *SunRay* terminals in order to have more working points.

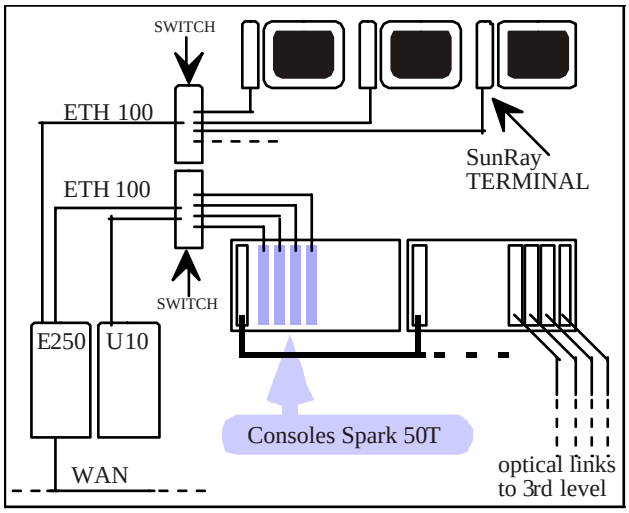


Figure 2: New implementation of the Control System.

After 6 months of development, tests and debugging and 3 months for the installation, the new system met all the upgrade targets, becoming operational in March 2000 (Table 2 summarizes the number and type of processors employed in the system in its present status).

Table 2: System main components

3rd level CPUs	Custom Mac LCIII	43
2nd level CPUs	FORCE 50T	4
1st level term.	SunRay	15
System server	Sun Enterprise 250	1
WWW server	Sun Ultra 10	1

5 NEW DEVELOPMENTS

Presently, we are working on the upgrade of the 3rd level layer of CPUs. This became essential for more than one reason:

- we are running out of spare CPUs;
- the new LabVIEW version does not run on 68K processors;
- dealing with MacOS 7, the remote development and debugging is hard to do.

On the basis of our present experience we looked for a VME controller that would allow us to reuse all the existing LabVIEW software even on a different platform. This led to the choice of a Pentium diskless board by VMIC [7] with Linux OS, which matches all the new requirements and still has a reasonable cost. Now, we have written the basic VME access routines and we plan to test the new 3rd level CPU at the beginning of 2002.

6 CONCLUSIONS

The system has evolved through 3 different operating systems and 5 major LabVIEW releases, smoothly withstanding these shocks. This has been possible thanks to the use of software written for personal computers with the characteristic of portability.

We are confident that we will always be able to use more suitable hardware, reusing the software work done. This is the meaning of "liberation from hardware".

7 ACKNOWLEDGEMENTS

We want to thank G. Baldini and M. Masciarelli for their commitment and essential contribution in the system implementation, O. Coiro and D. Pellegrini for their support in the hardware installation and setup.

We are also grateful to all the Accelerator Division staff for their continuous suggestions and encouragement for making a good and useful job.

A dedicated acknowledgement is due to C. Milardi for her tight collaboration to the high-level software development.

REFERENCES

[1] G. Vignola and DAFNE Project Team, DAFNE: The First Φ-Factory, EPAC'96, Sitges, June 1996, p. 22.

[2] G. Di Pirro et al. "DANTE: Control System for DAFNE based on Macintosh and LabView", Nuclear Instrument an Methods in Physics Research A 352 (1994) 455-475.

[3] LabVIEW, National Instruments Corporation, 11500 N Mopac Expwy, Austin, TX 78759-3504 USA (http://www.ni.com)

[4] Creative Electronic System S.A., Route du Pont-Butin 70 CH-1213 Petit-Lancy 1 Geneva Switzerland (http://www.ces.ch)

[5] Force Computers GmbH, Prof.-Messerschmitt-Str. 1, D-85579 Neubiberg/München (http://www.forcecomputers.com)

[6] Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA (http://www.sun.com)

[7] VMIC, 12090 S. Memorial Pkwy, Huntsville, AL 35803 USA (http://www.vmic.com)

WEAT004
physics/0111025

UPGRADE OF THE PHOTON BEAMLINE CONTROL SYSTEM ON THE SRS

B.G.Martlew, B.Corker, G.Cox, P.W.Heath, M.T.Heron, A.Oates, W.R.Rawlinson, C.D.Sharp,
CLRC Daresbury Laboratory, Warrington WA4 4AD, UK

Abstract

The SRS is a 2GeV synchrotron light source with 14 beamlines serving approximately 34 experimental stations. Control of the major elements of the beamlines (vacuum pumps, gauges, valves and radiation stops) is the responsibility of the main SRS Control System. As part of the long-term upgrade plan for the SRS Control System, a large programme of work has been undertaken to modernize beamline control. This work included: development of Linux based PC front end computers to interface to the existing CAMAC I/O system, replacement of the user interface by graphical synoptic diagrams running on Windows NT PCs, development of an ActiveX control for parameter display/control and a cache server to reduce loading on the rest of the control system. This paper describes the major components of the project, the techniques used to manage the new PCs and discusses some of the problems encountered during development.

1 INTRODUCTION

One of the last stages in upgrading the SRS control system to a modern, distributed, client-server design has been the replacement of the experimental beamline control system [1,2]. Previously, this consisted of a single 32-bit minicomputer interfaced to the plant via a serial CAMAC highway and 4 CAMAC crates. The user interface was a simple character based application running on the central minicomputer with remote terminals connected through serial RS-232 lines. This arrangement provided a functional and flexible solution but the user interface in particular was difficult to use and understand.

In the upgraded system, a Front End Computer (FEC) directly controls each CAMAC crate through a Hytec 1330 PC/CAMAC interface.[1] The FECs are industrial PCs running Linux. They are attached to the Control System network and handle monitoring and control requests from remote clients.

Each experimental station has been provided with a standard desktop PC running Windows NT. These provide the user interface through graphical applications that directly show the physical layout of the beamline together with real-time analogue and status information.

2 NEW DEVELOPMENTS

This system was based on a solution already in use on several of the newer experimental stations [3]. However, several changes were needed to allow easier migration from the old control system. Similarly, several improvements and technical developments were introduced during the project and new security and system management issues had to be addressed.

2.1 User Interface PCs

Previously, the beamline users had a simple text-based interface to the control system. This was provided through dedicated RS-232 serial lines and was coordinated by software running on a centralized minicomputer. It required the user to have a detailed understanding of the beamline and had little visual feedback to alert the user to potential problems on the beamline.

Each experimental station already had some form of data acquisition computer so the possibility of using this for control purposes was considered. Unfortunately, there is little standardization of these systems across the experimental stations and this alternative would have involved developing and supporting several different solutions. Instead it was decided to provide a dedicated desktop PC running Windows NT Version 4.0 for each station.

2.2 Linux Front End Systems

The client-server protocol currently used on the SRS runs over UDP/IP and has already been implemented on MS-DOS, OS-9 and Windows NT. The main server program, rpcserv, was ported to Linux and device support for both Borer and Hytec CAMAC stepper motor modules was added. Analogue input and output device support together with support for the status control and interlock monitoring system already existed. The support software for some devices requires precise delays of the order of a few 10s of microseconds to be generated in software. Initially problems were encountered due to the multi-tasking behaviour of Linux. However, careful coding and

[1] http://www.hytec-electronics.co.uk/1330.html

use of the round-robin scheduling algorithm eventually solved this problem.

RedHat Linux 6.2 provides a very stable and versatile platform for a FEC. At the time of writing some systems have been operational for over 6 months with no unscheduled reboots.

2.3 ActiveX Control

Early in the project, an ActiveX control was developed to simplify the representation of control devices in graphical user interface software. This control was written using Visual Basic 6 and is capable of handling communication with the device, colour-coded display of status, formatted display of analogue values, interlock warning messages and status control (Figure 1). This control has also proved to be useful in other projects and has also been used by the Accelerator Physics group.

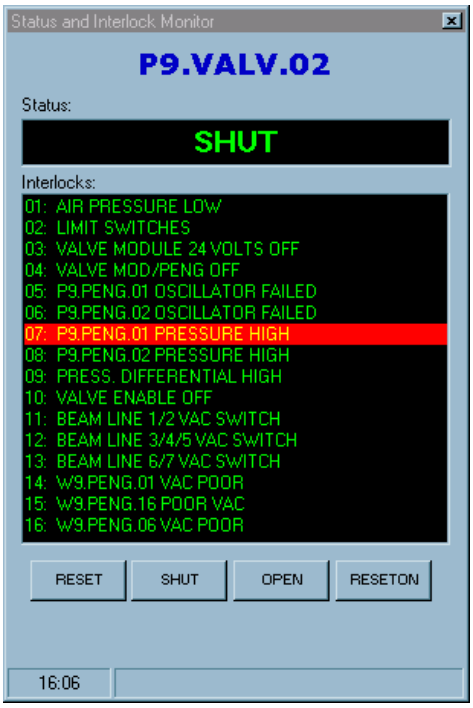


Figure 1: ActiveX control dialog box

2.4 Synoptic Control Applications

A graphical synoptic diagram showing the layout of relevant beamline components provides real-time read-back and active control for each experimental station.

These applications were developed using Visual Basic 6 and the ActiveX control described in 2.2 above. As well as the status of the local beamline components, the application displays the state of the associated beam port and useful storage ring beam parameters (Figure 2). The displays closely mimic the vacuum flow drawings for the beamline and are produced in cooperation with a beamline control engineer. Online help is available from the application. A contract programmer was hired to do the routine work of implementing and testing the applications.

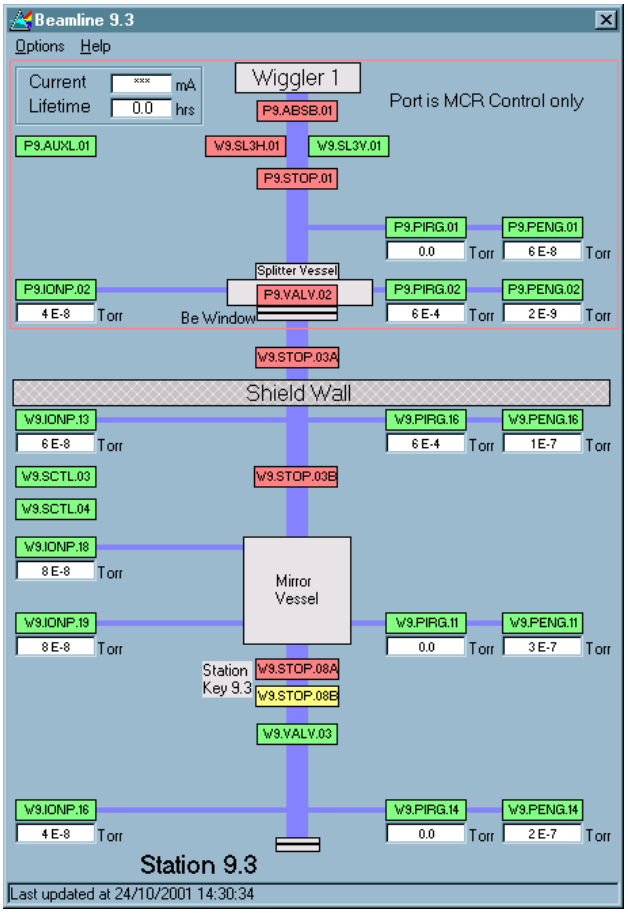


Figure 2: A typical synoptic application

2.5 Network upgrade

The SRS control system utilizes 2 class-C networks, one for high-reliability FECs and one for general server and console systems. Fibre optic and twisted-pair cabling had to be installed throughout the experimental area to provide suitable access points for the user interface PCs and Linux FECs.

2.6 Cache Server

Initial tests showed that the load on the main control system from 30 or more applications, each requesting port and beam parameters every couple of seconds, was reaching unacceptable levels. To overcome this problem a cache server was introduced. This server was prototyped in Visual Basic using COM objects and subsequently key components have been coded in C++ to provide extra performance and stability. The cache effectively reduces the load on the control system by performing a single read per parameter and serving the result back to each application. The refresh rate is configurable but is typically set at 10 seconds.

2.7 Security

Security is provided by limiting user access to the front end computers and by controlling the software allowed to run on the NT workstation.

User access is limited using the NT workstation local policy. A unique user ID is assigned to each experimental station. Only that user ID or a system administrator is allowed to log on locally. The unique ID logs on automatically when NT is started. No network access to the workstation is allowed. The floppy disk and CD ROM drives are disabled in the BIOS, and access to the BIOS protected by password.

Using an NT system policy, the allowed software for each unique user ID is controlled. The allowed software consists of the synoptic control for that specific experimental area, and a means of viewing the status of any other named parameter on the control system. To prevent users from changing the system configuration, the desktop context menus (mouse right click), access to the registry, and access to Explorer, are all disabled.

2.8 System Management

Once the accounts are set up and working, there is little on-going management to perform. New accounts are created as experimental stations come on-line, and a database of machines and user IDs maintained.

Software updates required on the local hard disk (for example the ActiveX control) are done by visiting the PC and logging on locally, rather than updating over the network. This means that the use of the PC as an active control station is not compromised by remote administration.

3 COMMISSIONING

Installation and commissioning of the system had to be completed without any interruption to routine operation of the SRS and its experimental stations. This was achieved by adopting a staged programme of work. Initially, a simple beamline with only 3 experimental stations was transferred to the new system. This allowed us to gain experience in using and managing the new software and hardware before attempting upgrade of further stations. We also became familiar with the likely problems that may occur during commissioning. The remaining 30 stations were upgraded in 3 stages over a period of 12 months.

4 FUTURE DEVELOPMENTS

There is an increasing interest among the beamline users for a facility to integrate beamline control into the experimental control and data acquisition systems. This will probably be provided in the near future by implementing a gateway server with a simple socket based interfaced. This will use an ASCII protocol for sending and returning data to and from the control system. A security database will limit access to devices based on the IP address of the requesting host and the parameter being accessed. This facility will allow much greater automation of the data gathering process and simplification of the beamline user's task.

Also, there are plans to improve the online help system. At present, only basic operating instructions are available but complete beamline specific information, fault diagnosis instructions and operating procedures could easily be added.

5 CONCLUSIONS

The upgrade of the photon beamline control system was one of the last steps in replacing obsolete equipment in the SRS control system. It has now been completed and has brought about a significant improvement in control facilities on the experimental stations. All the changes were implemented during routine shutdown periods and no disruption to normal operating time occurred.

Some of the developments undertaken for the project, notably the Linux FEC design and the ActiveX control have wider use in the SRS control system and will prove useful in future projects.

REFERENCES

[1] B.G.Martlew, M.J.Pugh, W.R.Rawlinson, "Planned Upgrades to the SRS Control System", EPAC'94, London, June 1994.

[2] M.T.Heron, B.G.Martlew, A.Oates, W.R.Rawlinson, "An analysis of the SRS Control System Upgrade", PCaPAC'00, DESY, Hamburg, October 2000.

[3] B.Corker, M.T.Heron, B.G.Martlew, W.R.Rawlinson, "A New Control System for Beamline 5D on the SRS at Daresbury", EPAC'98, Stockholm, June 1998.

WEAT005
physics/0111131

UPGRADE OF LINAC CONTROL SYSTEM WITH NEW VME CONTROLLERS AT SPRING-8

T. Masuda, T. Ohata, T. Asaka, H. Dewa, T. Fukui, H. Hanaki, N. Hosoda, T. Kobayashi, M. Kodera, A. Mizuno, S. Nakajima, S. Suzuki, M. Takao, R. Tanaka, T. Taniuchi, Y. Taniuchi, H. Tomizawa, A. Yamashita, K. Yanagida
SPring-8, Hyogo 679-5198, Japan

Abstract

We integrated an injector linac control system to the SPring-8 standard system in September 2000. As a result of this integration, the SPring-8 accelerator complex was controlled by one unified system. Because the linac was continuously running as the electron beam injector not only for the SPring-8 storage ring but also for NewSUBARU, we had to minimize the hardware modification to reduce the time for the development and testing of the new control system. The integration method was almost the same as that of the integration of the booster synchrotron. We report here on the integration of the linac control system with emphasis on the upgrade of the VMEbus controllers and software involving the operating system Solaris 7 as the real-time OS.

1 INTRODUCTION

At the beginning of the SPring-8, accelerator control systems for a linac, a booster synchrotron and a storage ring were designed and constructed independently. Since the storage-ring control system had been defined as the SPring-8 standard, an integration of the synchrotron control system was planned at first and finished in January 1999 as already reported [1]. A linac control system remained to be integrated to the standard system. As already shown in the previous work, the integration of the control system enabled seamless operation between the storage ring and the synchrotron, and made the efficient development of software possible.

The linac control system was originally designed in 1991 [2]. A total number of 25 VME systems were controlled by MVME 147SA-1 (Motorola 68030 CPU) boards with OS-9. The user interface software was developed with Motif on HP-UX. A special protocol was developed and used for the communication between VME systems and a control server workstation. Several X-terminals were connected to the server as the operator consoles.

Recently, system maintenance became more complex and the lack of a data logging system made machine diagnostics difficult.

In September 1999, we started to upgrade the present linac system to the new one by replacing a part of hardware and introducing a standard software scheme as described in below. At this time, we also introduced the standard database system for the linac control [3,4].

Because the linac played a role as an injector to the booster synchrotron and the NewSUBARU storage ring, it was necessary to avoid downtime due to the integration procedures. We separated the hardware upgrade work into two phases, that is, we replaced the system with minimum modification at the first phase and postponed an overall replacement to the near future. In the first phase, we replaced only CPU boards for the VME systems keeping the rest of the hardware in place. On the other hand, all the control software was newly developed, and the standard TCP/IP protocol was introduced.

We started preparations in September 1999 by setting the start time of the integration for the July 2000 shutdown period, and finally the new linac system started operation in September 2000.

2 NEW CONTROLLER

2.1 VMEbus CPU Board

As already described [3], we used Hewlett-Packard HP743rt's for the VMEbus controllers of the storage ring and the booster synchrotron [1]. Because the HP743rt model had been discontinued and it was not possible to purchase, we had to select a new CPU board considering the following criteria:

- The board should be designed on PC-base architecture. The processor unit should be Intel-Architecture (IA-32) or IA-32 compatible.
- The board should support widely used operating systems (OS). An OS with open source policy or a free license is preferred.
- Minimum running costs with easy maintenance and long product lifetime.

Additionally, we required that the board should be a VMEbus single-slot (if possible) or at most a double-slot with a storage unit, bootable from a flash disk, no cooling fan attached to the CPU, and less power consumption to

avoid overheating. Usually IA-32 based board is designed on PCIbus base but it can be applied to the VMEbus with a PCI-VME bus-bridge chip (Tundra Universe II). By examining candidate boards, we finally chose a Xycom XVME658 CPU board as the VMEbus controller. The features of the board are:

- CPU is an AMD K6-2 333MHz.
- Bootable from an IDE flash disk.
- Single slot but we use one more slot for the storage units such as the flash disk and a floppy drive.
- No CPU-cooling fan.

It was also essential to decide what OS should be used and examine its operability on the board before we came to the board final selection.

2.2 Solaris Operating System

As soon as we had chosen the IA-based CPU, we examined candidate operating systems such as, Linux, RT-Linux and Solaris. To port the current software, the compatibility of UNIX function calls is highly required for the OS. The rigid priority control of software processes is also essential, because it determines controllability of the system under the client/server multi-task architecture. A hard real-time feature such as the absolute deterministic operability is not necessary for our accelerator controls. Task switching latency, data transfer speed and interrupt response time due to a bus-bridge chip were studied as well on the bases of candidate CPU boards [5]. In our study, the Solaris 7 and the Linux OS with a kernel 2.2.9 showed good performance.

The Solaris has three scheduling policies, i.e. a system class (SY), a real-time (RT) class and a time-sharing (TS) class. These classes have a scheduling hierarchy structure as RT>SY>TS. Even in the TS class, the fixed priority control is possible and satisfactory. The RT class is available but its scheduling policy is beyond the system class. So system management with the RT-class is difficult. Because when we lose the control of a process in the RT class the system may hang.

Examining the survey and measurements of the various OS features, we finally chose the Solaris 7 as the OS. We used the TS class for the process scheduling.

3 SOFTWARE DEVELOPMENT

3.1 Methodology

The equipment experts developed device control software (Equipment Manager, EM [3]) running on the VME systems and graphical user interface (GUI) software on the operator consoles. The control group provided the software framework and program templates. The GUI builder tool and language were same as those of already mentioned [3]. The integration work is summarized as follows:

- Rearrangement of operation sequences and list up of equipment signals to make abstract commands (S/V/O/C format [3]) took three man-months.
- Development of the EM (GUI) software took twelve (eight) man-months, respectively.
- Database parameter set-up and making of access functions took one man-month.

The prototyping methodology was efficient for the rapid development of both EM and GUI programs. The device simulation codes were used for the testing and debugging of the EM, and as a result, the program tests using real devices ran very smoothly.

3.2 GUI

The man machine interface for the linac control is categorized in two groups of GUI called *operation panel* and *maintenance panel*. The former is used for machine tuning and leads the beam into five beam transport lines. The later is used for the maintenance of individual equipment.

The linac can be controlled using five *operation panels*, i.e. *top panel*, *gun control*, *RF control*, *magnet control*, and *vacuum operation*. The top panel was newly created on this upgrade as shown in Figure 1. The beam route can be switched by using this panel. The switching procedure includes sequences such as degaussing the bending magnets, opening/closing the beam shutters and setting the transport parameters. Because this procedure was rather complicated, the switching operation took a long time and recorded mis-operations in the previous system. But now, this time has been greatly reduced by introducing this panel.

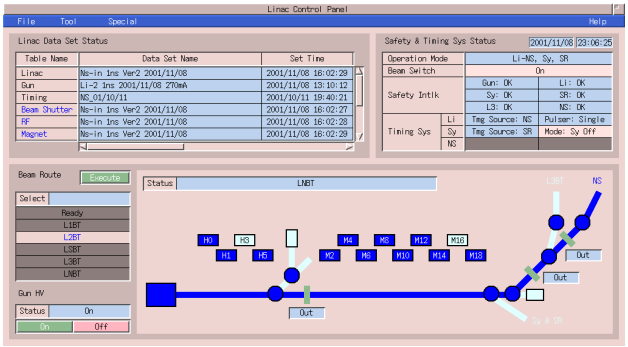


Figure 1: The top panel of the linac control.

As an example of equipment control panels, an RF control panel is shown in Figure 2. The symbols showing machine components are located in the panel based on a real machine layout. Using the panel, the machine operator tunes the beam energy and orbit by monitoring a beam image from profile monitors located along the linac.

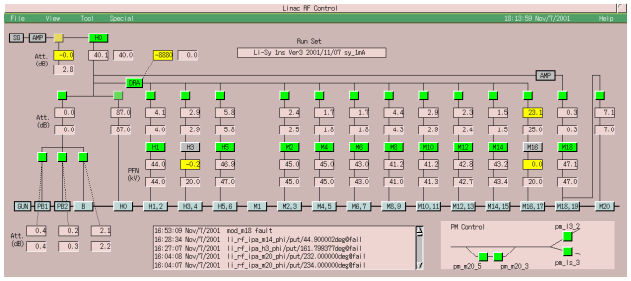


Figure 2: The RF control panel.

As a result of the replacement, the response time between a GUI action and a device reply has been reduced to about 20msec, while it was in the order of a second in the former system. The other significant improvement is that the GUI can be built without knowledge of X/Motif programming. This freedom of development will facilitate an advanced control of the machine.

3.3 Database System

The linac has over 270 setting values. A relational database management system consistently manages those values [4]. A total of 84 newly created tables in the database hold a set of parameters. A data acquisition process collects over 5.2kB of data from the linac every five seconds. The database system also keeps this data in 18 tables.

Though the linac control system needed over 100 new tables, the database system minimized the software effort to add them to the database with its standardized procedure.

Now, the linac machine status and logging data can be retrieved from the database, and we can monitor the data via WWW from anywhere in the site. Figure 3 is an example of the machine status monitoring by a WWW browser.

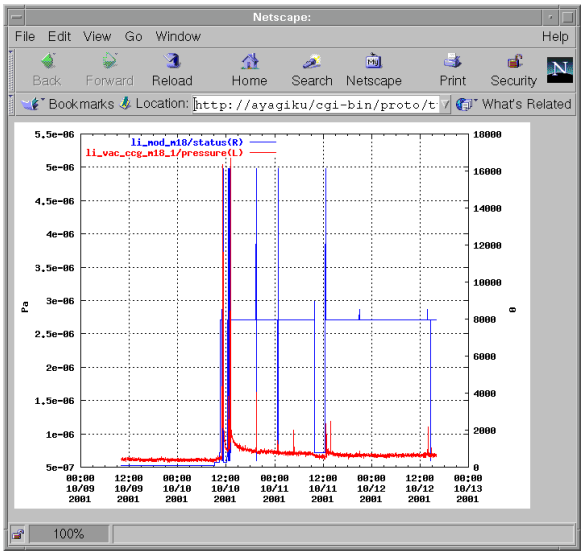


Figure 3: Example of the linac status monitoring with a WWW browser.

4 INSTALLATION

We started installation of the new CPU boards in July 2000. After the replacement of the CPU boards, we had some problems apparently caused by electric noise and/or the CPU overheating. The new system was first applied to the aging of the klystrons in the middle of August. During the tuning of the system, the operation of CPU boards stopped either suddenly or relating to the klystron modulator breakdown. Because the VME racks were settled close to the RF modulators, electric noise was the first suspect.

We found that the previous system was not strong enough against the line noise, and also that the cooling system of the VME boards was not well optimized to cool the CPU running at high frequency. After we improved the power line isolation and the cooling system in August 2001, the VME system recorded no trouble for two months.

The Solaris 7 with the TS class has worked well since the installation. It schedules multi-tasks with the rigid priority control and exclusion control at the same level already shown by the HP-RT real-time OS.

5 SUMMARIES

We replaced Motorola 68030 CPU boards running OS-9 with new IA-based CPU boards. A set of the XVME658 with Solaris 7 was chosen as the controller for 25 VME systems. We replaced all the control software. The equipment control software on the VME was newly created with the standard framework, and the GUIs for the operator consoles were also developed according to the unified panel look and feel.

We finished the integration of the linac control system to the SPring-8 standard system in September 2000. Now, the SPring-8 whole accelerator complex and beamlines are operated by the unified control system. The new VME system showed a good performance and adequate reliability. The data logging system with the database proved its usefulness again for the machine diagnostics by searching over buried signals.

At this phase, we minimized the hardware modification. However, we are planning to update the VMEbus I/O systems to be better able to withstand against the electric noise and compact enough for easy maintenance in the next phase.

REFERENCES

[1] N. Hosoda *et al*., "Integration of the Booster Synchrotron Control System to the SPring-8 Control System", Proc. of ICALEPCS'99, Trieste, Italy, 1999, p 93.

[2] H. Sakaki *et al*., "Design of SPring-8 Linac Control System Using Object Oriented Concept", Proc. of 1995 Part. Accel. Conf., Dallas, 1995.

[3] R. Tanaka *et al*., "The first operation of control system at the SPring-8 storage ring", Proc. of ICALEPCS'97, Beijing, China, 1997 p.1

[4] A. Yamashita *et al*., "Data Archiving and Retrieval for SPring-8 Accelerator Complex", Proc. of ICALEPCS'99, Trieste, Italy, 1999, p 434.

[5] T. Ohata and T. Masuda, "A study of a real-time operating system on the Intel-based VME controllers", in these proceedings of this conference.

Wednesday, 28 November 2001

WEB - Real-Time OS, Linux in Control Systems, Emerging Trends

WEBI001
cs.OS/0111035

OPEN SOURCE REAL TIME OPERATING SYSTEMS OVERVIEW *

T. Straumann, SSRL, Menlo Park, USA

Abstract

Modern control systems applications are often built on top of a real time operating system (RTOS) which provides the necessary hardware abstraction as well as scheduling, networking and other services. Several open source RTOS solutions are publicly available, which is very attractive, both from an economic (no licensing fees) as well as from a technical (control over the source code) point of view. This contribution gives an overview of the RTLinux and RTEMS systems (architecture, development environment, API etc.). Both systems feature most popular CPUs, several APIs (including Posix), networking, portability and optional commercial support. Some performance figures are presented, focusing on interrupt latency and context switching delay.

1 INTRODUCTION

Apart from hard-real time interrupt handling and scheduling services, there are other OS features of interest, such as the available APIs, target CPU architectures and BSPs (board support packages), support for multiple processors, networking, file systems, dynamic object loading, memory protection and so on.

Other important issues are licensing terms, development environment and debugging tools and the availability of these tools for specific host platforms.

The next section gives an overview of RTL and RTEMS looking at some of these issues. In section 3, some performance measurements are presented comparing the results for RTL and RTEMS to a commercial system (vxWorks).

2 RTLINUX AND RTEMS OVERVIEW

2.1 RTLinux

General Information RTL development started at the New Mexico Institute of Mining and Technology and is now maintained by FSMlabs Inc. which also offers commercial support. The basic mechanism is protected by a US patent; RTL (having a license for using the mechanism) itself is licensed under the terms of the GPL.

RTL is distributed as a patch against certain versions of Linux and a collection of kernel modules.

Further information about RTL is available at [1].

System Architecture The basic idea of RTL is striking simple: A slim layer of software is "hooked" into standard Linux' interrupt handlers and interrupt enabling/disabling primitives, thereby effectively taking over the machine which is then managed by a special real-time scheduler. Linux continues to run as a low priority task.

The real-time core manages all hardware interrupts, dispatching them appropriately, either to Linux or to real-time threads. The interrupt manager never allows Linux to disable interrupts. Instead, Linux disabling an IRQ actually invokes an RTL hook which marks the target interrupt as "disabled". If the interrupt manager detects such a marked IRQ, it holds off dispatching it to Linux until the corresponding call to re-enable the IRQ in question is intercepted.

Linux being a low priority task with no direct access to the interrupt hardware implies that any real-time thread introduced into the system can only very weakly interact with Linux (through special communication channels) and may only build upon the services of the RTL core, such as synchronization primitives and the scheduler. Note that this low-level environment does not provide a C or "math" library nor any of Linux' standard system services like networking, file systems or drivers. While the former (C library) functionality is easy to add, providing the latter is far more complex for obvious reasons. Communication with user space processes is established through special "real-time fifo" devices.

Following the philosophy of RTL, most of an application should be implemented in user space, as ordinary Linux programs. Only the real-time critical tasks go into a special module which is loaded into kernel memory using Linux' standard kernel-module loader.

API and General Features Version 3.1 of RTL offers (a subset of) the POSIX (1003.13) "pthreads", semaphores, condition variables and a proprietary interface to the interrupt subsystem. As already mentioned, no C-library is provided per-se. RTL special features include periodic scheduling with high timing resolution.

While high timing resolution is certainly desirable, its usefulness is reduced to some degree by the relatively high latencies (see measurement section).

As a consequence of the layered system architecture with Linux on top of the RTL core, an RTL application must be carefully separated into real-time critical and non-critical parts. Only the latter may use the powerful features of

* Thanks to Ric Claus for kindly borrowing me the MVME2306 computer

Linux, both in kernel or user space. Critical tasks *must not* e.g. write files or access non-RT drivers but they must delegate this work to non-real time code.

Supported Target Architectures RTL supports a subset of the CPUs and platforms supported by Linux. x86, PowerPC, Alpha and MIPS are currently supported by RTL; at least on x86, SMP is supported.

Development Environment RTL development is usually done using the well-known GNU tool chain which has been ported to a wide variety of host platforms. The RTL core provides support for debugging real-time modules.

2.2 RTEMS

General Information RTEMS stands for "Real Time Executive for Multiprocessor Systems", where the original meaning of the letter *M*, namely "Missile" and later "Military" has eventually reached a civilian status.

RTEMS was developed by OaR Corp. on behalf of the US DoD and is licensed under a GPL variant. OaR coordinates development efforts and offers commercial support for RTEMS and other related services.

RTEMS has reached production quality and is used by military, industrial and scientific projects. EPICS, a control systems software which is widely used in the accelerator community, has been ported to RTEMS as of the new 3.14 EPICS release.

More information about RTEMS can be found at [2].

System Architecture RTEMS was designed as a true RTOS from scratch, targeting embedded systems, possibly with less memory. Consequently, various system components are partitioned into separate modules ("managers" in RTEMS terminology) which are linked to the application as needed. The system can further be tailored to an application's specific needs by choosing appropriate configuration parameters.

A typical RTEMS application is built by compiling the application itself, which must provide the necessary configuration parameters, and linking it to the desired RTEMS managers (which are provided in libraries) thereby creating an executable for downloading to the target system or burning into ROM etc.

Since RTEMS is an RT system "from the ground up", all system services and libraries are directly available to any application task.

API and General Features RTEMS features POSIX (1003.1b), ITRON and "classic/native" APIs in C and ADA (native API only) language bindings. The usual components of an RTOS are available, such as multitasking (thread creation and control), synchronization primitives (mutexes, semaphores, message queues, events etc.), schedulers (fifo/round robin, rate monotonic), clocks etc.

RTEMS provides a port of the BSD TCP/IP networking stack and supports multiple (possibly heterogeneous) CPUs.

As in vxWorks, memory protection is not available; the system and application software share the same, flat memory space.

RTEMS itself does not ship a shell as powerful as vxWorks' nor does it offer a dynamic loader. However, there are ongoing efforts of creating application programs providing the respective features.

The only file systems currently implemented are a remote TFTP and a "in memory" (ramdisk) file system.

Supported Target Architectures RTEMS is designed to be easily portable and consequently it supports many CPU architectures, such as m68k, ColdFire, Hitachi SH, Intel i386, Intel i960, MIPS, PowerPC, SPARC, AMD A29k and HP PA-RISC.

Development Environment RTEMS uses the GNU tool chain.

3 RESPONSE TIME PERFORMANCE TEST

A key property of any hard-real time system is its "response time", i.e. the time it takes for the system to react to some external event under worst case conditions. Two important terms shall be defined here:

"Interrupt Latency" The time it takes from a device asserting an interrupt line until the system dispatching the corresponding interrupt handler (ISR) shall be called *interrupt latency*.

"Context Switch Delay" This term defines the time it takes to schedule a task. It involves the scheduler determining which task to run, saving the current task context and restoring the new one.

Of course, it is practically impossible to find the worst case conditions given the huge number of possible state combinations that can occur in a computer system.

Therefore, a statistical approach is taken to create "worst case" conditions. The idea is to let the system operate under heavy load for some time while measuring the latencies. The maximal delay recorded during the test is then assumed to reflect the "worst case".

3.1 Test Algorithm

A PowerPC 604 CPU (300MHz) on a MVME2306, PReP compatible board by Motorola was chosen to perform the measurements. BSPs for RTL, RTEMS and VxWorks were available, allowing for comparison of the three systems on the same target hardware.

The MVME2306 (like most PPC platforms) features timer hardware with a reasonable resolution, which can be set up to generate periodic interrupts. Because the running timer is readable "on-the fly", a precise measurement of latencies can easily be accomplished.

The test software package [3] consists of an initialization routine, an interrupt handler (ISR) and a simple "measurement" procedure.

The initialization code sets up the timer hardware, connects the ISR to the respective interrupt and spawns a task (MT) executing the measurement procedure at the highest priority available on the system under test.

The ISR determines the interrupt latency by reading the timer and notifies the MT by releasing a semaphore on which the MT blocks. This causes the system to schedule the MT (having become the highest priority runnable task), which, reading the running timer is able to determine the time that elapsed from the ISR releasing the semaphore until the MT actually getting hold of the CPU. After recording the delay, the MT again blocks on the semaphore.

This simple test was performed on a system heavily loaded with low priority tasks, networking and serial I/O traffic causing a large volume of interrupts (also at a priority lower than the timer hardware IRQ).

According to the definition, a hard-real time system must guarantee that the latencies experienced by the high priority ISR and MT stay below a certain *hard* limit, regardless of the amount of low-priority load (note that interrupts inherently have a higher priority than any normal task, hence a low-priority interrupt still interrupts a high-priority task).

Hence, the maximal recorded latencies during the test constitute a measure for the quality of a given system.

3.2 Results

The test was performed on the same hardware under the RTL, RTEMS and VxWorks systems. 2'000'000 timer interrupts were generated at a rate of 4kHz and the maximal and average latencies were recorded. Measurements were made under both, idle and loaded conditions.

The load that was imposed on the system under test consisted of "flood pinging" its network interface from a host computer, while letting a low priority thread copy characters from a TCP socket (connected to the host's "chargen" port) to the serial (RS-232) console. Thus, the loaded system was subject to heavy interrupt and kernel activity involving scheduling, synchronization primitives, networking and driver code sections among others.

The results are shown in Tab. 1. The idle systems all exhibit comparable figures. The situation changes, however, quite dramatically under load: Whereas RTEMS and VxWorks show similar performance, RTL's latencies are substantially higher on the loaded system. This is not really surprising given the far more complex interrupt dispatching that is needed to manage and emulate the Linux interrupts.

Somewhat surprising is RTEMS' increased scheduling latency when using the pthread API, as one would assume the implementation to merely consist of an inexpensive wrapper to the native API. Given the good performance of the latter, one can expect however, that making improvements should be relatively straightforward.

As can be seen, the average latencies are about an order of magnitude less than the respective maxima. Although our statistical test gives some lower bound of the maximal latencies, it is impossible to draw conclusions about the true worst case figures which are obviously extremely difficult to establish.

	Interrupt Latency		Context Switching	
	max	avg$\pm\sigma$	max	avg$\pm\sigma$
		Idle System		
RTL	13.5	(1.7$\pm$0.2)	33.1	(8.7$\pm$ 0.5)
RTEMS[1]	14.9	(1.3$\pm$0.1)	16.9	(2.3$\pm$ 0.1)
RTEMS	15.1	(1.3$\pm$0.1)	16.4	(2.2$\pm$ 0.1)
vxWorks	13.1	(2.0$\pm$0.2)	19.0	(3.1$\pm$ 0.3)
		Loaded System		
RTL	196.8	(2.1$\pm$3.3)	193.9	(11.2$\pm$ 4.5)
RTEMS[1]	19.2	(2.4$\pm$1.7)	213.0	(10.4$\pm$12.7)
RTEMS	20.5	(2.9$\pm$1.8)	51.3	(3.7$\pm$ 2.0)
vxWorks	25.2	(2.9$\pm$1.5)	38.8	(9.5$\pm$ 3.2)

[1] using pthreads

Table 1: Latency measurement results. All times are in μs. vxWorks and RTEMS use native threads unless otherwise noted. RTL uses the pthread API.

Usually, the interrupt handling parts of any system are highly hardware-architecture dependent. Therefore, while representative for the PowerPC, the interrupt latency figures stated here can not easily be generalized to other CPU architectures.

4 CONCLUSION

RTEMS and RTL are two quite different open-source RTOS solutions.

RTEMS seems to offer both, core features and performance which are comparable to a commercial system like vxWorks.

RTL could be interesting in situations, where the full power of a desktop system is needed, enhancing such a system by hard-real time features. This comes, however, at the expense of higher latencies (compared to RTEMS or vxWorks) and limitations of system services that are available to the real-time tasks.

Finally, it should be noted, that the simple benchmark presented in this paper does by no means constitute a thorough performance evaluation and comparison, an arduous task to which the interested reader is encouraged to contribute.

5 REFERENCES

[1] http://www.rtlinux.com

[2] http://www.rtems.com

[3] http://www.slac.stanford.edu/~strauman/rtoslat/

WEBT002
physics/0111041

EPICS: A RETROSPECTIVE ON PORTING iocCore TO MULTIPLE OPERATING SYSTEMS

M.R. Kraimer*, J.B. Anderson*, ANL, Argonne IL 60439, USA
J.O. Hill, LANL, Los Alamos NM 87545, USA
W.E. Norum, University of Saskatchewan, Saskatoon, Canada

Abstract

An important component of EPICS (Experimental Physics and Industrial Control System) is iocCore, which is the core software in the IOC (input/output controller) front-end processors. At ICALEPCS 1999 a paper was presented describing plans to port iocCore to multiple operating systems. At that time iocCore only supported vxWorks, but now it also supports RTEMS, Solaris, Linux, and WinNT. This paper describes some key features of how iocCore supports multiple operating systems.

1 INTRODUCTION

Originally input/output controller (IOC) meant a VME/VXI-based system, which used the vxWorks operating system. The ICALEPCS99 paper [1] described the plan for removing the VME/VXI and vxWorks dependencies. The basic plan was successfully followed, although many details changed.

The iocCore port involved the following major changes to EPICS Base:

- VME/VXI dependencies – These were solved by unbundling all hardware-specific support, i.e., it is all now built as separate products. This support can still be used on VME/VXI vxWorks systems.
- VxWorks libraries – This was solved by defining operating system independent (OSI) interfaces. Generic or operating-system-specific code implements the interfaces.
- Registry – vxWorks provided a facility that allowed iocCore to locate record/device/driver support during initialization. The registry now provides the same facility.
- Build Environment – The old build environment had separate makefiles for the host and for vxWorks targets while the new build environment uses a single makefile for all targets.
- Shell – iocsh together with the registry provides features previously supplied by the vxWorks shell.
- Interrupt level support – vxWorks made it very easy to interact with interrupt handlers. Parts of the original iocCore required interrupt level support. The new iocCore removes these requirements.

* Work supported by U.S. Department of Energy, Office of Basic Energy Sciences under Contract No. W-31-109-ENG-38.

2 OVERVIEW OF CHANGES

2.1 OSI Interfaces

Table 1 provides a brief summary of the OSI interfaces used by iocCore. The last column shows the different implementations required for the supported platforms.

Table 1: OSI Interfaces

Name	Replaces	Implementation
epicsRing	rngLib	Generic
epicsTimer	wdLib, osiTimer	Generic
epicsAssert	epicsAssert	default, vxWorks
epicsEvent	semLib	RTEMS, WIN32, POSIX, vxWorks
epicsFindSymbol	symFindByName	Default,vxWorks
epicsInterrupt	intLib	RTEMS,default, vxWorks
epicsMutex	semLib	RTEMS, WIN32, POSIX, vxWorks
epicsThread	taskLib	RTEMS, WIN32, POSIX, vxWorks
epicsTime	tickLib, osiTime	RTEMS, WIN32, POSIX, vxWorks
osiPoolStatus	memLib	RTEMS, WIN32, default, vxWorks
osiProcess	osiProcess	RTEMS, WIN32, POSIX, vxWorks
osiSigPipeIgnore	osiSigPipeIgnore	WIN32, default, POSIX, vxWorks
osiSock	osiSock	Linux, RTEMS, WIN32, default, solaris, vxWorks

2.2 Registry

vxWorks provides a function, symFindByName, which is used to dynamically locate global data and functions. This facility is unique to vxWorks and is not easily recreated in other environments. Instead a facility to register and find pointers to functions and structures is provided.

When databases are loaded, all record, device, and driver support is located dynamically. This requires that the support must first be registered. While building an IOC application, a Perl program is run. This program reads the database definition file that will be loaded at runtime by dbLoadDatabase and generates a C function to register the record, device, and driver support. This function is linked with the application and is called before IOC initialization.

2.3 Build Environment

Because iocCore is now built for multiple architectures, extensive changes were required for the EPICS build system. Although the new system has more functionality, it is simpler than the old system. In each source directory a single Makefile appears instead of three (Makefile, Makefile.Vx, and Makefile.Host).

2.4 Shell

The EPICS IOC shell (iocsh) is a simple command interpreter, which provides a subset of the capabilities of the vxWorks shell. It is used to interpret startup scripts and to execute commands entered at the console terminal. It can execute any command that is registered.

iocCore provides many test commands that are automatically registered.

2.5 Interrupt Level Support

The vxWorks intLock/intUnlock routines were an essential part of iocCore. Most operating systems do not allow such tight coupling between interrupt routines and user processes. In iocCore most code was modified so that intLock/intUnlock are no longer required. For code that still needs this capability, epicsInterrupt is provided. For operating systems like vxWorks, in which everything runs in a shared memory, multithreaded kernel environment, an implementation of epicsInterrupt is provided. For other operating systems a default version is provided, which uses a global lock (global means global to the iocCore process).

3 STATUS OF PORT

The current production release of EPICS base is EPICS base R3.13.5. EPICS base R3.14.x [2] releases are the releases containing the iocCore port.

3.1 Work Completed

The first beta release of 3.14 is now available. It supports the following targets:

- VxWorks 5.4
- RTEMS 4.6 – An open source real-time operating system.
- Solaris – Tested on Solaris 8.
- Linux – Tested on Redhat 6.2 and 7.1.
- Win32 – Tested on winNT, win98 and win2000.

3.2 Work Remaining

No new major functionality will be added to the 3.14 EPICS base releases, only bug fixes. With more testing we will be ready for the first regular release of 3.14. At this time we recommend that operational systems start migrating to 3.14.

For existing EPICS sites, one major platform is still missing: HPUX. Work is currently in progress to support HPUX11.

3.3 Hardware Support

As mentioned previously, the hardware-specific support that was included with R3.13 is now unbundled. Much of the VME support has been built and tested with R3.14 but only works on vxWorks.

An unbundled version of the sequencer, which was originally developed by William Lupton at KECK and is now supported by Ron Chestnut at SLAC, has been available for some time. This is the version that is supported with R3.14. It has been built and tested on all supported R3.14 platforms.

The unbundled version of GPIB support, which has been available from Benjamin Franksen at BESSY, is the version supported with R3.14. It has been modified to work with R3.14. It includes a driver for an Ethernet GPIB interface. With this interface iocCore GPIB support is available on all supported R3.14 platforms.

4 SUPPORTING A NEW PLATFORM

It should be easy to port iocCore to additional environments as long as the environment supports multithreading, GNU make, and Perl.

Two parts of EPICS base must be extended: configure and libCom/osi.

EPICS base has a directory configure/os. For each supported platform, this directory contains files describing how to build components for that platform.

EPICS base has a directory src/libCom/osi/os that has the following structure:

- default
- posix
- <platform specific>

The osi directory contains definitions of the OSI interfaces. The implementation of these interfaces must be provided for each platform. The implementation can be provided in three ways.

- default – The directory default provides an implementation of many of the interfaces. If a default can be used, nothing needs to be done for the new platform.
- posix - The posix directory provides implementations for several of the interfaces but requires that the platform provide both POSIX real time and POSIX threads support. If the POSIX implementation of an interface can be used, then nothing needs to be done for the new platform.
- <platform specific> - If neither the POSIX nor default implementation will work for the new platform, then a platform-specific implementation must be provided.

Table 2 gives an idea of how much work is involved to support a new platform.

Table 2: Platform-Specific Lines of Code

Implementation	**Lines of Code**
VxWorks	1283
RTEMS	1488
Solaris	89
Linux	145
Win32	3651
Posix	1262
Default	950

Lines of code include header, C, and C++ source files. The small number of lines of code required for Solaris and Linux is because both support POSIX real time and POSIX threads and both can use most of the default code. The Win32 implementation is much larger because the POSIX and most of the default code is not used.

5 OTHER 3.14 CHANGES

The Channel Access (CA) client interfaces are now thread safe on all operating systems (OSs), and the library now requires a multithreaded operating environment on all OSs. We hope that this uniformity will simplify testing and lead to better control over quality. The library now uses a preliminary plug-compatible interface when communicating with in-memory services such as the process control database. The code was restructured to not hold a mutual exclusion lock when calling user callbacks, thereby reducing subtle opportunities for deadlocks between in-memory services and the CA client library. Changes were also made to support unlimited vector lengths and client-specified dispatch priority in the server.

6 COMPATIBILITY

An important consideration in designing the iocCore port was compatibility for existing IOC applications and also for Channel Access clients.

Although EPICS base uses new build rules, the old rules are still supported, but only for vxWorks IOC applications. This allows easy conversion of existing applications to convert from R3.13 to R3.14.

Even though Channel Access was completely rewritten, the old CA client interface is still supported. Thus most existing CA clients are supported without any modifications.

7 CONCLUSION

The source code for EPICS iocCore was reorganized so that it can now be easily adapted to any operating environment with support for multi-threading. A new beta release is now available with these changes and other important upgrades including an improved build environment, a new command line shell, and communication of unlimited length arrays.

REFERENCES

[1] M. Kraimer, “EPICS: Porting iocCore to Multiple Operating Systems,” Proc. of the 1999 ICALEPCS Conference, Trieste, Italy, pp. 33-35, 2000.

[2] http://www.aps.anl.gov/epics/modules/base/R3-14.php

L4-LINUX BASED SYSTEM AS A PLATFORM FOR EPICS IOC-CORE

J. Odagiri, N. Yamamoto and T. Katoh, KEK, Oho 1-1, Tsukuba, JAPAN

Abstract

The EPICS Input/Output Controller (IOC) core-program, iocCore [1], is now portable to multi-platforms. The Linux operating system, among them, seems to be a promising candidate for a platform to run iocCore, considering the recent high appreciation in desktop and server use as well as control fields.

The Linux kernel, however, is not suitable for time-critical applications, since it responds to external events with unpredictable latency. We summarize three known causes of the latency, and then discuss some of the different solutions and how they affect the functionality of iocCore.

As a possible alternative, we propose an approach that dispatches user-level processes by a real-time kernel aiming at a consistency of availability with predictable responsiveness.

1 INTRODUCTION

In EPICS 3.14, the latest version, iocCore can run on Linux with OSI (Operating System Interface) libraries supplied as a part of the distribution [1]. The OSI libraries interface iocCore and POSIX threads (POSIX 1003.1c) with a real-time extension (POSIX 1003.1b).

However, the real-time extension can support only so-called “soft” real-time applications, where relatively rare misses to the deadline can be acceptable. It applies even if the POSIX threads are implemented on top of kernel-level threads because the sources of the unpredictability are inherent in the Linux kernel, itself.

In section 2, the causes of unpredictable latency in the Linux kernel are summarized. Section 3 describes the advantages and disadvantages of some of the real-time extensions of Linux in view of the availability to iocCore. In section 4, we propose a solution based on L4-Linux, a port of the Linux kernel onto a real-time kernel named L4 [2]. Conclusions are given in Section 5.

2 SOURCES OF UNPREDICTABLE LATENCY

There are three sources of unpredictable latency in the Linux kernel with widely different impacts: non-preemptive kernel, interrupt disabling and address space switching.

2.1 Non-preemptive Kernel

A process under Linux runs in kernel-space while it executes a system call. The term “non-preemptive kernel” implies that the Linux kernel does not switch the execution from a process in kernel-space until it invokes the scheduler in its own context. Real-time kernels, however, must switch the execution upon return from an interrupt handler if the interrupt gets a higher priority process ready to run. The Linux kernel does not allow preemption in the kernel, since it cannot protect the consistency of kernel data if multiple processes concurrently run in the kernel. A higher priority process that gets ready to run has to wait until the process currently running in the kernel exits from the kernel. While typical latency due to non-preemptiveness is known to be several tens of milliseconds, it can reach 100 milliseconds or more in the worst case [3, 4].

2.2 Interrupt Disabling

Mutual exclusion to keep the consistency of kernel data is required between a process in the kernel and an interrupt handler, and between interrupt handlers as well. Since interrupt handlers can not sleep, the only way to achieve it is to disable interrupts while the process or an interrupt handler executes the critical section. Even though the typical execution path of such critical sections is very short compared to the whole execution path of a system call, it can reach, in time, up to several hundreds of microseconds [5]. It can even be much longer in special cases [4].

2.3 Address Space Switching

Linux gives each process an independent address space, in more concrete terms, a set of mapping tables to translate a virtual address into a physical address. Some of the entries of the tables are cashed in a Translation Look aside Buffer (TLB) of the Memory Management Unit (MMU). Upon process switching, all of the cashed entries need to be flushed to invalidate the old mapping. The process newly switched in starts without having the cashed entries, and then cases TLB misses as it evolves. This brings in unpredictable latency of tens of microseconds.

3 POSSIBLE SOLUTIONS

3.1 Real-time Tasks in Linux Kernel Space

Some approaches, such as RTLinux [5] and RTAI [6], introduce a hardware abstraction layer, a small kernel, under the Linux kernel. The real-time kernel, as well as its tasks, resides in Linux kernel space, and it takes complete precedence on interrupt management and CPU scheduling over the Linux kernel. This approach is free from all of the three causes of unpredictable latency. First, the real-time tasks have essentially nothing to do with the non-preemptive-ness of the Linux kernel because they do not call for Linux services. Second, the Linux kernel is not allowed to disable interrupts in this system. Third, the real-time tasks run in the Linux kernel space, which has been mapped into address spaces of every Linux process. No matter which process an external event interrupts, the real-time tasks are ready to handle it without switching any address spaces.

In this approach, however, all iocCore programs, including the run-time database, need to be moved into the Linux kernel in order to use real-time tasks for the threads of iocCore. It seems not only unrealistic, but also unfavorable, since debugging an application easily crashes the Linux kernel. On the other hand, if iocCore resides in user-space, it must run under the standard Linux kernel.

3.2 User processes Under Improved Linux

If the causes of latency, at least the dominant latency that stems from the non-preemptive-ness, are removed from the Linux kernel, user processes can be put to use for real-time applications. This approach is the most convenient way to run iocCore, since the OSI libraries based on the POSIX threads can benefit from the improvement just by replacing the Linux kernel, as long as the POSIX threads use kernel level threads. Possible evolution of the Linux kernel is clearly isolated from that of iocCore at the POSIX interface.

The most promising way to achieve this seems to be to make the Linux kernel preemptive by converting SMP spin-locks to a measure for mutual exclusion between multiple processes on a single processor. This approach includes the changes to have the kernel honor specified priorities and to switch the process on a return from interrupt if it is possible and required. This improves the latency down to, *typically*, around one millisecond [4].

As some argue, however, the predictability of the latency still remains essentially at a soft real-time level. It is true that the requirement to the spin-lock is essentially the same regardless of whether it is for between processes on different CPUs, or for between processes on a single processor. However, it does not necessarily mean that the usage of spin-locks for SMP in the existing Linux kernel is optimized enough to ensure real-time responsiveness when it is converted. The average performance of SMP stays even if a few parts of the kernel code allow a process to run too long with holding a spin-lock. It, however, directly affects the longest latency if the spin-lock is used to make the Linux kernel preemptive. Checking up on the whole kernel codes to fix such problematic parts should take much effort, since the Linux kernel is too huge and complex to be fully analyzed. Furthermore, the check must be iterated every time any part of the Linux kernel is modified, or a new driver module is added to the kernel.

3.3 User processes under real-time kernel

The two approaches discussed in the previous subsections aim at either high predictability or high availability. To be available, the threads of iocCore must run in user-space. To be predictable, the Linux kernel must be excluded from the execution path of dispatching the threads. If a system allows threads in user-space to be dispatched by a real-time kernel, consistency of availability and predictability is possible.

This approach should be free from any unpredictable latency due to the non-preemptiveness of the Linux kernel. It must also be able to avoid the disabling of interrupts by the Linux kernel. It accepts, though, the unpredictability due to address space switching for the cost to use user-space. The threads must be able to issue real-time kernel system calls, as well as Linux system calls, when predictable responsiveness is not necessary. At present, candidates for this approach are very few. One is LXRT, an option of RTAI [6]. Another is L4-Linux, as described in the next section.

4 IOC-CORE ON L4-LINUX

4.1 What is L4-Linux?

L4-Linux was developed at Dresden Institute of Technology in corporation with IBM Watson Research Center [2]. It is a port of Linux kernel as a server task on top of a real-time micro kernel, named L4, or its successor, named Fiasco [7], as illustrated in Fig. 1.

L4 is a preemptive micro kernel, which provides its tasks with only three primitives, threads, address spaces, and Inter Process Communication (IPC). The entity of the Linux server, as well as its “processes”, is an L4 task. The Linux “processes” call the Linux server for a service through an IPC call of L4. Page faults caused by a process are also transformed into IPC with the Linux server, and then handled in the same way as standard Linux systems. Every Linux “process”, being a L4 task, can also issue L4 system calls.

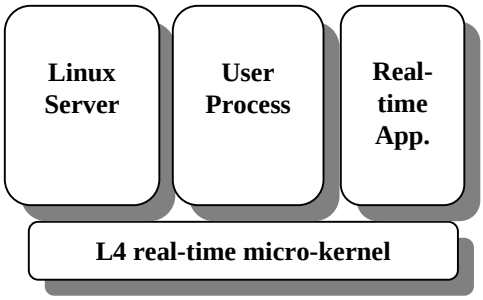


Figure 1: Architecture of L4-Linux

4.2 L4-Linux as a real-time platform

L4-Linux was designed in order to run a real-time application and non-real-time Linux applications on a single computer. Linux processes, hence, in themselves are not for real-time applications. However, the basic architecture of L4-Linux allows a process to be turned into a real-time process, basically, if it is given a higher priority than the Linux server. The process can preempt execution from the Linux server because both the "process" and the Linux server are just a task under L4's scheduling. The "process" can keep running, as long as it does not issue any Linux system calls, until it suspends itself by calling an L4 system call, or being preempted by another real-time process with higher priority.

A problem, unfortunately, was found in relation to virtual memory management. In L4-Linux, there are two different sets of page tables for the virtual memory space of a process. One is in the Linux server and the other is in the L4 kernel. The former is the one that the Linux server manipulates as does the standard Linux kernel. The latter is the real page tables that the MMU refers to. The two sets of page tables are forced to be equal upon page faults. This doubly layered memory management has two negative impacts on "real-time processes". First, a newly created thread through a Linux clone system call shares the page tables with its creator, not in the L4 kernel, but in the Linux server. The cloned thread gets its own page tables in the L4 kernel. This results in the need for address space switching upon "thread switching". The other one, which is more serious, is a coherency problem between the two sets of page tables. For example, a process can cause page faults even after a mlockall system call has been issued to make itself memory-resident, because mlockall returns with only the page tables in the Linux server updated, but leaving the real ones unchanged. Page faults to update the real page tables decimate the assumption that real-time processes run without depending on the Linux server.

With our patches to work around the coherency problem, interrupt response was measured with heavy disk I/O activity as a background. Including 3 times of "thread switching" into the real-time thread in the worst case, the worst latency was measured to be roughly 700 microseconds in 10^5 times of trials on a Celeron 300 MHz CPU [8]. It also includes the latency due to disabling of interrupt by the Linux server. L4-Linux can be configured so as not to allow the Linux server to disable interrupts. If this option is enabled and the number of context switching is reduced, the latency may be reduced down to around 100 microseconds.

4.3 iocCore on L4-Linux

In order to confirm the ability of the system to run iocCore, OSI-libraries for L4-Linux were implemented, though this is not complete. Each of their functions relies only on L4 system calls to preserve real-time responsiveness, as far as it is supposed to be called constantly from the iocCore threads. Some other functions that create threads or semaphores issue Linux system calls for resource allocation at a cost of losing the responsiveness temporarily, assuming that they are called only in the initialization sequences of iocCore. Channel Access- related threads, however, rely on Linux for TCP/IP socket services, and they work in much the same way as they do on the standard Linux kernel. Other than that, iocCore threads can run with lower predictable latency under L4 kernel control.

5 CONCLUSIONS

Though real-time tasks in the Linux kernel space can ensure the lowest and most predictable latency, running iocCore in the Linux kernel is an unfavorable solution. On the contrary, an improved Linux kernel gives the highest availability to iocCore, but its responsiveness essentially remains at a soft real-time level. If a platform allows threads in user-space to be dispatched by a real-time kernel, it can provide iocCore with not only an available basis, but also predictable responsiveness. L4-Linux is one of the possible candidates for this kind of approach.

REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] http://os.inf.tu-dresden.de/L4/LinuxOnL4/
[3] http://www-online.kek.jp/~nakayosi/
[4] http://www.mvista.com/realtime/
[5] http://www.rtlinux.org/
[6] http://www.aero.polimi.it/projects/rtai/
[7] http://os.inf.tu-dresden.de/fiasco/
[8] J. Odagiri et.al., "Porting EPICS to L4-Linux Based System, " PAC 2001, Chicago, USA, Jun. 2001

WEBT004
physics/0111054

IMPLEMENTATION OF JFPC (JAVA FOR PROCESS CONTROL) UNDER LINUX – A MECHANISM TO ACCESS INDUSTRIAL I/Os IN A JAVA/LINUX-ENVIRONMENT

H. Kleines, P. Wüstner, K. Settke, K. Zwoll,
Forschungszemtrum Jülich, D-52425 Jülich, Germany

Abstract

Using JAVA for process control requires the access to I/O-Hardware. This problem is addressed in several ongoing standardisation efforts for Real Time Java. A spin-off of these standardisation efforts is the Siemens product JFPC (Java for Process Control), which is based on the ideas of OPC (OLE for Process Control). It is available under the proprietary RT kernel RMOS and under Windows NT. Based on a contract with Siemens, ZEL - the central electronics facility of Forschungszentrum Juelich – has implemented JFPC under Linux. Additionally, several JFPC providers, e.g for accessing industrial components via PROFIBUS DP have been implemented.

1 INTRODUCTION

The Java programming language has become popular also in control applications for physics experiments, because of its simplified object model, its safety and its relative platform independence [1]. Forschungszentrum Jülich uses Java for small control applications, too. Because of Java`s security concepts and the abstraction from the underlying system, explicit access to I/O hardware is not supported in a pure Java environment. The product JFPC (Java for Process Control) which has been developed by Siemens for Windows NT and their proprietary real time kernel RMOS, offers an OPC-like access to process data. Forschungszentrum Jülich has done a port of major parts of JFPC to Linux in order to access industrial process periphery from an Linux-based Java application

2 HARDWARE ACCESS VIA JAVA

Explicit hardware access is not supported by the extensive Java APIs, but it is required in typical embedded and RT (real time) applications. So several hardware access APIs have been defined in the ongoing specification work for RT Java.

Besides the Java standardization in the "Java Community Process", which is controlled by SUN, there is standardization work going on in the J-Consortium, an independent organization of companies (HP, Microsoft, Siemens, Aonix, Omron,.....). The primary goal of the J-Consortium is to promote the development of standards for embedded and RT Java technologies. The J-Consortium was founded to get independent of SUNs intellectual property rights and to achieve a higher degree of openness and vendor-neutrality.

The outcome of the Java Community Process was a draft for the "RT Specification for Java" (RTSJ) [2], published by the "RT for Java Expert Group" as JSR-000001. RTSJ also defines I/O-access according to a simple flat approach: Typed data can be read/written by method calls of a RawPhysicalMemory object, by specifying the offset in the corresponding memory area.

The RT Java working group of the J-Consortium produced the standards draft covering the "RT Core" extensions to Java [3], which provide RT programming capabilities to the Java platform, as RTSJ does. But both documents are incompatible.

Because almost all applications in the domain of industrial automation require some kind of I/O access, but only some of them have RT requirements, the necessity for an I/O-Access API also on standard VMs is obvious. This issue is addressed by a further standards draft, the RTDA (RT Data Access) specification [4], produced by the RTDA working group of J-Consortium. The RTDA specification is incompatible to "RT Core " and RTSJ. Main focus of RTDA is the I/O-access mechanism, which is much more elaborate than in RTSJ and in RT Core. Functionally it corresponds to the approach of the Siemens product JFPC introduced in the following section.

3 OVERVIEW OF JFPC

JFPC is a Java class library (with OS-dependant modules in native code) providing an API for I/O data access, that is based on the ideas of OPC [5]. Data access to hardware registers is abstracted by proxy objects – the so called Items. Items are addressed by symbolic names, conforming to a URL-like notation, e.g.: */io/com202_1/slave_27/Bit/inbit_01*

This leads to a hierarchical structure of the namespace, reflecting the logical and physical architecture of the automation system, e.g. the existence of subordinate fieldbus systems

Access to JFPC is done via one central Server object, and the Items are subordinate objects. Items can be collected in groups, which contain also schedulable code (activation routine). Activation of groups can be done by timers, by interrupts or simply by monitoring the change of Items. All input Items of a group are read implicitly before calling the activation routine and all output Items are written implicitly after termination of the activation routine, thus implementing a PLC-like behaviour.

JFPC has a modular design, where the Server relies on a name service to locate Items and on providers to access the real hardware. There is a documented provider interface, enabling the extension of JFPC by new providers for new hardware components. Typically, providers consist of a Java part and a native code part, required to access the real hardware.

One specific provider - the so called "Remote Provider" - implements a proprietary application protocol above TCP, that allows client/server configurations of JFPC. So clients can access Items in the server via the Internet.

Each JFPC-System is configured dynamically during run time by reading a configuration file defining all target Items and the required providers with their associated parameters.

4 IMPLEMENTATION OP JFPC UNDER LINUX

4.1 General Approach

Forschungszentrum Jülich has received the complete source code of JFPC from Siemens. Main task of the port to Linux is the implementation of the so called RTS (run time system) provider. It is responsible for the handling of group activation. Thus it implements the timer services and the implicit pre-activation read of Itmes and post-activation write of Items, by calling optimized native C functions for reading and writing of items, which are registered at the RTS-Provider. Additionally specific providers for dedicated hardware have to be implemented. This includes Java code as well as native C code. Additionally Linux device drivers for all boards had to be implemented. We selected one digital output and and one digital input board as well as two PROFIBUS DP controllers.

PROFIBUS DP has been selected, because of its dominating market position. Thus a wide range of industrial equipment can be connected directly to this fieldbus and Forschungszentrum Jülich uses it as the common integral path to the world of industrial automation [6]. Hence evaluation of JFPC in the context of slow control applications requires the implementation of JFPC providers for PROFIBUS DP.

4.2 RTS-Provider

Only the native C code component of the RTS-Provider has to be changed in order to port it to a different operating system. The most important function to be implemented is *Java_DE_siemens_ad_jpc_server_RtsTimerEvent_waitTimerEvent0()*, which starts a timer and waits for its expiration in a thread context. This is simply mapped to the *nanosleep()* call, with additional use of *gettimeofday()* in order to reduce the jitter in the case of cyclic timers. Of course, all the timer control structures had to be modified in order to include process id and other additional bookkeeping data. An alternative approach using Linux interval timers with *setitimer()* has not been followed because of the unclear semantics of signals in a multithreaded environment.

4.3 Provider for the SMP16-EA216

SMP16-EA216 is digital output board by Siemens for the SMP (Siemens Microprocessor Periphery) bus. It has 32 optoisolated digital outputs organized in 4 independant ports. Signaling on the SMP bus is similar to the ISA bus, and the output ports are mapped by static configuration into the I/O space of the CPU. A Linux device driver has been implemented, which offers dedicated *ioctl()* calls, that do 8bit and 16bit write operations on these ports. Outputs are cached internally by the EA216 and can be read back. An additional bitmask parameter allows output of dedicated bits. The I/O base address of the board is an installation parameter of the device driver, and the device special name contains consecutive board numbers generated during installation (ea216_1 for board 1, etc). Implementation of the JFPC provider was straight forward, just by mapping the *writeItem* method to the appropriate *ioctl()* call. The URL-like name of an Item contains the special device file name as well as the byte or bit address in the range 0.0 to 3.7 (e.g. ..../ea216_1/Bit2.1 or .. /ea261_2/Byte0).

4.4 Provider for the SMP16-EA217

SMP16-EA217 is digital input board by Siemens for the SMP (Siemens Microprocessor Periphery) bus. It

has 32 optoisolated digital inputs organized in 4 independant ports. The input ports are mapped by static configuration into the I/O space of the CPU. A Linux device driver has been implemented, which offers dedicated *ioctl()* calls, that do 8bit and 16bit read operations on these ports. Implementation of the JFPC provider was analogous to the EA216, just by mapping the *readItem* method to the appropriate *ioctl()* call.

4.5 Providers for the CPCI-FZJ-DP

CPCI-FZJ-DP is PROFIBUS DP controller in the CompactPCI formfactor developed by Forschungszentrum Jülich [6]. It supports only the cyclic services of PROFIBUS DP. PROFIBUS DP provides uses a simple programming model, because process data of slaves is mapped to a DPRAM via cyclic transfers on the PROFIBUS. An application on the master just reads and writes this DPRAM, asynchronously to the operation on the bus. The position of the process image of each slave in the DPRAM is determined statically by bus configuration. So basically a generic PROFIBUS DP provider has nothing else to do than to read and to write data in the DPRAM, by specifying the following parameters: PROFIBUS address of the slave, the length of the data, the relative start address of the data in the process image of the individual slave and a flag indicating input or output. An additional issue for JFPC is the interpretation of the data, because Java is strictly typed, and the representation of the above mentioned parameters in the JFPC configuration file

A possible approach for the implementation of providers would follow a hierarchical structure, with one provider for PROFIBUS DP, one provider for each type of slave, and additional providers for submodules in the case of modular slaves. We didn't choose this approach, because different slaves, even modular slaves, don't differ in their access mechanism, but only in the interpretation of data. So we chose a flat approach with different providers according to the interpretation of data and implemented the following four providers:

- Bit-Provider: generic provider for digital I/Os
- Word-Provider: generic provider for analog I/Os
- Provider for the Siemens stepper motor controller 1STEP
- Provider for Heidenhain encoders

All providers share a common native code part for accessing process data in the DPRAM. The Java code is device-specific, e.g, does scaling and format conversion in the case of the Word-Provider.

4.6 Provider for the CPCI-COM202

CPCI-COM202 is a PROFIBUS DP controller for CompactPCI, developed by Siemens. It supports not only the cyclic Services of DP but also DPV1 and DPV2 (isochronous PROFIBUS). The existing Windows NT driver of Siemens basically gives access to the DPRAM of the board. We implemented a device driver for Linux, that maps this functionality to appropriate *ioctl()* calls. Above the driver we implemented a library supporting the cyclic services of PROFIBUS DP, which was almost identical to the corresponding library for CPCI-FZJ-DP. Thus the corresponding Provider could be reused.

FUTURE WORK

The JFPC port to Linux is still not completely stable. As soon as this problem is solved, performance analysis of the final system will be done. Also the jitter in group activations controlled by a interval timer will be analyzed. For the examination of RT behaviour, the implementation of JFPC under LynxOS, using HPs ChaiVM, is planned.

REFERENCES

[1] O. Barana, A. Luchetta, G. Manuchi, C. Talericio, "A gneral purpose Java tool for action dispatching an dsupervision in nuclear fusion experiments", *Proceedings 12th IEEE Real Time Congress on nuclear and plasma science*, Valencia, Spain, June 2001

[2] G. Bollella, et al. , "The Real Time Specification for Java", Addison-Wesley, 2000

[3] "Real Time Core Extensions for the Java platform", Draft International J-Consortium Specification, http://www.j-consortium.org

[4] "Real Time Data Access", Draft International J-Consortium Specification, http://www.j-consortium.org

[5] P. Brisch, G. Hinsken, K.-H. Krause, "Echtzeitprogrammierung in Java", Publicis MCD Verlag, München, 2000

[6] H. Kleines, K. Zwoll, M.Drochner, J. Sarkadi, "Integration of Industrial Automation Equipment in Experiment Control Systems via PROFIBUS – Developments and Experiences at Forschungszentrum Jülich", *Proceedings 11th IEEE Real Time Conference*, Santa Fe, USA, June 1999

Wednesday, 28 November 2001

WEC - Closed Loop and Feedback Systems

WECI001
physics/0112043

ORBIT FEEDBACK USING X-RAY BEAM POSITION MONITORING AT THE ADVANCED PHOTON SOURCE

Glenn Decker, Om Singh, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

The Advanced Photon Source (APS) was commissioned in 1995 as a third-generation x-ray user facility. At that time orbit control was performed exclusively with broadband rf beam position monitors (BPMs). Since then, emphasis has been placed on incorporating x-ray beam position monitors into the orbit control algorithms. This has resulted in an order of magnitude improvement in long-term beam stability vertically, using x-ray BPMs (X-BPMs) on bending magnet beamlines. Additional processing will allow similar improvements horizontally, once systematic effects associated with variable insertion device (ID) x-ray beams are properly compensated. Progress to date and upgrade plans will be presented, with an emphasis on the details of the required digital signal processing.

1 INTRODUCTION

The APS is an x-ray synchrotron radiation user facility based on a 7-GeV electron storage ring. Highly collimated synchrotron radiation x-rays are emitted from the electron beam as it traverses bending magnet and IDs. A primary performance measure is the allowable amount of particle beam motion (and thus x-ray beam motion).

Table 1: APS Beam Stability Specification

	Horizontal (rms)	**Vertical (rms)**
Initial Design Values	17 microns	4.5 microns
Low Emittance / 1% Coupling Values	12.6 microns / 900 nanorad.	0.59 microns / 120 nanorad.

Shown in Table 1 are beam stability specifications for the APS. As originally stated in the APS initial design circa 1991, they are incomplete insofar as no explicit angular pointing stability specification is given. Further, both the initial values and the present low-emittance values are incomplete since no frequency band or time scale is indicated. The low-emittance mode of operation has smaller horizontal and vertical particle beam sizes in comparison to the initial design. The values in Table 1 are generally understood to apply to frequencies up to 30 Hz and down as low as reasonably achievable. Typically, APS users are most interested in beam position drifts over minutes to hours, fill-to-fill reproducibility (24 hours), and run-to-run reproducibility (months). To get an idea of scale, 100 nanoradians is the angle subtended by an object 1 mm high, observed from a distance of 10 kilometers.

2 SYSTEM DESCRIPTION

Excellent descriptions of the APS orbit correction systems are available in the literature [1,2,3,4]. Key elements include 360 broadband rf BPMs, 48 narrowband rf BPMs, 38 bending magnet x-ray BPMs and 48 ID x-ray BPMs. Data from these monitors is used to control up to 317 combined-function horizontal/ vertical corrector magnets. Of the 317 correctors, 38 are mounted on thin-walled vacuum spool pieces allowing, in principle, correction bandwidth up to 200 Hz. The balance of the correctors is otherwise identical, but mounted around the standard thick-walled aluminum vacuum chamber. Eddy currents in the aluminum limit their bandwidth to less than 10 Hz.

X-ray BPMs are constructed of metallized, chemical-vapor-deposited (CVD) diamond wafers placed edge on to the x-ray beam [5]. Photocurrents are measured for blades placed above and below the beam for bending magnet beams and arrayed symmetrically in sets of four for collimated ID beams (Figure 1).

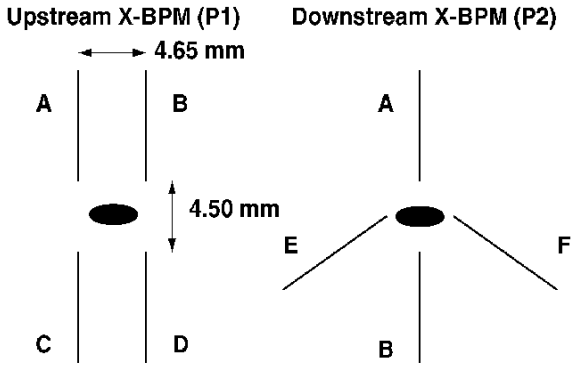


Figure 1. APS insertion device X-BPM geometry.

A pair of x-ray BPMs is mounted just inside the accelerator enclosure in each beamline "front end," connecting the accelerator vacuum to the beamline located outside of the enclosure, on the experiment hall floor. Each x-ray BPM is mounted on an X-Y translation stage with resolution better than 1 micron and better than 10-micron reproducibility. Software that uses these translation stages for convenient device calibration is in

place. Two portable translation stage IOC crates have been developed for periodic calibrations and maintenance. Data acquisition for the x-ray BPM photocurrent signals has recently been completed for all installed units [1].

Data acquisition for the broadband rf BPMs takes place in customized VXI modules with simple boxcar averagers generating EPICS process variable (PV) values. A separate fiber optic communications link provides data to the real-time feedback system residing in a nearby VME input/output controller (IOC) crate (21 total). For the narrowband rf and x-ray BPMs, a dedicated 32-channel, 16-bit analog-to-digital converter (ADC) resides in the local real-time feedback IOC crate. Also included in the real-time feedback crates are two Pentek 4284 digital signal processing (DSP) modules, one Pentek 4283 DSP, a VMIC 5588 reflective memory module to provide real-time data transfer between crates, a 12-bit ADC for tracking auxiliary BPM status information, and an MVME 167 IOC module using a Motorola 68040 central processing unit (CPU). Also included is a power supply interface module for the real-time system.

Two of the DSPs in each real-time feedback crate perform the matrix multiplication operations connecting up to 160 broadband rf BPM readbacks to two local corrector power supplies, one horizontal and one vertical, at a 1.5-kHz rate. The third DSP is dedicated to processing data from the 16-bit ADC. Functions include digital filtering of position signals from the narrow-band rf BPMs, and computation and filtering of position signals from raw x-ray BPM blade photocurrent signals. Data is transferred to the vertical real-time feedback DSP and broadcast around the facility through the reflective memory network. This same data is deposited in dual-port RAM for transfer to the EPICS IOC module. Further filtering is performed in the IOC to provide pretty well de-aliased, approximately 1-Hz bandwidth signals as EPICS process variables.

A total of 21 real-time feedback IOC crates are located around the APS storage ring circumference, all connected by the reflective memory network (Figure 2). Each of 20 “slave” crates provides coverage for two APS sectors. The twenty-first crate has additional processing and provides supervisory and diagnostic support, including a forty-channel virtual oscilloscope, which can monitor any of the signals accessible on the reflective memory. This crate also provides process variables indicating total rms beam motion averaged over a maximum of 80 locations. These values are logged every 60 seconds and can be compared to the values in Table 1 to verify compliance with beam stability requirements.

A twenty-second crate has recently been added to the reflective memory network, known as the “data pool” IOC. Its purpose is ultimately to replace the “DC” orbit correction software that presently runs at the workstation level with a 2.5-second update rate. The real-time feedback system is limited in flexibility owing to the 160 by 38 feedback matrix size, and therefore explicitly avoids correcting DC orbit motion by using a 0.1-Hz high-pass filter. The workstation-based software in principle corrects everything up to this 0.1-Hz frequency, but in practice, a “dead band” at 0.1 Hz is seen. This is very unfortunate, since transient effects caused, for example, by ID gap closures, can persist for tens of seconds and have tens of microns in amplitude. The transients have been reduced somewhat by sending information from the DC algorithm to the real-time system [6].

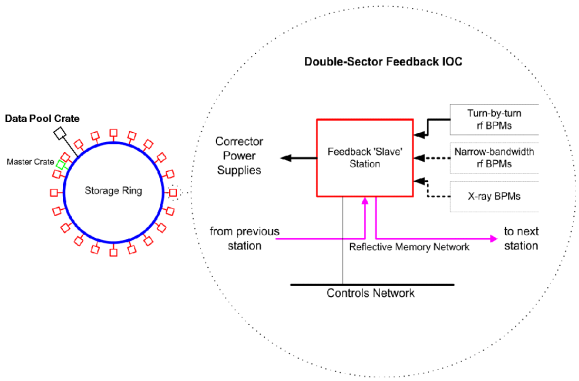


Figure 2. Layout of real-time feedback systems.

To remove the source of the transient, an ID gap feedforward solution for this has been tested using the workstation-based software. Unfortunately, because the convergence time is so slow, the generation of the feedforward lookup tables is extremely time consuming in addition to being subject to longer time scale systematic errors. By pooling all BPM data in a single IOC and running the present workstation-based software on that IOC, we hope to be able to speed up the DC orbit correction algorithm from 0.4 Hz to 50 or even 100 Hz. Feedforward algorithms need not run on this IOC, however the elimination of the 0.1-Hz dead band is essential to the determination of lookup tables for the dozens of IDs installed in the APS accelerator enclosure.

3 PERFORMANCE

As with any system of this complexity, performance is controlled to a large extent by the system configuration. A Tcl application is provided for this purpose, written by the APS operations analysis group [7]. This tool allows for the selection of arbitrary BPMs, correctors, and BPM weights, and allows for different accelerator configurations (e.g., low emittance vs. standard quadrupole and sextupole magnet settings). The tool uses a singular value decomposition algorithm to invert the BPM/corrector response matrix (if possible), giving the option of limiting the number of eigenvalues. Once an

inverse response matrix (IRM) is available, a separate workstation-based Tcl application called controllaw uses the IRM to perform the actual orbit correction. Options like update rate, gain, despiking, etc. are available with this application at run time.

Typical "flavors" of orbit correction can roughly be categorized as being either global or local. Local correction typically involves a high density of steering correctors straddling an x-ray source point, allowing independent control of displacement and angle, without interfering with the steering of nearby beamlines. While allowing a high degree of flexibility, local steering has the disadvantage of requiring relatively large corrector changes (Amps) for each unit of steering (microns or microradians). During normal operation, local steering is generally only performed on demand from a particular beamline.

For standard operation, once local beamline steering is complete, a "global" algorithm using many more BPMs than correctors is used. This has the advantage of a large corrector "lever arm," in addition to being relatively insensitive to systematic errors affecting any individual BPM. In effect, a smooth curve is arrived at providing the best fit to the available BPM data. At the APS generally two correctors are used in each of the forty sectors. For the horizontal plane, only rf BPMs are presently in use (ID x-ray BPMs await the availability of the data pool IOC and feedforward). As many BPMs as possible are used. In the vertical plane, the broadband rf BPMs exhibit a hypersensitivity to bunch fill pattern [8] and are not used during top-up operation (injection every two minutes with shutters open). Use of the narrow-band rf BPMs and bending magnet x-ray BPMs together with two correctors per sector performs very well for the vertical plane.

Studies were performed using the bending magnet BPMs in a "local" configuration during user beam operation. For the bending magnet beamline in question, only one of the two available x-ray BPMs ws included in the algorithm, with an increased BPM weight in the matrix. The second unit, not included, was used to monitor the performance independently. In this case, stability was limited by the performance of the broadband rf BPMs straddling the source point, but was improved by an order of magnitude in comparison to operation without x-ray BPMs.

Using the "global" vertical corrector configuration (with as few broadband rf BPMs as possible) has allowed what is most likely the first true submicron beam stabilization with duration of more than five days. Shown in Figure 3 are data collected over a six-day period from two x-ray BPMs installed in an APS bending magnet beamline front end. Shown on the horizontal axis is the average of the two monitor readbacks, while the vertical axis shows the readback difference divided by longitudinal separation between the two units. Thus the vertical axis gives the x-ray beam's pointing angle in the vertical plane. Figure 3 is therefore a phase-space representation of beam stability in the frequency band extending from a fraction of a Hz down all the way to a five-day period. Admittedly, the horizontal axis is artificially small; after all, in this case both units were included in the orbit correction algorithm so it is not surprising that a small spread, less than half a micron rms, is observed. Even if one unit were to have systematic errors, the algorithm would simply track them in such a way as to minimize the average of the two units' readbacks. What is compelling, however, is that the rms vertical pointing angle over this five-day period (each day is represented by a different color dot) is 180 nanoradians. Thus, at a distance of 50 meters from the source, down along the x-ray beamline, the beam spot was stable to better than 10 microns rms over this five-day period. This is pretty darn good; I venture to say it is the best in the world.

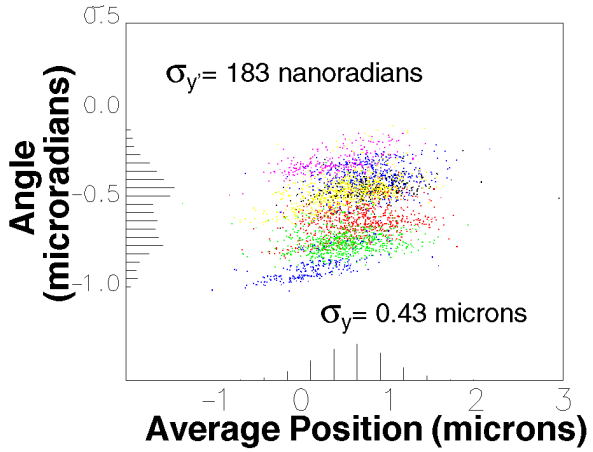


Figure 3. Phase-space representation of APS vertical beam position stability over a five-day period. Different colors correspond to separate days.

4 UPGRADE PLANS

While performance to date has been excellent in some areas, much remains to be done. Elimination of the 0.1-Hz system dead band described above will take the highest priority. Implementation of the 100-Hz data pool IOC for this purpose will allow further essential upgrades.

A major effort has been ongoing for the past two years to realign accelerator girders in order to reduce systematic errors in the ID x-ray BPMs associated with stray synchrotron radiation impinging on the photosensitive blades [9]. While this effort has done much to reduce background signals by a factor of between three and ten, it is clear that significant ID gap-dependent systematic errors remain, on the order of tens of microns. Research toward feedforward algorithms is

promising, whereby a process monitors ID gaps and writes corrections to local steering correctors as the gap is changed in such a way as to reduce any residual steering error caused by the ID. In addition, this same process, or one like it, could make position offset corrections to the ID x-ray BPMs for that beamline, providing position readbacks that are relatively gap-independent. For this purpose, insertion device gap information will be placed on the reflective memory network, allowing real-time access by e.g., the new data pool IOC.

The present limitation on feedforward is the fact that in order to generate look up tables, one must carefully scan each of the over 22 IDs in series, with orbit correction running. Early tests show that, using existing algorithms, fully one hour of machine studies time is required for each ID, with only modest improvement, a factor of approximately three. By using the data pool, collection of feedforward data should take no longer than the time to configure the system and close the gap – one minute vs. one hour. Further, systematic drifts that creep in during the very slow measurement will be much reduced.

The real-time (1.5-kHz update rate) feedback system was completed by 1998 in its present incarnation [4]. It uses only the broadband rf BPMs to reduce beam noise in the band from 0.1 to 30 Hz. While new narrow-band and x-ray BPMs have become available since that time, they have been used exclusively for DC orbit control, running on a workstation in the main control room. This circumstance has mainly arisen from a limit in the area of applications programming. While narrow-band and x-ray BPM data are available to the 1.5-kHz system, the emphasis has been put on DC orbit control, which has been a difficult and high-priority challenge. Once the insertion device x-ray BPMs have been comfortably integrated into DC orbit control, including the new data pool IOC, attention will turn back to the 1.5-kHz real-time system. At present, orbit stability in the 0.1- to 30-Hz band is sitting around 1.3 microns rms vertically and 2.0 microns horizontally. As can be seen from Table 1, additional work in the real-time system is required if true submicron stability is to be realized. A long-range goal is to integrate all available BPM data (rf and x-ray) into the fast feedback algorithm.

5 CONCLUSIONS

Orbit correction technology at the APS is at a very advanced stage with the advent of new x-ray BPM data acquisition and processing. Long-term vertical pointing stability has been improved to better than 100 nanoradians rms for times scales from 10 seconds to 24 hours. Inclusion of all rf and x-ray BPM data on the reflective memory network will allow two orders of magnitude increase in the DC feedback update rate. This, in turn, will make possible feedforward algorithms to further eliminate beam motion transients induced by insertion device gap changes.

6 ACKNOWLEDGEMENTS

The real-time feedback system was conceptualized in the early 1990s by Youngjoo Chung. John Carwardine and Frank Lenkzsus corrected all the early conceptual errors and turned it into a working system [4]. Deming Shu performed the mechanical design of the x-ray BPMs in the tunnel.

This work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

7 REFERENCES

[1] F. Lenkszus, “New X-Ray BPM Data Acquisition System for the APS,” PAC’01, Chicago, June 2001, to be published.

[2] O. Singh, G. Decker, “Operational Experience with X-ray Beam Position Monitors at the Advanced Photon Source,” PAC’01, Chicago, June 2001, to be published.

[3] G. Decker, “Strategy for Achieving True Submicron Orbit Stabilization at the Advance Photon Source,” PAC’01, Chicago, June 2001, to be published.

[4] J.A. Carwardine, F.R. Lenkszus, “Real-Time Orbit Feedback at the APS,” AIP Conference Proceedings No. 451, p. 125, 1998.

[5] D. Shu et al., Nucl. Instrum. Methods A 319, 56 (1992).

[6] C. Schwartz, L. Emery, “Compensating the Frequency Deadband of the APS Real-Time and DC Transverse Orbit Correction Systems,” PAC’01, Chicago, June 2001, to be published.

[7] M. Borland, “Applications toolkit for accelerator control and analysis,” PAC’97, Vancouver, BC, Canada, pp. 2487-2489, 1998.

[8] Y. Kang, G. Decker, J. Song, “Damping Spurious Harmonic Resonances in the APS Storage Ring Beam Chamber,” PAC’99, New York, pp. 3092-3094, 1999.

[9] G. Decker, O. Singh, “A method for reducing X-ray background signals from insertion device X-ray beam position monitors,” Phys. Rev. ST Accel. Beams 2, 112801 (1999).

WECT002
physics/0111104

THE WAVEFRONT CONTROL SYSTEM FOR THE NATIONAL IGNITION FACILITY

Lewis Van Atta, Mark Perez, Richard Zacharias, and William Rivera
Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Abstract

The National Ignition Facility (NIF) requires that pulses from each of the 192 laser beams be positioned on target with an accuracy of 50 μm rms. Beam quality must be sufficient to focus a total of 1.8 MJ of 0.351-μm light into a 600-μm-diameter volume. An optimally flat beam wavefront can achieve this pointing and focusing accuracy. The control system corrects wavefront aberrations by performing closed-loop compensation during laser alignment to correct for gas density variations. Static compensation of flashlamp-induced thermal distortion is established just prior to the laser shot. The control system compensates each laser beam at 10 Hz by measuring the wavefront with a 77-lenslet Hartmann sensor and applying corrections with a 39-actuator deformable mirror. The distributed architecture utilizes SPARC AXi computers running Solaris to perform real-time image processing of sensor data and PowerPC-based computers running VxWorks to compute mirror commands. A single pair of SPARC and PowerPC processors accomplishes wavefront control for a group of eight beams. The software design uses proven adaptive optic control algorithms that are implemented in a multi-tasking environment to economically control the beam wavefronts in parallel. Prototype tests have achieved a closed-loop residual error of 0.03 waves rms.

1 INTRODUCTION

A primary requirement for NIF is that each beam shall deliver its design energy into a 600-μm hohlraum ICF target. The total design energy for 192 beams is 1.8 megajoules. A goal for the system is that 50% of the design energy should be contained within a 100-μm focal spot at the target plane. This is about twice the diffraction-limited 80% energy spot size.

To meet the spot size requirement and goal, NIF subsystems are designed to limit wavefront aberrations. Optics have stringent specifications for rms surface gradient, power spectral density and surface roughness. Stringent specifications are also maintained for optical component mounting. NIF systems are designed to mitigate the effects of temperature and humidity variations and vibrations. An active alignment system is employed to point the beams accurately into the target. Even with these efforts to control wavefront aberrations, the spot size requirement and goal could not be met without a wavefront control system.

2 WAVEFRONT SYSTEM DESIGN

A block diagram of the NIF main laser optical system is shown in Figure 1, with the NIF Wavefront Control System components highlighted. The path of the NIF beam is first summarized and then the wavefront control functions applied to the NIF beam (or the wavefront control beam surrogate) are described.

The NIF preamplifier beam (1.053-μm wavelength, called 1ω) enters the main laser chain near the focus of the transport spatial filters (TSF), directed away from the target. The beam exits the filter collimated and passes through the booster amplifier heading toward the laser main amplifier cavity. A Pockels cell is set to allow the beam to enter the cavity, where it makes four passes through the main amplifier before the Pockels cell is switched to allow the beam to exit. The beam then exits the cavity, passes through the booster amplifier and the TSF, and heads towards the target chamber. The beam is frequency-converted to 351-nm wavelength, (called 3ω) at the target chamber.

Wavefront control functions are implemented as follows. Between shots, a continuous wave probe beam is co-aligned with the NIF beam prior to injection into the main laser. The probe beam follows the NIF beam path. A 39-actuator large-aperture deformable mirror (DM) operates at the far end of the laser cavity where the beam bounces twice. This two-bounce configuration doubles the effective stroke of the DM. At the TSF output, a tilted sampling surface reflects a small fraction of the probe beam towards a pick-off mirror near the TSF focus that sends the sampled beam through relays to the output sensor. Within the output sensor, a 77-lenslet Hartmann sensor measures the wavefront. The Hartmann sensor's video output is read by a framegrabber in the Wavefront Control System's Hartmann front-end processor (FEP). The Hartmann processor FEP transmits measured wavefront data to the mirror control FEP, which calculates the surface displacements to be applied to the DM to correct the wavefront aberrations in the beam.

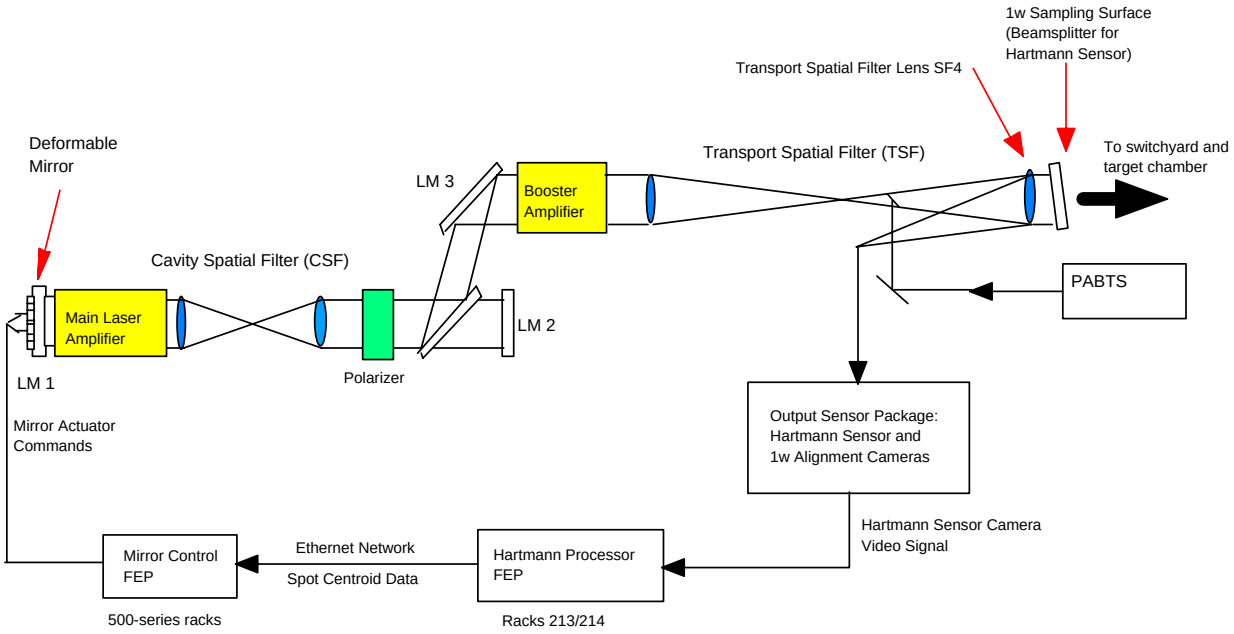


Figure 1: Block diagram of the NIF main laser optical system.

2.1 Wavefront System Requirements

There are several functions that the wavefront system should perform. Required functions are to: (1) control the 1ω output wavefront (including pointing) of each beam in the time immediately preceding a laser system shot, (2) apply compensation for previously measured pump-induced wavefront distortion, (3) measure beam output wavefront during a laser system shot, and (4) control the output wavefront (excluding pointing) during routine system operations between shots.

Other factors that influence the Wavefront control system requirements are expected system cost, and experience with Beamlet, the single-beam NIF prototype laser. The wavefront control system requirements are shown in Table 1. The initial design has met most requirements with the exception of closed-loop bandwidth. The bandwidth is processor-limited, and it is expected that by the time NIF is implemented, faster processors will be available to help meet this requirement without major changes to the software or design architecture.

3 SYSTEM CALIBRATION AND OPERATION

3.1 Control System Calibration

The wavefront measuring system is calibrated by inserting a wavefront reference fiber at the focal point of the TSF. Since the fiber light source is smaller than the TSF focal spot, the spot pattern at the Hartmann sensor when viewing the fiber reference beam is the same as the pattern the sensor would see when viewing the probe beam, if the beam and all upstream system components had diffraction-limited performance. The aberrations (imperfections in the separations of the lenslet array focal points) that are seen with the reference source inserted are due to the aberrations in the measuring system (sampling surface, relay optics, output sensor optics, and the sensor itself). By designing the control system to use the sensor focal spot image of the wavefront reference as the target wavefront to which the system wavefront is controlled, the system is being forced to generate, as closely as it is able, a perfect focal spot in the TSF. This also implies that all aberrations in all the optics beyond the TSF focus, including the TSF output lens, are uncorrected by the baseline wavefront control system.

Table 1: NIF Wavefront Control System Requirements

Requirement	**Value**
Maximum residual low frequency spatial angle	20 µradians
Maximum open-loop time before a shot	1 second
Minimum closed-loop bandwidth	1 Hz (0.5 Hz demonstrated)
Compensation range for simple curvature (double-pass reflected wavefront)	15 waves at 1ω
Order of aberrations corrected	•4th order
Measurement accuracy at 1ω	0.1 waves
Lenslet spacing	•1/2 of demagnified actuator spacing

Next, the wavefront reference source is replaced by the probe beam, and the wavefront control system is calibrated by an online procedure. Each of the 39 actuators is individually poked and pulled relative to the best-flat starting point. The offset for each Hartmann spot is thus related to the displacement of each actuator. From this information, a gain matrix relating actuator movement to Hartmann sensor focal spot movement is derived.

3.2 Nominal Closed-Loop Operation

Once the calibrations are complete, the loop is closed wherein the measured Hartmann offsets from the reference positions are multiplied by the gain matrix, yielding the actuator offsets to control the mirror to flat (with appropriate loop gain for stability). This is the configuration used during pre-shot alignment. After alignment is complete and the shot sequence has begun, an additional Hartmann offset file is subtracted from the wavefront sensor data prior to being applied to the gain matrix. These additional offsets represent the uncorrected prompt pump-induced wavefront aberrations measured from a previous shot. By subtracting out these offsets, we are setting the wavefront to the conjugate of the expected prompt aberration of the upcoming shot. Thus, at shot time, the wavefront is flat.

The control system operates with a closed-loop bandwidth of up to 0.5 Hz. To achieve this, the wavefront sensor is read at a 10 Hz rate (with a goal of 30 Hz). The sensor is read in standard RS-170 video, which is captured by a DataCell Snapper 24 framegrabber. The digitized image is fed into a SPARCengine Axi computer that calculates centroids for all 77 lenslet spots. This information is sent via a dedicated ethernet line to the Motorola MVME 2306 controller that calculates the offsets of the spots from their reference positions. These offset data are input to the mirror control law, which calculates the required DM actuator displacements. Each Hartmann processor serves eight beams, and each mirror control processor serves four DMs. Each mirror control chassis contains two mirror control processors.

4 SOFTWARE DESIGN

The NIF wavefront control system software was designed using an object-oriented approach. The NIF facility is expected to be in operation for 30 years, and it is reasonable to expect to make software or hardware changes over that time. By using modular hardware and software architecture, the system maintainability is improved significantly.

The object structure of the wavefront control system essentially conforms to the generalized hardware interfaces (e.g. digital to analog conversion, network communication, image frame grabbing) and major sensing and control activities (spot tracking, display building, and control law processing) that take place. Abstract interface layers are used to minimize the effect on software of prospective hardware changes. This approach allows us to create as many beams per processor as the central processing unit hardware and memory will allow.

The real-time tasking design of the wavefront control software is basically interrupt-driven. The interrupt source is the video sync signal contained in the Hartmann sensor composite video captured by the framegrabber. Completion of this and each successive task (spot tracking, display building, spot data transmission) sends a signal to start the next successive task. Similarly, the mirror control processor waits for spot centroid data to arrive and processes it in a sensor task (which reads the data and subtracts the reference spot positions to produce the wavefront error), then the mirror task is signaled to use the wavefront error to compute the mirror control command.

5 RESULTS

In tests with the LLNL wavefront interferometry lab, the wavefront control system has achieved 30 Hz operation for one beam and a residual error of 0.03 waves rms when controlling the wavefront to a reference measured from an optical reference flat.

REFERENCES

[1] R. Zacharias et al., “The National Ignition Facility Wavefront Control System,” Third International Conference on Solid State Lasers for Application to Inertial Confinement Fusion, Monterey, CA, June 1998.

This work performed under the auspices of the U.S. DOE by LLNL under contract No. W-7405-Eng-48.

WECT003
physics/0111047

READOUT AND CONTROL OF A POWER-RECYCLED INTERFEROMETRIC GRAVITATIONAL WAVE ANTENNA *

Daniel Sigg and Haisheng Rong, LIGO Hanford Observatory,
P.O. Box 1970 S9-02, Richland, WA 99352
Peter Fritschel, Michael Zucker, Department of Physics and Center for Space Research,
Massachusetts Institute of Technology, Cambridge, MA 02139
Rolf Bork, Nergis Mavalvala, Dale Ouimette, LIGO Project,
California Institute of Technology, Pasadena, CA 91125
Gabriela González, Department of Physics and Astronomy, Louisiana State University,
Baton Rouge,LA 70803

Abstract

Interferometric gravitational wave antennas are based on Michelson interferometers whose sensitivity to small differential length changes has been enhanced by adding multiple coupled optical resonators. The use of optical cavities is essential for reaching the required sensitivity, but sets challenges for the control system which must maintain the cavities near resonance. The goal for the strain sensitivity of the Laser Interferometer Gravitational-wave Observatory (LIGO) is 10^{-21} rms, integrated over a 100 Hz bandwidth centered at 150 Hz. We present the major design features of the LIGO length and frequency sensing and control system which will hold the differential length to within 5×10^{-14} m of the operating point. We also highlight the restrictions imposed by couplings of noise into the gravitational wave readout signal and the required immunity against them.

1 INTRODUCTION

The interferometric gravitational wave detectors currently under construction by LIGO [1], VIRGO [2], GEO [3] and TAMA [4] are expected to reach strain sensitivity levels of $\sim 10^{-22}/\sqrt{\text{Hz}}$ at 150 Hz over baselines of several hundred meters up to several kilometers [5]. To achieve this sensitivity all of these interferometers implement a Michelson laser interferometer enhanced by multiple coupled optical resonators [6, 7].

LIGO implements a power-recycled Michelson interferometer with Fabry-Perot arm cavities (see Fig. 1). Using optical cavities is essential in reaching the ultimate sensitivity goal but it requires an active electronic feedback system to keep them "on resonance". The control system must keep the round-trip length of a cavity near an integer multiple of the laser wavelength so that light newly introduced into the cavity interferes constructively with light from previous round-trips. Under these conditions the light inside the cavity builds up and the cavity is said to be on resonance [8]. Attaining high power buildup in the arm cavities also requires that minimal light is allowed to leave the system through the antisymmetric port, so that all the light is sent back in the direction of the laser where it is reflected back into the system by the power recycling mirror. Hence, an additional feedback loop is needed to control the Michelson phase so that the antisymmetric port is set on a dark fringe.

* Work supported by National Science Foundation cooperative agreement PHY-9210038

2 ENVIRONMENTAL INFLUENCES

It is important to distinguish low (< 50 Hz) and high frequency behaviour of the instrument. The low frequency region is typically dominated by environmental influences many orders of magnitude larger than the designed sensitivity and in many cases also many orders of magnitude larger than what can be tolerated for stable operations. It is the high frequency regime which yields good sensitivity and which is used for detecting gravitational waves. To suppress low frequency disturbances many active feedback control systems are needed to compensate 4 longitudinal [9] and 14 angular [10] degrees-of-freedom in the main interferometer alone. Additional feedback compensation networks are needed to locally damp the suspended mirrors (13×4 dofs), to control the mode cleaner (5 dofs) and to control the laser (2 dofs).

For example, seismic motion of the ground [13] is many orders of magnitude larger than the required gravitational wave sensitivity. In LIGO a multi-stage passive seismic isolation stack [11] together with a single-stage pendulum suspension system [12] is used to isolate the optical components from ground vibrations. This system works well for frequencies above ~ 10 Hz, but gives no suppression at frequencies of a Hz and below.

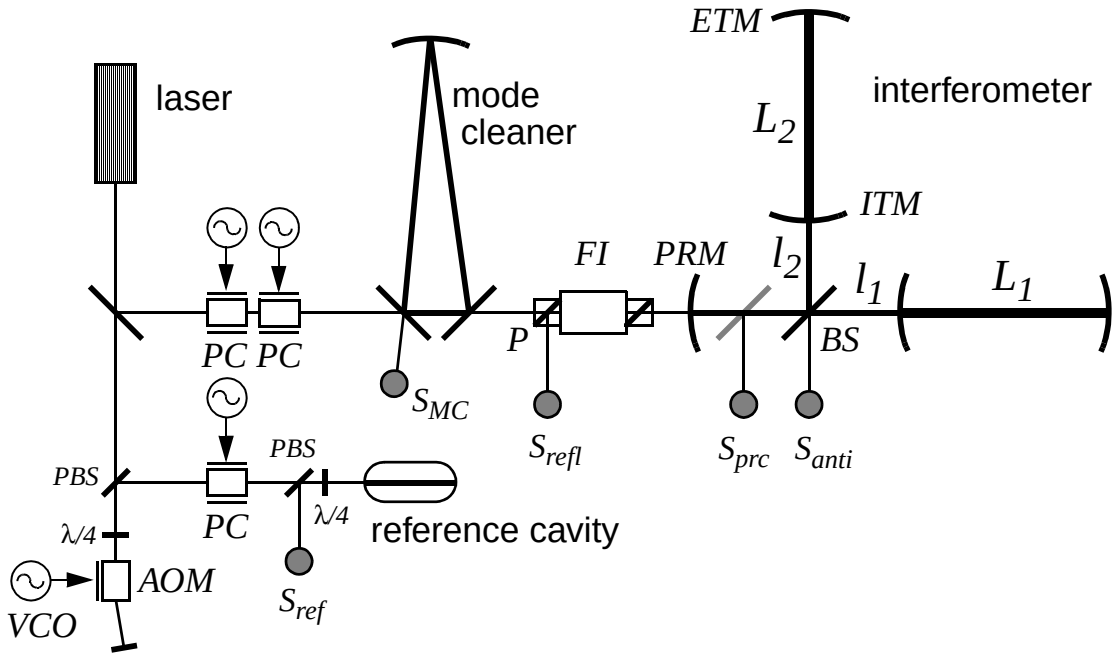


Figure 1: Schematic view of the optical path in LIGO. The light of a frequency stabilized Nd:YAG laser is passed through a triangular mode cleaner cavity before it is launched into a Michelson interferometer. To stabilize the laser frequency a small fraction of the light is sampled, doubly passed through an acousto-optic modulator (AOM) which serves as a frequency shifter, passed through a Pockels cell and sent to a reference cavity. Using a polarizing beamsplitter (PBS) and quarter-wave plate (λ/4) the light reflected from the reference cavity is measured by a photodetector to obtain the error signal, $S_{\rm ref}$, which in turn is used to adjust the laser frequency. The main laser light is passed through a pre-modecleaner (not shown) and two Pockels cells which impose the phase-modulated rf sidebands used to lock the mode cleaner and the Michelson interferometer. The mode cleaner locking signal, $S_{\rm MC}$, is measured by a photodetector in reflection of the mode cleaner cavity. The light which passes through the mode cleaner is sent through a Faraday isolator (FI) which also serves the purpose—together with a polarizer (P)—to separate out the reflected light signal, $S_{\rm refl}$. The main interferometer consists of a beamsplitter (BS), two arm cavities each of them formed by an input test mass (ITM) and an end test mass (ETM), and the power recycling mirror (PRM). Additional locking signals are obtained at the antisymmetric port, $S_{\rm anti}$, and by sampling a small amount of light from inside the power recycling cavity, $S_{\rm prc}$.

3 FEEDBACK COMPENSATION NETWORK

In order to implement feedback each degree-of-freedom which is under control of the compensation network has to be measurable. LIGO implements the Pound-Drever-Hall reflection locking technique [14] to keep cavities on resonance and a variant of this technique is used to control the angular degrees-of-freedom [15]. These techniques work well near resonance where they behave linearly but have a strong non-linear behaviour far way from resonance giving no or misleading signals. The first step of engaging the feedback compensation network is to catch the system on resonance with a highly sophisticated computer code [16] running on a digital control system.

A schematic view of the length control system for the common mode degrees-of-freedom is shown in Fig. 2. The signal $S_{\rm refl}$ measuring the common arm length of the interferometer is fed back to a combination of test masses, mode cleaner length and laser frequency to achieve the required laser frequency noise suppression of $< 10^{-6}{\rm Hz}/\sqrt{\rm Hz}$ in the frequency band of interest. To maintain maximum optical power in the system—and thus maximum signal to shot noise ratio—the control system must hold the common cavity length within $< 2\times10^{-12}$ m rms of its resonance point. A similar but less complicated system is deployed to control the differential degrees-of-freedom. Their the differential arm cavity length has to be held within $< 5\times10^{-14}$ m rms of its operating point to not pollute the gravitational wave signal with laser frequency noise.

4 CONCLUSIONS

So far LIGO has successfully demonstrated that the interferometer can be locked and kept on resonance for hours. The main goal in the near term is to improve the sensitivity which is still many orders of magnitude away from design, to engage the remaining feedback control paths and to fine-tune servo parameters.

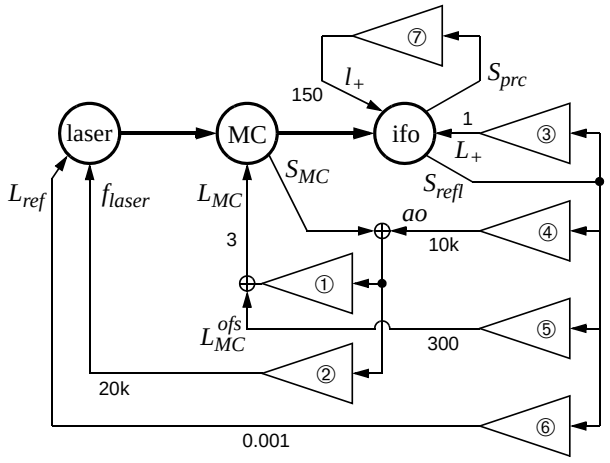


Figure 2: Common mode control system. The mode cleaner error signal, $S_{\rm MC}$, is split into two paths: the mode cleaner length path (1) feeding back to the position of a mode cleaner mirror, L_{MC}, and the laser path (2) feeding back to the laser frequency, $f_{\rm laser}$, using the VCO/AOM. The in-phase reflection signal, $S_{\rm refl}$, of the interferometer (ifo) is split into four paths: the arm cavity path (3) feeding back to the common arm cavity mirror positions, L_+, the additive offset (ao) path (4) feeding back to the error point of the mode cleaner control system, the mode cleaner length offset path (5) feeding back to the mode cleaner mirror position, L_{MC}^{ofs}, and the tidal path (6) feeding back to the reference cavity length, $L_{\rm ref}$, using the thermal actuator. The in-phase signal at the power recycling cavity port, $S_{\rm prc}$, is mostly sensitive to the power recycling cavity length, l_+, and is feed back to the recycling mirror position (7). The numbers in the feedback paths indicate unity gain frequencies in hertz.

5 REFERENCES

[1] A. Abramovici, W. Althouse, J. Camp, J.A. Giaime, A. Gillespie, S. Kawamura, A. Kuhnert, T. Lyons, F.J. Raab, R.L. Savage Jr., D. Shoemaker, L. Sievers, R. Spero, R. Vogt, R. Weiss, S. Whitcomb, and M. Zucker, "Improved sensitivity in a gravitational wave interferometer and implications for LIGO," Phys. Lett. **A218**, 157–163 (1996).

[2] B. Caron, A. Dominjon, C. Drezen, R. Flaminio, X. Grave, F. Marion, L. Massonnet, C. Mehmel, R. Morand, B. Mours, V. Sannibale, M. Yvert, D. Babusci, S. Bellucci, S. Candusso, G. Giordano, G. Matone, J.-M. Mackowski, L. Pinard, F. Barone, E. Calloni, L. DiFiore, M. Flagiello, F. Garuti, A. Grado, M. Longo, M. Lops, S. Marano, L. Milano, S. Solimeno, V. Brisson, F. Cavalier, M. Davier, P. Hello, P. Heusse, P. Mann, Y. Acker, M. Barsuglia, B. Bhawal, F. Bondu, A. Brillet, H. Heitmann, J.-M. Innocent, L. Latrach, C.N. Man, M. PhamTu, E. Tournier, M. Taubmann, J.-Y. Vinet, C. Boccara, P. Gleyzes, V. Loriette, J.-P. Roger, G. Cagnoli, L. Gammaitoni, J. Kovalik, F. Marchesoni, M. Punturo, M. Beccaria, M. Bernardini, E. Bougleux, S. Braccini, C. Bradaschia, G. Cella, A. Ciampa, E. Cuoco, G. Curci, R. DelFabbro, R. DeSalvo, A. DiVirgilio, D. Enard, I. Ferrante, F. Fidecaro, A. Giassi, A. Giazotto, L. Holloway, P. LaPenna, G. Losurdo, S. Mancini, M. Mazzoni, F. Palla, H.-B. Pan, D. Passuello, P. Pelfer, R. Poggiani, R. Stanga, A. Vicere, Z. Zhang, V. Ferrari, E. Majorana, P. Puppo, P. Rapagnani, and F. Ricci, "The VIRGO interferometer for gravitational wave detection," Nucl. Phys. **B54**, 167–175 (1997).

[3] K. Danzmann, "GEO 600 — A 600-m Laser Interferometric Gravitational Wave Antenna," in *First Edoardo Amaldi conference on gravitational wave experiments*, E. Coccia, G. Pizella and F. Ronga, eds. (World Scientific, Singapore, 1995), p. 100–111.

[4] K. Tsubono, "300-m Laser Interferometer Gravitational Wave Detector (TAMA300) in Japan," in *First Edoardo Amaldi conference on gravitational wave experiments*, E. Coccia, G. Pizella and F. Ronga, eds. (World Scientific, Singapore, 1995), p. 112–114.

[5] R. Weiss, "Electromagnetically coupled broadband gravitational antennae," MIT Res. Lab. Electron. Q. Prog. Rep. **105**, 54–76 (1972).

[6] J.-Y. Vinet, B.J. Meers, C.N. Man, and A. Brillet, "Optimization of long-baseline optical interferometers for gravitational-wave detection," Phys. Rev. **D38**, 433–447 (1988).

[7] B.J. Meers, "The frequency response of interferometric gravitational wave detectors," Phys. Lett. **A142**, 465–470 (1989).

[8] A.E. Siegman, *Lasers*, (University Science, Mill Valley, Calif., 1986), Chap. 13, p. 663.

[9] P. Fritschel, R. Bork, G. González, N. Mavalvala, D. Ouimette, H. Rong, D. Sigg, and M. Zucker, "Readout and control of a power-recycled interferometric gravitational wave antenna," Appl. Opt. **40**, 4988–4998 (2001).

[10] P. Fritschel, G. González, N. Mavalvala, D. Shoemaker, D. Sigg, and M. Zucker, "Alignment of a long-baseline gravitational wave interferometer," Appl. Opt. **37**, 6734–6747 (1998).

[11] J. Giaime, P. Saha, D. Shoemaker, and L. Sievers, "A passive vibration isolation stack for LIGO: design, modeling, and testing," Rev. Sci. Instrum. **67**, 208–214 (1996).

[12] A. Gillespie and F. Raab, "Thermal noise in the test mass suspensions of a laser interferometer gravitational-wave detector prototype," Phy. Lett. **A178**, 357–363 (1993).

[13] T. Lay, and T.C. Wallace, *Modern global seimology*, (Academic Press, San Diego, California, 1995), p. 179.

[14] R.W.P. Drever, J.L. Hall, F.V. Kowalski, J. Hough, G.M. Ford, A.J. Munley, and H. Ward, "Laser phase and frequency stabilization using an optical resonator," Appl. Phys. **B31**, 97–105 (1983).

[15] Y. Hefetz, N. Mavalvala, and D. Sigg, "Principles of calculating alignment signals in complex optical interferometers," J. Opt. Soc. Am. **B14**, 1597–1605 (1997).
D. Sigg, and N. Mavalvala, "Principles of calculating the dynamical response of misaligned complex resonant optical interferometers," J. Opt. Soc. Am. **A17**, 1642–1649 (2000).

[16] M. Evans, N. Mavalvala, P. Fritschel, R. Bork, R. Gustafson, W. Kells, M. Landry, D. Sigg, R. Weiss, S. Whitcomb, and H. Yamamoto, "Lock acqusition of a gravitational wave interferometer," submitted to Opt. Lett. (2001).

WECT004
physics/0111087

A DISTRIBUTED FEEDBACK SYSTEM FOR RAPID STABILIZATION OF ARBITRARY PROCESS VARIABLES*

B. Bevins and A. Hofler, Jefferson Lab, Newport News, VA 23606, USA

Abstract

In large process control systems it frequently becomes desirable to establish feedback relationships that were not anticipated during the design phase of the project. The "Generic Lock" architecture discussed in this paper makes it possible for system operators to implement new feedback loops between arbitrary process variables quickly and with no disturbance to the underlying control system. Any available process variables may be selected for the input and output of the loops so created, regardless of their physical or logical separation. The system allows multiple user interface points distributed through the control system while ensuring consistency among the feedback loops. This system can be used to quickly prototype and test new control philosophies or to control temporary hardware arrangements without any new software development. It is implemented at the Thomas Jefferson National Accelerator Facility using the Common Device (CDEV) [1] framework on top of the Experimental Physics and Industrial Control System (EPICS) [2]. This paper discusses the architecture, implementation, and early usage of the system

1 INTRODUCTION

One of the fundamental entities in any digital process control system is the discrete closed-loop feedback controller. Though many such controllers are typically designed into a large system, it is difficult to anticipate all possible useful feedback relationships. Hardware details, operating modes, and even control philosophies may evolve over time to meet changing needs. Control systems must have the flexibility to evolve with them.

A Generic Lock Server has been developed at Jefferson Lab that enables on the fly creation and configuration of feedback loops using any available control system I/O signals. With this system, feedback loops can be created as prototypes for new control ideas or to satisfy temporary operational requirements with absolutely no disruption of the existing control system and no new programming effort required. It builds upon previous work at the lab in software server based process control [3]. At present only single-input, single-output (SISO) loops are available, but multiple-input, multiple-output (MIMO) loops will be added in the near future.

2 DESIGN

2.1 Background

The control system software for the CEBAF accelerator at Jefferson Lab is implemented in two layers: EPICS databases running on the input/output controllers (IOC's) and programs running on Unix hosts that communicate with the IOC's by Channel Access (CA). Among the latter are programs that implement feedback loops to stabilize various machine parameters against disturbances at < 1Hz [4]. They are referred to as "Slow Locks" to distinguish them from the Fast Feedback System, which uses dedicated hardware to achieve much faster sampling rates [5]. They come in various distinct flavors: orbit locks, energy locks, current locks, and helicity-correlated asymmetry locks. Some calculate their feedback gains with data from on-line accelerator model servers. Others empirically measure their response functions during an explicit calibration step [6]. Still others use operator-entered gains that have been calculated off-line or optimized interactively. Though they share some code, each lock type exists in its own dedicated server with its own GUI. Adding new lock types has been difficult and time consuming.

2.2 The Generic Lock Server

An effort is now underway to unify the various lock types into a common architecture that will be more easily maintainable and extensible. The first fruit of this effort is the Generic Lock Server. This server can host multiple lock types simultaneously and allows the interactive creation and destruction of new locks at runtime. It stores configuration information in a human readable Extensible Markup Language (XML) file. The Generic Lock Server presently handles a new class of general purpose Proportional, Integral, Derivative controller (PID) locks as well as the specialized current and asymmetry locks.

2.3 The PID Locks

The functionality of the PID locks is derived from the EPICS CPID record, extended to allow both the input and output signals to be arbitrarily specified at runtime. Any variable in the control system accessible through CDEV

* Work supported by the U. S. Department of Energy, contract DE-AC05-84-ER40150

can be used. This includes all CA signals as well as signals from other CDEV servers, including model servers and other locks.

A GUI allows users to control all the PID locks and create new ones from anywhere on the network. Once a new lock has been created, the user enters names for input and output of the feedback loop, PID gain parameters, and output signal limits. The lock can then be activated.

3 IMPLEMENTATION

3.1 Server Architecture

The lock server programs are implemented in C++ as CDEV Generic Servers [7] where each lock is a virtual device that exposes a set of virtual attributes containing the lock's operating parameters. The Generic Lock Server uses this same setup, but takes advantage of more recent developments in C++. It uses the containers and strings of the C++ standard library (formerly the standard template library) to avoid memory management issues. This has resulted in a very robust product being developed in much less time that would otherwise be possible.

Figure 1 shows the relationships among the main classes used by the Generic Lock Server. In order to

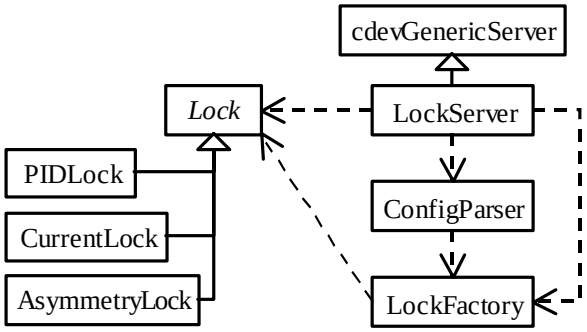


Figure 1: Simplified Lock Server Class Diagram

minimize coupling among the components, the LockServer engine uses a LockFactory class to construct locks of various types. The abstract Lock class maintains a private list of all the locks that have been instantiated and controls access to them. Each derived lock type registers itself with the factory at initialization. All that is necessary to add a new lock type is to relink the server with the new object file for the lock type. The ConfigParser and LockFactory classes are singletons.

3.2 Configuration Files

The XML configuration file is both parsed and written using the "non-commercial" Qt/X11 toolkit from Trolltech [8]. The Qt XML Document Object Model (DOM) interface is encapsulated in the ConfigParser class used by the LockServer. A partial XML configuration file is shown below.

Within the main <lockConfig> element, there are zero or more <Lock> elements and zero or more <device> elements. Each <Lock> triggers the construction of a new lock of the specified type with the specified name. Each <device> contains zero or more <attribute> elements. The parser attempts to map the names of each <device>/<attribute> pair into a CDEV device/attribute existing in the server and set its value accordingly.

```
<lockConfig>
 <Lock name="PIDLock02" type="PIDLock" />
 <device name="PIDLock02" >
  <attribute name="GainD" value="0" />
  <attribute name="GainI" value="1" />
  <attribute name="GainP" value="0" />
  <attribute name="InputName"
            value="ILI1L_PHASEerror"/>
  <attribute name="Interval" value="4" />
  <attribute name="MaxChange" value="0.1" />
  <attribute name="MaxPos" value="25" />
  <attribute name="MinPos" value="15" />
  <attribute name="OutputName" value="R1XXPSET" />
  <attribute name="SetPoint" value="0" />
  <attribute name="Description" value="North Linac First
            Pass Gang Phase" />
 </device>
<!--More locks of various types would follow here. ->
</lockconfig>
```

3.3 User Interface

The GUI's for the slow locks are scripted in Tcl/Tk. The PID Lock GUI is shown in Figure 2. All the PID locks are presented in a scrollable list and each one can

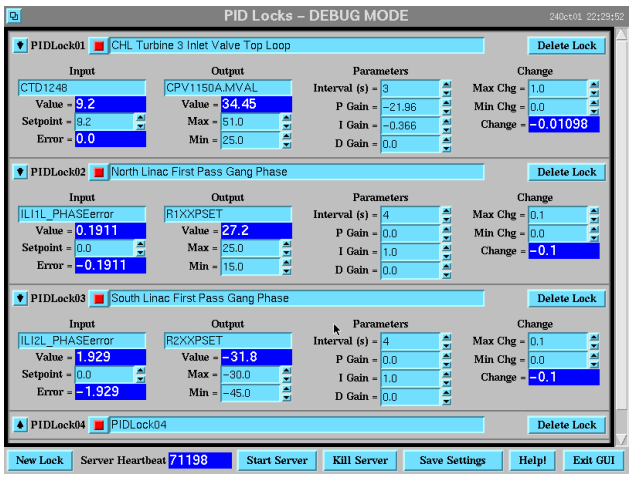


Figure 2: The PID Lock GUI

be collapsed or expanded using the arrow buttons on the left. Output values from the locks are in the darker boxes while user inputs are in the lighter boxes. The button to the left of each lock's textual description field turns the lock on and off. A lock can be deleted using the button in the upper right corner of its display. A new lock is created using the button in the far lower left. The new

lock is assigned the next available device name, first reusing the slots of any deleted locks. When a lock is deleted or a new lock is created, every instance of the GUI running anywhere on the network is notified to update its display.

4 FUTURE DIRECTIONS

4.1 Additional Lock Types

The remaining slow lock types in use at Jefferson Lab will be integrated into the Generic Lock Server architecture. For this to happen the architecture must be extended to allow MIMO lock types. It must also allow for locks that must be calibrated at runtime like the Orbit Locks and locks that use the on-line accelerator model server for calibration like the Energy Locks.

4.2 Server Security

Access to all CEBAF IOC's is controlled through CA security. To function, the Generic Lock Server must run with a user ID that is allowed to write to the IOC's. However, because it is designed to be flexible, the CDEV Generic Server has no built-in security model. This means that any user connected to the accelerator network can write to the virtual attributes of the lock devices and create a lock that writes to any channel in the system, effectively circumventing CA security. Plugging this loophole is a top development priority. A security layer will be added to the lock server allowing writes only by specific groups of users from specific machines. In the longer term, it would be useful if the attributes of a lock would inherit the security of the lock's output channel.

4.3 PID Auto-tuning

Determining appropriate gains for PID controllers is a non-trivial task. To make the PID Lock class more usable by non-experts it would be very desirable to have a means to automatically characterize the closed loop transfer function and determine appropriate PID gains for optimum stability, if they exist. Many such algorithms exist [9]. If one can be found that is sufficiently general, it might be integrated into the lock server architecture or could exist as a separate process, making it more useful for tuning EPICS CPID records as well.

4.4 Dynamic Linking

The very low degree of coupling among the various lock and server classes means that it is not necessary to statically link the lock classes with a server. Following the model that CDEV uses with its service classes, the code for individual lock types could be dynamically loaded as needed. This would allow completely new types of locks to be added to a running server without even restarting it, much less rebuilding it.

4.5 Expanding the Lock Namespace

One limitation imposed by CDEV is that all device names must be declared in static Device Definition Language (DDL) files that are used by clients to map device names to the appropriate services and servers. Thus, although new locks can be created in the server with arbitrary names, clients cannot find them unless they use names pre-declared in a common DDL file. This prohibits the use of descriptive device names for dynamically created locks. A dedicated client like the Generic Lock GUI could be designed to dynamically discover the lock names, but they would still not be available to general purpose clients like archivers.

CDEV can be configured to fall through to a particular service when a device name is not found in the DDL file. For many sites that use CDEV with EPICS, this default is configured to use CA. The Portable CA Server (PCAS) [10] suggests a way to make fully dynamic virtual devices. A PCAS could be set up to host the virtual device/attribute pairs needed by such devices.

5 ACKNOWLEDGEMENTS

Recognition of the need for the Generic Lock Server and the PID Locks as well as ideas for their implementation grew out of many discussions with the members of the Jefferson Lab Operations Group and the Beam Applications Team, especially Yves Roblin.

6 REFERENCES

[1] J. Chen, G. Heyes, W. Akers, D. Wu and W. Watson, "CDEV: An Object-Oriented Class Library for Developing Device Control Applications", Proceedings of ICALEPCS95, Chicago.

[2] L. R. Dalesio, et. al., "The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future", Proceedings of ICALEPCS93, Berlin.

[3] M. Bickley, B. A. Bowling, D. A. Bryan, J. van Zeijts, K. S. White and S. Witherspoon, "Using Servers to Enhance Control System Capability", Proceedings of PAC 1999, New York.

[4] J. van Zeijts, S. Witherspoon and W. A. Watson, "Design and Implementation of a Slow Orbit Control Package at Thomas Jefferson National Accelerator Facility", Proceedings of PAC 1997, Vancouver.

[5] R. Dickson and V. Lebedev, "Fast Feedback System for Energy and Beam Stabilization", Proceedings of ICALEPCS99, Trieste, Italy.

[6] A. Hofler, D. Bryan, L. Harwood, M. Joyce and V. Lebedev, "Empirically Determined Response Matrices for On-line Orbit and Energy Correction at Thomas Jefferson National Accelerator Facility", Proceedings of PAC 2001, Chicago.

[7] W. Akers, "An Object-Oriented Framework for Client-Server Applications", Proceedings of ICALEPCS97, Beijing.
[8] Trolltech AS, http://www.trolltech.com/products/qt/
[9] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed., pp.375-389, Addison-Wesley, 1995.
[10] J. O. Hill, "A Server Level API for EPICS", Proceedings of ICALEPCS95, Chicago.

WECT005
physics/0112042

ACTIVE VIBRATION SUPPRESSION R&D FOR THE NLC

J. Frisch, L. Hendrickson, T. Himel, A. Seryi, SLAC, Stanford, CA 94025, USA

Abstract

The nanometer scale beam sizes at the interaction point in linear colliders limit the allowable motion of the final focus magnets. We have constructed a prototype system to investigate the use of active vibration damping to control magnet motion. Inertial sensors are used to measure the position of a test mass, and a DSP based system provides feedback using electrostatic pushers. Simulation and experimental results for the control of a mechanically simple system are presented.

1 STABILIZATION REQUIREMENTS

The beam size at the interaction point of the NLC is approximately 2 X 200 nanometers (for 1 TeV CM system) [1,2]. In order to maintain luminosity, the relative positions of the beams at the IP must be stabilized to approximately 1 nanometer.

Vibration of the final focus quadrupoles is a potentially serious source of beam motion at the IP. Ground motion is one of the causes for this vibration. While ground motion in a quiet site could be sufficiently small, the "cultural" noise from man-made sources may substantially change the ground motion spectrum. Moreover, noises of the accelerator complex itself will contribute to this spectrum. Measurements at existing labs, showing 1.5nm of RMS motion above 1Hz at LEP [3] and 70nm at HERA [4], illustrate the magnitude of uncertainty of the initial conditions.

The NLC is being designed with careful engineering consideration of vibration issues. However, stabilization requirements for the final quadrupoles are hard to predict until the latest stage of design. Therefore, rather than to build a stabilization system to meet a specific requirement, work is underway to develop general stabilization technologies to be implemented at the final stage of NLC design.

2 STABILIZATION TECHNOLOGIES

Beam Based Systems. The interaction of the electron and positron beams at the IP causes a beam deflection that is related to the beam offset. This allows the offset to be measured to a fraction of the spot size at the beam rate of 120Hz. This beam deflection provides the only long-term measure of the relative positions of the beams.

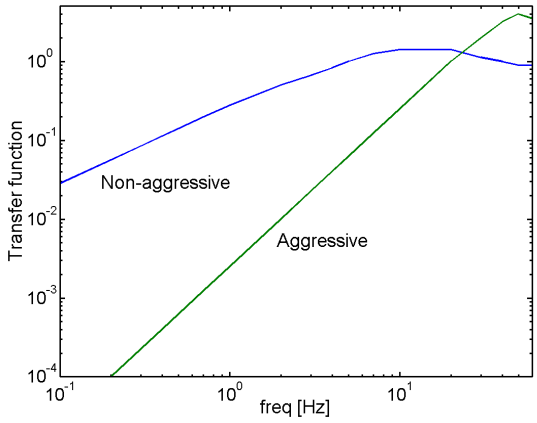


Figure 1: Simulated effect of beam-beam feedbacks.

A variety of feedback algorithms can be used with the beam – beam deflection data, with the selection based on the trade off between low frequency attenuation, and high frequency amplification of noise. Figure 1 shows the simulated effects of "aggressive" and "non-aggressive" beam feedbacks.

For the non-aggressive feedback the noise attenuation at 1Hz is 4 times, while the maximum amplification is 1.3 at 10-20Hz. For the aggressive feedback, the attenuation at 1Hz is 400 and the amplification at 50-60Hz is 4 times. Feedback algorithms with intermediate performance are also possible. One should note that the beam-beam feedback performance is ideal, i.e. it does not account for BPM resolution or jitter of beam size.

Very fast intra-train beam – beam feedback is also under consideration. This system relies on sufficient bandwidth and low latency to allow operation within the 400 nanoseconds NLC bunch train [5].

Interferometer Based Systems. Optical interferometers can be used to measure the distance between the final focus magnets and a fixed point or points on the ground. These "optical anchor" systems were proposed in the NLC ZDR [2] and are being developed at the University of British Columbia [6].

Inertial Based Systems. Inertial sensors can be used to measure the motion of the final quadrupoles relative to the "fixed stars". At low frequencies the position noise of an inertial sensor must rise sufficiently slow and a transition to a beam based system must be made.

A simple inertial vibration stabilization system has been constructed and is being used to test feedback hardware and algorithms.

3 VIBRATION FEEDBACK TEST SYSTEM DESIGN

Mechanical Design. An approximately 30kg aluminum block is suspended on 6 springs which define the 6 solid body degrees of freedom (see Figure 2). Six inertial sensors are used to provide feedback signals; a seventh measuring vertical motion is used to evaluate the feedback performance. Seven actuators, with two operated in parallel, are used to control the motions of the block.

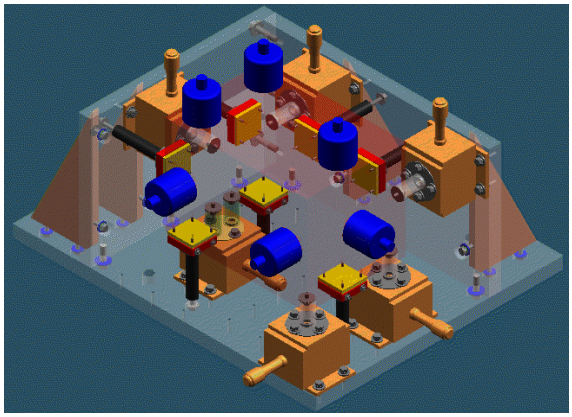


Figure 2: Drawing of the test system.

There is a trade-off on the support spring stiffness. High stiffness springs, which result in high normal mode frequencies, provide relatively low amplitude motion and good stability in the absence of feedback. They have the disadvantage of coupling high frequencies to the supported object, which are difficult to control with feedback.

Low stiffness springs allow large amplitude low frequency motions, but attenuate high frequencies. This approach was chosen for the test system, with normal mode frequencies in the range of 3-10Hz.

Actuator Design. For a system with a low stiffness support, the actuators must also have low stiffness. In order to close the feedback loop with high gain, the time delay of the actuators must also be short compared with other time constants in the system. While piezo-electric actuators are commonly used for this type of application, it is difficult for them to meet the above requirements.

We chose pure electrostatic actuators. A pair of electrodes, each 5x5cm, separated by approximately 1 mm, is used. With a maximum voltage of 1000V, the force produced is approximately 0.01 Newtons – sufficient to stabilize the mass motion. The response time is limited by the high voltage amplifiers (Trek 601C) which have a large signal bandwidth > 8 KHz.

Sensors. The test system uses commercial compact geophones (Geospace HS-1) with a 4.5Hz resonance and low noise (Analog Devices AD624 based) amplifiers. The resulting system has a (calculated) noise level of ~2nm/Hz$^{1/2}$ at 1Hz. The geophones are sensitive to the velocity of a test magnet inside of a pick-up coil, so their sensitivity decreases as 1/f^3 below resonance.

In the NLC the final quadrupoles will be mounted inside the multi-Tesla detector solenoid, and will require non-magnetic inertial sensors. In addition, the noise level of the commercial geophones is currently limiting the performance of the test system.

New capacitive, non-magnetic accelerometers are being developed for this project. Electronics has been demonstrated with a sensitivity of 0.01nm/Hz$^{1/2}$ [7]. Mechanical design of a sensor is underway. This system is expected to be a factor of 50 lower noise at 1Hz, and 500 lower noise at 0.1Hz than the compact geophones currently in use.

Data Acquisition System. The data acquisition system is constructed from VME format hardware. A 16 bit 250KHz, 8 channel A-D / D–A card (Pentek 6102) is used to interface with the sensors and actuators. Closed loop feedback is performed with a TMS320C40 50MHz DSP (Pentek 4284) which interfaces to the A-D / D-A using a C40 port. This interface removes the real time communication from the VME backplane. A 68040-based crate controller (Motorola MVME167) running VxWorks is used for program loading and non-realtime data communications. The system is interfaced to Matlab for algorithm development.

4 TEST SYSTEM ALGORITHM

Calibration and Orthogonalization. Each of the 6 actuators is driven at each of 110 frequencies, while each of the sensor outputs is measured. 50 measurement frequencies are logarithmically spaced from 1 - 30Hz. 10 are clustered near each of the resonance frequencies.

A 96 parameter fit is then done to the normal mode frequencies and Qs, sensor frequencies and Qs, actuator to mode couplings, and sensor to mode couplings. Figure 3 shows the quality of the fit. The resulting fit is tested by driving the combination of actuators corresponding to a single mode, and measuring the corresponding set of sensors. The lack of resonant peaks for the other modes indicates that the orthogonalization is good (Figure 4).

Feedback algorithm. The sensor measurements are converted to orthogonal modes. Then six independent feedback loops control vibration in each mode.

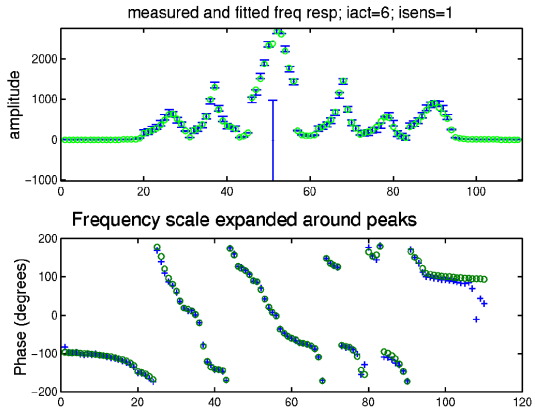


Figure 3: One of 36 measured calibration curves. The fit is shown by green circles.

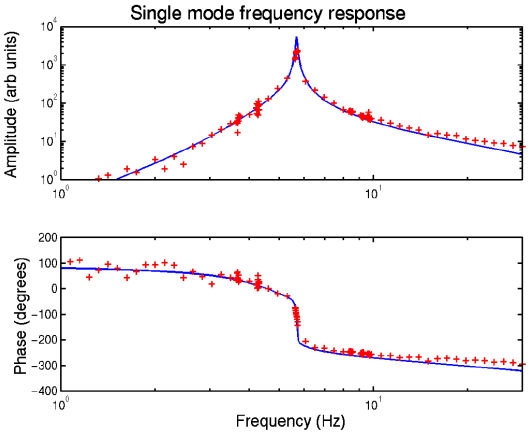


Figure 4: Frequency response of single mode. Measurements (symbols) and model.

The measured frequency response and noise spectral densities are used as inputs to generate an optimal feedback. Note that in the present system expected rather than measured noises are used, and identical mode frequency responses are assumed in order to simplify testing.

The feedback can be implemented as either state space, or transfer function (ZPK). State space is more numerically stable, but less computationally efficient. Results described are for state space running at a loop speed of 500Hz (not believed to be a limit on performance) with the gain set to 0.4 times optimal. The reduced gain is necessary to avoid instabilities. These instabilities are under study, but may be due to actuator saturation during system start-up.

System Performance. The test system performance is measured using the test sensor that measures vertical motion of the block. One should note that the test system is located in a fairly noisy lab environment – comparable to the "HERA" site, and that the noise of this test sensor becomes significant below ~1Hz.

The suspension of the block attenuates high frequencies, but introduces a low frequency resonance. The inertial feedback largely eliminates this resonance (see Figure 5).

In order to evaluate the performance of the system we can simulate the effect of beam-beam feedback on the measured output of the test sensor (lines with symbols in Figure 5). One can see that in a somewhat quieter place the aggressive beam feedback would already satisfy our requirements. The applicability of the aggressive feedback may however be limited by presence of uncorrelated jitter in the incoming beam.

Further improvements of the system performance are expected from optimization of the algorithm and in particular from new less noisy inertial sensors.

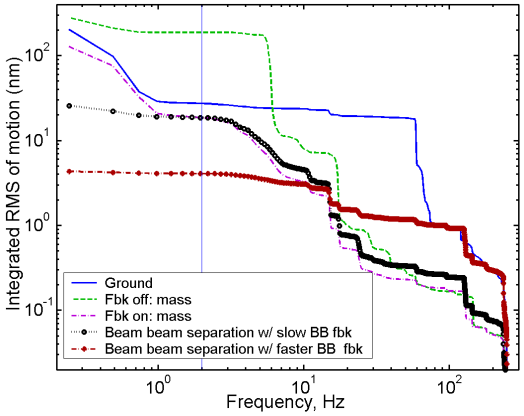


Figure 5: Integrated power spectrum showing performance of the test system.

5 FUTURE PLANS

The present system is mechanically simple, with the solid body normal mode frequencies of the block much lower than any internal resonant frequencies. When the operation of this system is well understood, an elongated structure with internal mode frequencies comparable to those of a real final focus magnet will be stabilized.

REFERENCES

[1] The NLC Design Group, "The Zeroth Order Design Report for the Next Linear Collider" SLAC-494
[2] 2001 Report on the NLC, SLAC-R-571, 2001
[3] V. Juravlev et al., CERN-SL-93-53, 1993
[4] C. Montag et al., DESY HERA 95-06, 1995
[5] S. Smith "Design of an NLC Intrapulse Feedback System, 2001, SLAC LCC note 0056
[6] T. Mattison, in SLAC-WP-18, 2001, p. 567
[7] R. Lansom, "Electronics Testing for Nanometer Vibration Reading on the Next Linear Collider Project", EURLF summer intern program. www.slac.stanford.edu/grp/th/eurls-sise/1999/

WECT006
physics/0111180

BEAM FEEDBACK SYSTEMS AND BPM READ-OUT SYSTEM FOR THE TWO-BUNCH ACCELERATION AT KEKB LINAC

K. Furukawa*, N. Kamikubota, T. Suwada
High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki, 305-0801, Japan
T. Obata
Mitsubishi Electric System Service, Tsukuba, Ibaraki, 305-0045, Japan

Abstract

In order to double the positron injection rate into the KEKB ring, a two-bunch acceleration scheme has been studied at the linac, in which bunches separated by 96 ns are accelerated in 50 Hz. In this scheme the stabilization of energy and orbit of each bunch is indispensable. Thus the beam energy and orbit feedback systems have been upgraded.

Since beam characteristics are acquired through beam position monitors (BPM), their read-out system was improved to meet two-bunch requirements. Combined waveforms from BPM's were adjusted with delay cables avoiding overlaps and they enabled simultaneous measurement of beam positions of two bunches.

The beam energies of two bunches were balanced by tuning rf pulse timings, and the average energy was stabilized by adjusting the accelerating rf phases. The average beam orbits were also stabilized. Slow feedback systems at the injector section for charge and bunching stabilities are being planned as well. These systems were used successfully in the test beams and will be employed in the routine operation.

1 INTRODUCTION

The electron/positron linac at KEK injects 8-GeV electron and 3.5-GeV positron beams into KEKB rings, where CP-violation study is carried. Since the efficiency of the experiment increases by shortening the injection time, several mechanisms have been introduced to accomplish it [1, 2]. Especially much effort has been made to improve of the positron injection time, since it is longer compared with electron [3].

One of such effort is a two-bunch acceleration plan, which have been studied and applied [4, 5]. In this scheme two bunches of positron are accelerated in one rf pulse, which is 50 Hz, and they may double the injection rate. The time space between two bunches is, however, restricted by the rf frequencies of linac and rings, and the smallest space is 96.29 ns since the common frequency is 10.38 MHz. Thus a precise beam control and diagnosis are necessary.

The beam diagnosis so far has been made by employing strip-line-type beam position monitors (BPM), wire scanners for transverse profiles and streak cameras for longitudinal profiles. In order to maintain the stable beams, it is essential to have these beam instrumentations work for both two bunches. The two-bunch read-out of BPMs is especially important, because it is used in number of orbit and energy feedback loops to stabilize beams.

* e-Mail: <kazuro.furukawa@kek.jp>

2 BPM AND READ-OUT SYSTEM

Along the 600-m linac, 90 BPMs are installed and their signals are transferred to one of 18 measurement stations. Signals are delayed and combined so as not to overlap each other, and are fed into a 5-Gs/s waveform digitizer (Sony-Tektronix TDS-680B/C) as in Fig. 1 [6]. Although the BPM signal is fast bipolar signal, the readout precision is optimized using the interpolation function of the digitizer. All 18 digitizers are triggered by a single distributed signal which is synchronized with beam repetition and rf frequencies.

The waveform is read through the GPIB and a signal from each electrode is analyzed with predetermined response function once per second by a VME computer (Force 68060). Response functions include 3rd-order position-mapping functions, attenuation factors of various components and position offsets against the center of corresponding quadrupole magnet derived from a beam-based alignment.

Since the timing and amplitude ranges of BPM signals are different depending on the beam modes and locations, the process is driven by a control database system [7].

Acquired beam positions at 18 stations are sent to central computers once per second and are served for various beam-energy and orbit feedback systems to maintain stable beam operation.

3 TWO-BUNCH OPERATION

The BPM system was improved for two-bunch operation.

3.1 Improvements to BPM System

As written above, it is important to acquire beam positions of two bunches along the linac simultaneously to study beams.

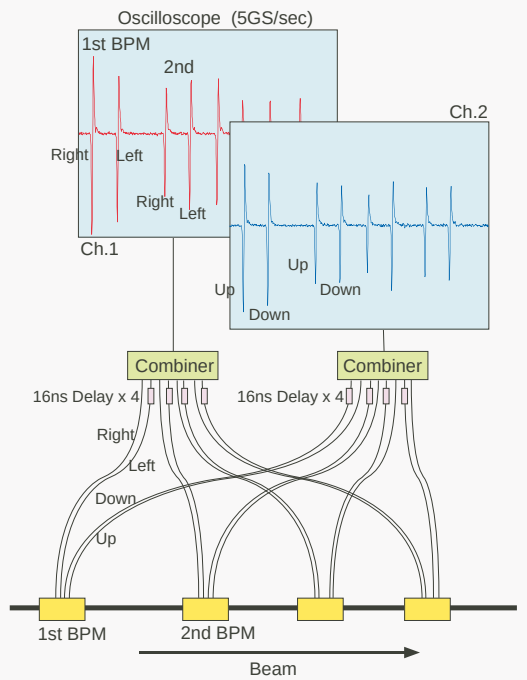


Figure 1: Signals from multiple BPMs are arranged so that combined waveforms can be processed properly.

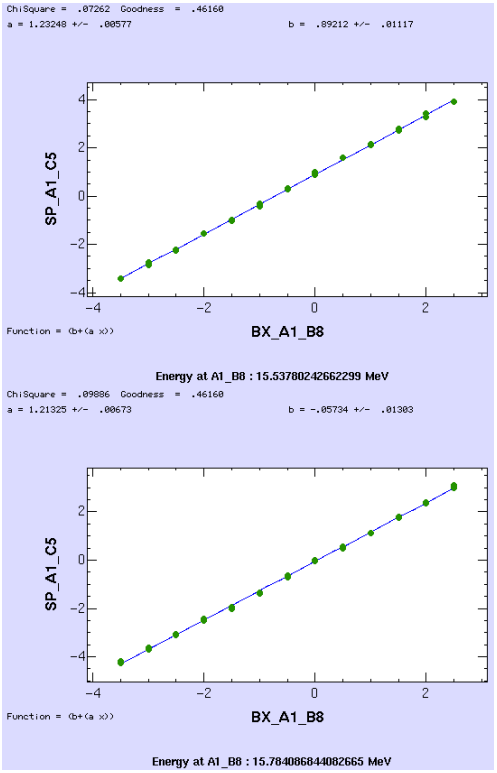


Figure 2: A software panel to measure the beam energies of two bunches with a steering magnet (BX_A1_B8) and a BPM (SP_A1_C5). After adjusting the timing system, they became almost the same energy 15.5 MeV and 15.8 MeV.

In our instrumentation, signals from those two bunches appear as two signals separated by 96.29 ns on the waveform. Although it was sometimes necessary to add more delay lines so as to avoid waveform overlapping, there was no need to add specific hardware to handle such signals with small separations.

Calibration factors were re-examined since delay lines were added, and beam-timing database for the signal analysis was extended to accommodate two-bunch information.

Processing functions/commands for BPMs on the central computers are also extended or added for two bunches, keeping old functions as before for single-bunch operations.

With these modifications the BPM processing system was extended for two-bunch operation without any performance loss in both precision and speed. It has been used in beam operation since March 2001.

3.2 Operation Software with BPM

Most of operation software which utilize the BPM information was extended to meet both single- and two-bunch operations. One of such examples is Fig. 2, which measures beam energies of two bunches by correlation between a steering-magnet field and the beam-position response at the bunching section.

3.3 Two Bunch Controls

In order to accelerate the beams properly, the beam characteristics of two bunches need to be adjusted to be the same. For example, in order to adjust the beam energy differences, we change the beam timing and rf pulse timing. The beam

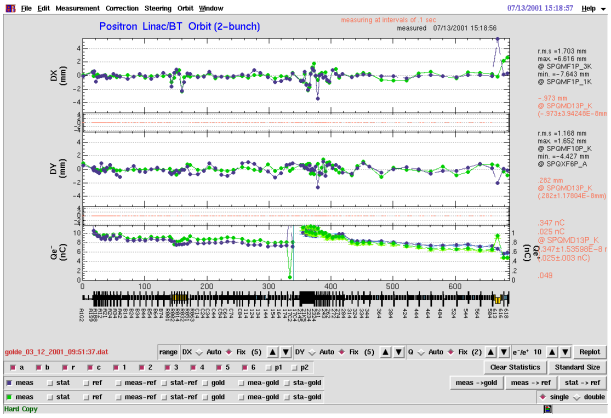


Figure 3: Beam orbit and charge of electrons (left) and positrons (right) along the 600-m linac. Two slightly different lines indicate the first and the second bunches.

timing can be changed by a 10-ps step [4] and the rf pulse timing can be changed by a 1.75-ns step at each sector independently. Most of other parameters in the linac are not sensitive against the time separation of 96.29 ns.

With such adjustments the 10-nC primary electron bunches are accelerated up to 3.7 GeV and positrons are generated as shown in Fig. 3.

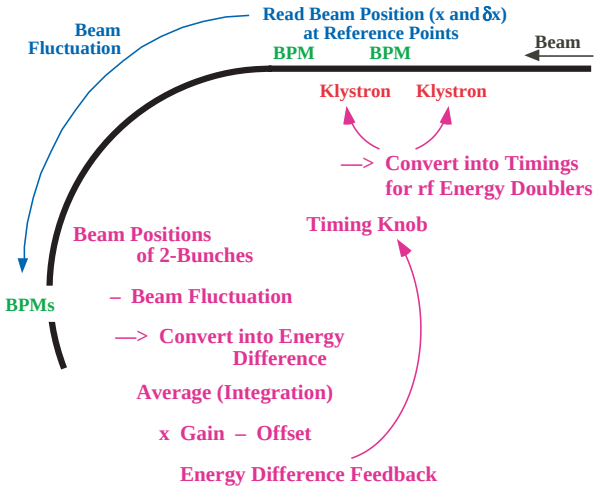


Figure 4: Energy-difference feedback loop at the J-arc.

3.4 BEAM FEEDBACK LOOPS

The beam feedback loops in the linac for energy and orbit stabilization [2] were also extended to control two-bunch beams. Since we don't have many mechanisms to control two bunches independently, most feedback loops were modified to use positions derived from charge-weighted averages of two bunches. With these changes those loops can maintain the average orbit and energy. In software, only the monitoring function was extended to read average positions if two bunches are accelerated. For the positron injection about 20 beam feedback loops are used, and they are all extended for two bunches.

While normal energy and orbit feedback loops use charge-weighted average positions, feedback loops to minimize energy differences use a position difference between two bunches as shown in Fig. 4.

Although the energy difference does not change frequently, such loops stabilize the beam in a long term.

4 SUMMARY

The data-acquisition system for the linac BPMs was upgraded to provide beam positions in two-bunch operation without losing any original features. Along with improvements of the streak camera and wire scanner systems, it has been still indispensable to study and operate on linac beams. The system is also used by many operation software including beam-energy and orbit feedback systems.

5 REFERENCES

[1] K. Furukawa *et al.*, "Beam Switching and Beam Feedback Systems at KEKB Linac", *Proc. of LINAC2000*, Monterey, USA., 2000, p.633.

[2] K. Furukawa *et al.*, "Energy Feedback Systems at KEKB Injector Linac", *Proc. of ICALEPCS99*, Trieste, Italy, 1999, p.248.

[3] K. Furukawa *et al.*, "Towards Reliable Acceleration of High-Energy and High-Intensity Electron Beams", *Proc. of LINAC2000*, Monterey, USA., 2000, p.630.

[4] S. Ohsawa *et al.*, "Increase of Positrons by High-intensity Two-bunch Acceleration Scheme at the KEKB Linac", to be published in *Proc. of PAC2001*, Chicago, USA., 2001.

[5] Y. Ogawa *et al.*, "Two-Bunch Operation of the KEKB Linac for Doubling the Positron Injection Rate to the KEKB Ring", to be published in *Proc. of APAC2001*, Beijing, China, 2001.

[6] T. Suwada *et al.*, "Stripline-Type Beam-Position-Monitor System for Single-Bunch Electron/Positron Beams", Nucl. Instr. and Meth. **A440** (2000) 307.

[7] N. Kamikubota *et al.*, "Data Acquisition of Beam-Position Monitors for the KEKB Injector-Linac", *Proc. of ICALEPCS99*, Trieste, Italy, 1999, p.217.

Wednesday, 28 November 2001

WED - Configuration and Databases

WEDT001
physics/0111084

DECISION SUPPORT FACILITY FOR THE APS CONTROL SYSTEM*

D. A. Dohan, Advanced Photon Source, ANL, Argonne, IL 60561, USA

Abstract

The Advanced Photon Source is now in its fifth year of routine beam production. The EPICS-based [1] control system has entered the phase in its life cycle where new control algorithms must be implemented under increasingly stringent operational and reliability requirements. The sheer volume of the control system (~270,000 records, ~145 VME-based input-output controllers (IOCs), and ~7,000,000 lines of EPICS ASCII configuration code), presents a daunting challenge for code maintenance.

The present work describes a relational database that provides an integrated view of the interacting components of the entire APS control system, including the IOC low-level logic, the physical wiring documentation, and high-level client applications. The database is extracted (booted) from the same operational CVS repository as that used to load the active IOCs. It provides site-wide decision support facilities to inspect and trace control flow and to identify client (e.g., user interface) programs involved at any selected point in the front-end logic. The relational database forms a basis for generalized documentation of global control logic and its connection with both the physical I/O and with external high-level applications.

1 INTRODUCTION

The Advanced Photon Source facility has been in existence for about ten years, with the past five years under full beam production. The facility consists of an electron linac, followed by an accumulator ring and a synchrotron, injecting into an 1100-m storage ring at 7 GeV. There are typically between 20 and 30 parallel user experiments running at any time. After storing beam, the injector linac is used for free-electron laser development. Recently the storage ring has switched to top-up operation, in which beam is injected at 2-minute intervals, keeping the circulated beam current constant to within 1%. In order to allow FEL operation in parallel with top-up operation, an interleaving system is under development, allowing rapid switching of the linac beam between injection and FEL operation.

Concurrent with these ongoing requests for new beam capabilities and control features has been an increased emphasis on reliable operation, both in terms of reducing beam off time (MTTR) and perhaps more importantly, in terms of reducing the frequency of beam dropout (MTBF). Changes in the large, installed, operational code base cannot be made without detailed understanding of the software and the behavior of the components of the control system.

The present paper describes work in progress at the APS to create a facility to assist the control systems application developer in analyzing the installed, operational control system. This decision support system utilizes a relational database approach to capture and query the controls knowledgebase.

2 THE APS CONTROL SOFTWARE

The main backbone of the APS accelerator control system [2] consists of a network of 145 front-end IOCs. Connected to the IOC layer is a network of operator interface workstations. Operators interact with the accelerator equipment mainly through MEDM [3] display files. Logging and automation tasks are handled using SDDS- and Tcl/TK-based tools. At the machine and equipment interface, embedded controllers are programmed to provide fixed well-defined functionality within limited control subdomains.

The subject of this paper is the middle, integration layer, which uses the EPICS real-time control system software.

3 PROCESS VARIABLE RELATIONAL DATABASE

3.1 The Relational Schema

EPICS software is based on the concept of ‘records’ – executable code units that perform hardware I/O. Other types of records (e.g., sequence, calc, fanout,) provide control logic capability. Each record has a well-defined set of ‘fields,’ commonly referred to as process variables or PVs. External clients may connect to these PVs, to read or optionally set their values. A particular type of field is a ‘link’ field, which is used to control the flow of logic between EPICS records. Records relating to a subsystem of the accelerator complex are grouped and stored in text files

* Work supported by the U.S. Department of Energy, Office of Basic Energy Sciences under Contract No. W-31-109-ENG-38.

known as 'db files.' These db files assist the developer in organizing and partitioning software into manageable units of related control software. The IOC does not retain knowledge of the source db file from which any record is derived.

To provide an integrated view of the entire control system logic, a project was initiated to model the installed EPICS control software in a relational schema, using the ORACLE relational database (rdb). A simple schema was adopted, as shown in entity relation diagram of Figure 1.

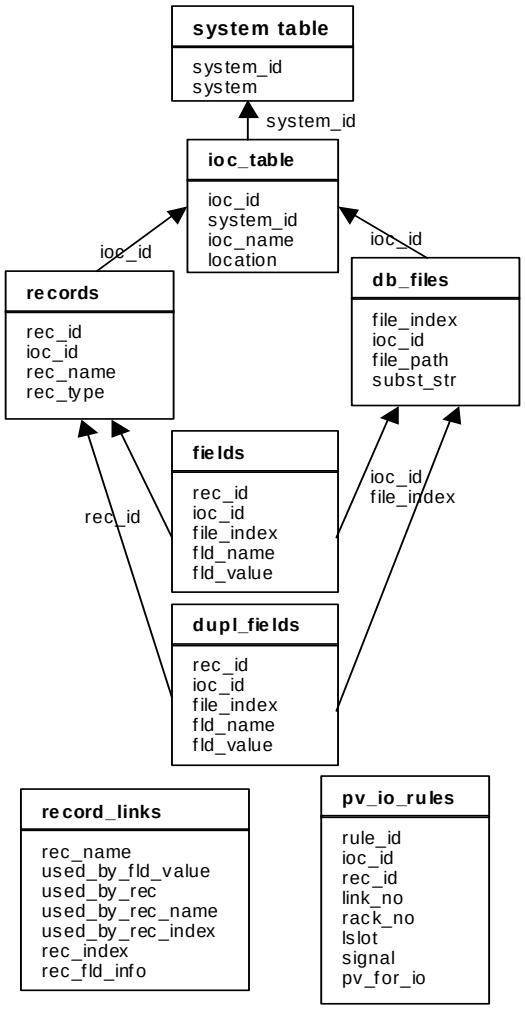


Figure 1: EPICS PV entity relation diagram.

In this schema, the EPICS record and field names are treated as data rather than as attributes. Each of the fields defined in an EPICS db source file becomes a row in the fields table. In the APS control system, there are ~7,000,000 user defined fields. The foreign key for each entry in the fields table references a specific row in the records table. Each entry in the records table has a foreign key to a specific IOC. To assist in database drill down and access, the IOCs are grouped into systems organized by accelerator or by function.

3.2 Database Loading

The file descriptor of each IOC's startup command file is stored in its boot prom. At the APS, the standard boot procedure is modified such that this file descriptor is written out to an IOC-specific 'bootparams' file whenever an IOC reboots. A Perl script periodically checks the modification dates of these bootparams files to determine if any IOC has rebooted. On detection of a reboot, the startup file is parsed to determine the EPICS database definition and instance files to be loaded into the IOC. The EPICS static database access routines are used to extract the record and field definitions from the db files. The script writes this information, along with the respective db file paths in a set of flat files. The script determines if any fields have been multiply defined, and records this information along with their parent source filespecs. Although duplicate PVs are legal and ignored by the IOC, their usage presents a potential trap, which may be otherwise difficult to locate.

An ORACLE server was set up in a stand-alone workstation to contain the EPICS data. A periodic task in this server monitors the directories of the Perl extraction files, and triggers a loader process whenever new flat files have been created. The ORACLE tables are typically updated with the new IOC information within ~15 min after an IOC reboot. As a check of the validity of the ORACLE data, a nightly task is run in which flat files are generated from the ORACLE tables and compared with original flat files. This round trip file compare provides an independent check of the validity of the data.

3.3 User Queries – the pv_search Application

EPICS database inspection capabilities are provided using the standard ORACLE three-tier application server architecture. User requests, entered in a Web browser, are submitted to an ORACLE application server, which formats an SQL query and forwards it to the ORACLE server. Users enter optional IOC, db file, record name, record type, field name, and field type selection criteria. The user then selects one of the records returned by the query to obtain its field detail. In the detail display, EPICS link-type fields are displayed as hypertext links. Selecting a link field will display the details of the linked record.

The record_links table in Figure 1 is derived from the PV database and contains all instances of inter-record links in the entire set of IOCs. This derived table is used to extract the records linked to the actively displayed record. This list is displayed as hypertext links, allowing the user to extract the detail of the linking record. This powerful feature allows the user to bidirectionally explore EPICS logic.

3.4 MEDM Clients

At the APS, the primary operator interface to the accelerator is through the use of MEDM display files. These display files are stored in a single directory structure, under CVS control. A Perl script was written to examine all the files in this directory tree and to recursively extract references to EPICS process variables. When an EPICS record is displayed using the pv_search application, a list of all client MEDM screens that reference any field in the record is displayed. This identifies to the developer the display files that will need to be modified if, for example, the record name is changed.

3.5 Decision Support - Data Mining

The powerful querying capability of SQL has been used to 'data mine' the process variable database to provide a number of decision support queries.

Orphan records. This query extracts all records that:

- have no other PVs linked to it
- do not link to other PVs
- do not connect to hardware
- have no client application connection (MEDM, SDDS logger, etc.)

Records matching this category are potentially 'orphan' records, possibly left over from a prior or test subsystem. Since they probably provide no useful function, they are candidates for removal from the IOC.

Multiply-defined hardware output records. In the EPICS environment, it is legal to define more than one record for a single hardware I/O point. Although this feature is occasionally used as a convenience function, it can lead to equipment behavior problems if conflicting logic attempts to write to a single output channel. Queries have been written to extract and list instances of multiple hardware output records.

Equipment database. For EPICS records that are connected to external hardware I/O, the DTYP fields provide information related to the hardware device. This feature has been used to query the PV database to extract a list of hardware devices in the APS control system. The query report yields a total of 5800 individual devices directly associated with the APS control system. This equipment inventory is used in assisting a hardware spares strategy.

Wiring List. A project has been initiated to centralize wiring list documentation, using the Allen-Bradley equipment as a prototype. Allen-Bradley link, chassis, rack, slot, and terminal relational tables have been defined. A Web-based application allows developers to modify and update the ORACLE tables from any location that has access to a Web browser. Chassis and I/O card templates facilitate data entry. The derived pv_io_rules table shown in Figure 1 connects the wiring information with the process variable database. Display of the wiring details of an I/O card also lists the process variable connected to each terminal. This includes multiple PVs assigned to a single I/O point discussed above.

4 DISCUSSION

Complex and extensive mature systems like the APS control system have developed and evolved over years of field testing. Documentation of the rationale for coding decisions is often difficult to locate, or is incomplete or missing entirely. The control source code is often the only source of this documentation. A centralized control software inspection capability has proven to be useful in this effort. The PV-centric ORACLE relational database along with the querying power of SQL has provided intuitive mechanisms for locating relevant and related hardware and code documentation.

Work in progress includes extending the wiring database to include all the APS control hardware. This equipment 'connection' database will not only catalog all the hardware modules and ancillary equipment but will also indicate where each component is connected to the control system. It will be fully integrated with the PV database to allow cross queries between hardware and software.

Work is in progress to extend the PV clients to other types, including SNL, alarm handler, and SaveCompareRestore. The Perl extraction scripts are being expanded to parse the source files (db and startup files) for code documentation at the record, db file, and IOC level. The ultimate goal is to develop a central mechanism for creating and retrieving control system documentation.

REFERENCES

[1] http://www.aps.anl.gov/epics has extensive information on all parts of EPICS.

[2] W. P. McDowell et al., "Status and Design of the Advanced Photon Source Control System," Proc. of the 1993 Particle Accelerator Conference, Washington, DC, May 1993, pp. 1960-62, 1993.

13] K. Evans Jr., "An Overview of MEDM," Proc. of ICALEPCS '99, Trieste, Italy, October 4–8, 1999, pp. 466-468, 2000.

WEDT002
physics/0111088

THE KLOE/DAΦNE STATUS LOGGING, ANALYSIS AND DATABASE SYSTEM

G. Mazzitelli, F. Murtas, P. Valente
Laboratori Nazionali di Frascati - INFN, Frascati (Roma), I-00044, Italy

Abstract

The KLOE experiment [1] at the Frascati ϕ-factory DAΦNE [2], designed to measure $\Re(\varepsilon'/\varepsilon)$, began preliminary data taking in the Fall of 1999. A large database structure, which logs information coming from the DAΦNE control system and the KLOE slow control and data acquisition systems, has been developed. Data from detector monitoring, online event processing and machine operating conditions are easily accessible for online and offline analysis by means of Web tools and histogramming tools. The system allows powerful real-time data correlations which are necessary for the ongoing program of luminosity and background improvements. Data flow and handling processes are presented.

1 DAΦNE CONTROL SYSTEM

Data in the DAΦNE collider are stored by the control system [3] in the local memories of the 45 front-end VME-CPU's distributed all over the accelerator area. The front-end tasks get commands from the high-level user environment and continually update their own database with information from the devices. Data are available through direct memory access to the CPU's memory. This front-end database is constituted by different data types tailored to specific machine elements (non-homogenous database); this means that in order to use this data or to correlate parameters of different devices, specific routines must be implemented.

Two system tasks have been developed in order to collect all the parameters (Dumper), to synchronize and align them, and then to store them to disk (Storer) [4]. The Dumper continuously fetches data from the front-end memories and writes the different data-types from each machine element aligning them in a homogenous database. This memory resident database is accessible from high-level tasks:

- for monitoring purposes, such as the watchdog process checking for faulty magnets, bad vacuum or CPU failures;
- for the user interface display (from any operator console);
- for online correlation and analysis (as described in Sect. 3).

The Storer then synchronizes the memory database and writes it on the mass storage at given time intervals.

In addition, the Dumper continously reads the status of the satellite systems not belonging to the control system environment (such as the spectrum analyzer), merging this data in the homogenous database. In the general scheme of DAΦNE controls, sketched in the right part of Fig. 1, the main processes and data flow are reported. In addition to the control and monitoring functions, and the communication with the KLOE slow control, an additional process serves DAΦNE data to the controls of the synchrotron radiation facility through the UDP protocol (UDP server). The handling of data relative to the DEAR experiment (running at the second interaction region) is not shown in the figure, since it is monitored as any other part of the machine and the data can be presented through the DAΦNE interface.

2 KLOE SLOW CONTROL

The KLOE slow control system [5] is intended not only for the control and monitoring of the low and high voltages used for the detector (more than 70 DAQ crates, 24 HV crates), but also for the monitoring of the trigger and background rates. The hardware is entirely based on the VME standard. Several VME serial interface boards control the HV and LV settings and the power supplies through low-level processes running on a standard KLOE level-2 VME CPU.

Different low-level processes (one for each part of the detector) implement the low-level VME functions. All the data coming from the monitoring are stored to memory and are handled by high-level processes running on a remote machine, which also runs the processes providing the user interface, generating and handling alarm conditions and communicating with satellite systems. The drift chamber gas control and the superconducting magnet system are also monitored by dedicated high level processes, communicating on TCP/IP sockets with the remote controls. Finally, the high-level also communicates with the KLOE run control, both for the logging of the relevant detector parameters, and for the setting of HV and LV at run start.

The user interface is *entirely* realized using HTML language, so that the monitoring and control functions are implemented by CGI (Common Gateway Interface) programs running on a dedicated Web server, which also displays the anomalous conditions and handles the alarms. The alarm conditions are generated by two different watchdog processes: one running on the low-level VME CPU and accessing the shared memories of the control processes of

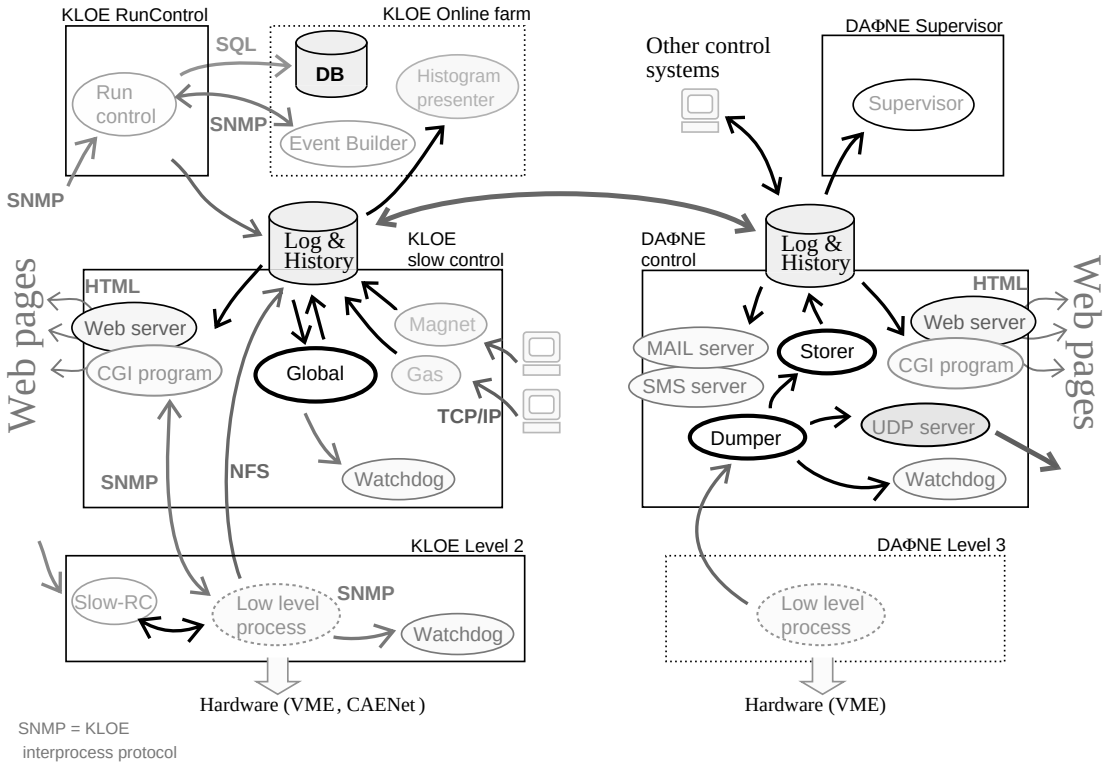


Figure 1: Layout of the KLOE/DAΦNE controls and of the joint logging and database system.

the subdetectors (via SNMP, the KLOE inter-process communication protocol [6]); the other checking the high-level programs on the main machine, and sending the main alarm conditions via the GSM short message service and e-mail.

All the slow control processes write the monitor parameters to a general 'run condition' mass storage. The same area is used by DAQ monitoring processes, running on the KLOE online farm, which perform a fast event reconstruction of acquired data [6]. These monitoring processes produce the relevant quantities from online analysis, such as luminosity, beam position, and background level.

The general scheme of the KLOE controls, with the interactions of the different low and high-level processes, occupies the left part of Fig. 1.

3 KLOE/DAΦNE COMMUNICATION AND DATA INTEGRATION

The DAΦNE and KLOE mass storages, shared between the two control systems, are used by many monitoring processes to log a number of parameters, in general at very different time intervals: 3 seconds for the scalers counting the hits in the endcap and quadrupole calorimeters, 15 seconds for 'fast' variables such as current and roundness of the beams, or the status of the low and high voltages, 1 minute for the 'slow' variables such as the KLOE magnetic field, 5 minutes for the machine vacuum and orbit; or even at non-constant time intervals for the physics quantities, such as luminosity and beam position and momentum measurements, needing some data to be acquired and analysed by the DAQ monitoring processes.

Thus two collaborating tasks, continously running on the two main machines, merge the information coming from the various parts of the two systems and synchronize them:

- the Dumper/Storer on the DAΦNE high-level machine, reading and aligning the non-homogenous data from the distributed VME memories and writing them to the DAΦNE homogenous database;
- the Global collector process (see Fig. 1) on the KLOE slow control machine, reading data from the shared memories of low-level processes (via the SNMP interprocess protocol), stored data from the high-level processes monitoring satellite systems, and the physics quantities from the online event reconstruction.

Some elaboration is also performed starting from low-level data, producing background estimators from spurious hits in the detector, beam lifetimes from the fit of the history of the machine currents, etc.

Finally, the common global database is produced, in which all machine parameters are correlated with the detector quantities. Since there is a wide range of time variations of monitored parameters and of logging time intervals from the various processes (producing or retrieving data), all the available quantities in the final common database are synchronized and stored either in a 'fast file' (at 15 seconds intervals) or in a 'slow file' (at 1 minute intervals).

The global database is then accessed by both the DAΦNE

and KLOE controls Web servers to display a rich wealth of information:

- as already described in the previous sections, the online status;
- the long-term history of the machine and detector conditions;
- a number of statistics of the most relevant quantities.

A number of pages displaying the status and the history of the various parameters and their correlations are made available and extensively used in both control rooms: mostly oriented to the luminosity/machine background optimization in the DAΦNE case, and to keep under control the detector performance and the data quality for KLOE.

In order to help the two teams of physicists in the continous improvement and optimization of the machine/detector operation, the presentation of the monitored quantities and of their correlations is fundamental. This has been realized with two complementary tools taking advantage of the common database: a general-purpose Web interface, running on the DAΦNE Web server, mainly oriented to the display of correlations between different machine and detector quantities, and a histogramming application based on the ROOT libraries (from the CERN program library), running on the KLOE online machines [7], mainly oriented to the presentation of the time charts of any machine or detector parameter. Both the presenting tools access the common database described above, and can display the last few hours status or build the history on a many days base.

An asynchronous builder process continously runs on the DAΦNE supervisor machine, reading the common global database: on the basis of the stored parameters a number of statistics of the machine and detector performance are elaborated, such as delivered luminosity, beam lifetimes, data-taking efficiency, etc. The last-hours, daily or longer term statistics of the most relevant indicators are then made available through the Web interface.

As described above, the supervisor machine also runs a watchdog process checking the most relevant parameters and alarm conditions. A dedicated server takes care of signalling anomalous conditions through the e-mail and GSM short message service, and in addition the daily statistics can be broadcast to the authorized personnel. This can be done in push/pull mode: the operators of the two teams can get the desired information on-demand, or wait for the regular updates.

For safety reasons the KLOE slow control also implements an independent watchdog process, signalling the detector experts the main alarm conditions via e-mail and GSM short messages.

4 CONCLUSIONS

The DAΦNE and KLOE control systems manage and monitor the machine and detector elements through a set of low-level programs. The high-levels of the two systems are strongly connected and integrated: the information coming at different times from the detector and machine controls, from the external control systems, and from the online event reconstruction on the KLOE farm are collected, synchronized, stored, pre-analyzed and displayed from a number of tools, based on a common general database accessed mainly by the two supervisor Web sites and by dedicated programs. Duplication is very limited, since the same low-level parameters are elaborated and shown with different approaches, oriented either to machine optimization, or to detector stability and data quality control.

5 ACKNOWLEDGMENTS

We are grateful to M. Masciarelli for the work on the DAΦNE side, to A. Balla and G. Corradi for the work on the KLOE side of the control system.

6 REFERENCES

[1] F. Bossi for the KLOE Collaboration, *"KLOE Results"*, XX International Symposium on Lepton and Photon Interactions at High Energies, Rome, Italy (2001).

[2] M. Preger for the DAΦNE Team, *"Status of DAΦNE"*, HEACC 2001, Tsukuba, Japan (2001).

[3] G. Di Pirro, G. Mazzitelli, I. Sfiligoi, A. Stecchi, *"The Evolution of the DAΦNE Control System: A History of Liberation from Hardware"*, ICALEPCS 2001, San Jose (CA), USA (2001).

[4] G. Di Pirro, G. Mazzitelli, A. Stecchi, *"Data Handling Tools at DAΦNE"*, EPAC 2000, Wien, Austria (2000), LNF-00/021 (P).

[5] A. Mastrogiacomo, F. Murtas, P. Valente, *"The KLOE Slow Control System"*, KLOE memo 206 (unpublished), 2000.

[6] M. Adinolfi *et al.*, *"The KLOE DAQ system"*, Submitted to Nucl. Instrum. Meth. A.

[7] M. Martemianov, F. Murtas, P. Valente, *"The KLOE Online Presenter"*, KLOE memo 236 (unpublished), 2001.

WEDT003
physics/0111136

INTRODUCING I/O CHANNELS INTO THE DEVICE DATABASE OPENS NEW POTENTIALITIES FOR CONFIGURATION MANAGEMENT *

T. Birke, B. Franksen, R. Lange, P. Laux, R. Müller, BESSY, Berlin, Germany

Abstract

The reference RDBMS for BESSY II has been set up with a device oriented data model. This has proven adequate for e.g. template based RTDB generation, modelling etc. But since assigned I/O channels have been stored outside the database (a) numerous specific conditions had to be maintained within the scripts generating configuration files and (b) several generic applications could not be set up automatically by scripts. In a larger re-design effort the I/O channels are introduced into the RDBMS. That modification allows generation of a larger set of RTDBs, map specific conditions into database relations and maintain application configurations by relatively simple extraction scripts.

1 INTRODUCTION

Within the last few years *Relational DataBase Management Systems* (RDBMS or DB in short) have become essential for control system configuration. With availability of generic applications and hardware standards the interplay of components as well as the adaptation to site specific needs has become crucial. Proper networking has been the central problem of the past. Today's challenge is a central repository of reference and configuration data as well as an appropriate standard suite of DB applications. Target is a management system providing consistency in a programmable, comprehensible and automatic way for development, test and production phases of the control system. Instead of careful bookkeeping of innumerable hand-edited files the ever needed modifications of the facility require 'only' change of atomic and unique configuration data in the DB and eventually adaptation of structures in the DB (applications). The update of tool configurations is then consistently accomplished by direct DB connection, renewal of snapshot files generated by extraction scripts etc.

2 STATUS AND ACHIEVEMENTS

Very early a device oriented approach has been chosen for the description of the 3rd generation light source BESSY II. A naming convention was developed for easy identification and parsing of classifying properties like installation location, device family, type and instance. Around the 'bootstrap' information contained in the device names a first reference database has been set up [1], describing wiring, calibrations, geometries etc.

2.1 Device I/O

Utilisation of the EPICS control system toolkit implies the network protocol called *Channel Access* for the I/O of process variables. At BESSY the device model results in a scheme `<DEVICENAME>:<channel>`.

Generation of the EPICS *Real Time DataBases* (RTDB) for device classes with high multiplicity and simple I/O (power supplies, vacuum system, timings) is based on two components: device class specific data are stored in the DB. Functionality and logic of `<channel>` is modelled with a graphical editor and stored in a template file. Generating scripts merge both into the actual RTDB.

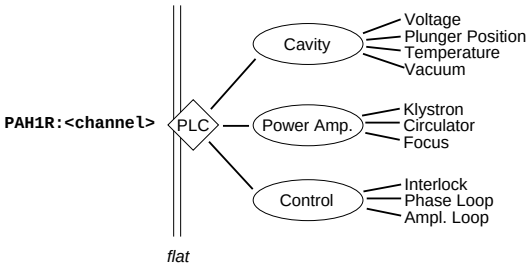


Figure 1: RF Power Units are controlled by flat CAN I/O to complex, turn-key PLC systems. Addressing of sub-units is done via channel (name) grouping.

For unique and complex systems (RF, insertion devices) structuring takes place in `<channel>` where no common naming convention has been defined so far. Contrary to the previous approach of complex template/atomic substitution here all channels involved are assigned to sub-units and hierarchies. The structure is fully implemented in the DB and for RTDB generation only connected with simple atomic templates.

Intermediate systems (scraper, GP-IB devices) are either adapted to one of these approaches or simply set up by ad-hoc created files.

2.2 Conditioning

Documentation of and transitions between different facility operation conditions are handled by a save/restore/compare

* Funded by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) and the Land Berlin.

tool that is working on a set of snapshot files. Coarse configuration is provided by SQL retrievals of name lists. Hierarchies and partitioning are derived from device-name patterns and mapped into directories and files. This is feasible only because the relevant channels are restricted to setpoint, readback and status. Deviations of this scheme are few and easily maintainable by hand.

Information required by modelling tools for conversion between engineering units of device I/O and the physics views (magnet function, length, position, conversion factor etc.) are fully available in the DB [2]. Therefore the linear optics correction tools (orbit, tune) are instantaneously consistently configured provided the installed hardware matches the entries in the DB.

2.3 *Alarm, Archiving*

As long as the alarm handler has to monitor only hardware trips DB retrieval of device name collections and script based interpretation of device name patterns are helpful at least for the simple devices with high multiplicity: typical channels are ON/OFF status. Again grouping and hierarchies can be derived from the class description embedded in the device name.

For the complex devices (RF) control functionality and logic has already been mapped into DB structures (see 2.1). Here genuine DB calls produce alarm handler configuration files with sophisticated error reporting capabilities.

In a similar fashion creation of configuration files for the data collector engine(s) of the archiving system is simplified by DB calls: for each device class a limited number of signals and associated frequencies are of interest for long term monitoring. The DB basically serves as source for device name collections.

2.4 *Data Transfer*

High level software using the CDEV API (configured with a *device description language* file) and 'foreign' networks (connected by the *Channel Access gateway*) benefit also from the DB: For CDEV access definition and permission to selected I/O channels for each device class is defined by appropriate prototype descriptions (in analogy to the RTDB templates). The associated lists of devices are compiled with DB calls. *CA* gateway takes advantage of the naming convention by regular expression evaluation.

2.5 *Installation Procedures*

Facility modifications typically change the device inventory. The DB supports consistent propagation of innovations [3]: During an installation campaign devices are added or deleted in the DB. In addition, new device classes are modelled within script logics and template (prototype) descriptions. Running the configuration scripts on all control system levels updates the configurations within development environment, test system and production area.

3 SOLVABLE DEFICIENCIES

A number of restrictions imposed by the present device oriented DB model are solvable by minor structural modifications and consequent introduction of channels necessary for the adequate description of essential device properties.

3.1 *Device Ambiguities, Hierarchies*

Dependent on the specific point of view, definition of device classes and assignment of equipment comprises ambiguities. E.g. the device class *magnet* (M) is tightly bound to the model aspect of the storage ring, whereas the the class *power supply* (P) covers the engineering aspects of the current converters. Pulsed elements (Kicker, Septa) are very similar devices, but do not fit into this scheme - neither into the model nor the I/O aspects. The decision to assign a single dedicated device-class (K) introduces a new pattern (fig. 2) and breaks the naming structure.

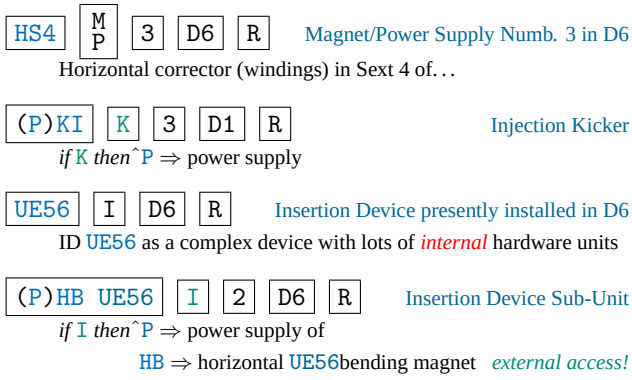


Figure 2: Ambiguities in device class definition result in a complicated naming structure

Difficulties get worse for complex pieces of equipment: insertion devices have to be treated as 'units'. They can be completely replaced by different entities with similar complexity. Typically they consist of a number of devices partly belonging to already described device classes (like bipolar power supplies).

The straightforward solution here is the extension of the device naming convention. Substructures and naming rules have to be introduced for the (up to now monolithic) device property describing name element ('genome chart'). Corresponding DB structures have to be implemented.

Similarly naming rules have to be developed where structuring takes place in <channel> (fig. 1). In summary, the distinction and boundary between device and <channel> are dictated by specifics of the I/O connectivity and arbitrary with respect to the required DB structure. It does not necessarily coincide with the most adequate classification aspects.

3.2 Context Dependent Role of Devices

Several global states are well defined for the whole facility (e.g. 'shutdown', 'machine development', 'user service' etc.) or for sub-sections or device collections: 'injection running', 'wave length shifter ON' etc. The role of devices depend on these states: in the context of 'shutdown' insufficient liquid helium level at super-conducting devices has to generate a high severity alarm. For power supplies even OFF states are then no failure. But during 'user service' alarms should notify the operator already when power supply readbacks are out of meaningful bounds.

Similar case distinctions have to be made for all conditioning applications (reload constraints for save/restore, active/inactive elements for modelling), for data archiving (active periods, frequency/monitor mode), for sequencing programs etc. Today only the most general conditions are statically configured and available from the database/configuration scripts. Exceptions are partly coded within the applications or have to be handled by the operators — an error-prone situation.

A clearly laid out man-machine interface provides an effective protection against accidental maloperation. Presentation details and accessibility of devices should be tailored to the user groups addressed: operators, device experts, accelerator physicists. The required attributes attached to the device channels should be easily retrievable from the DB.

4 STRUCTURAL SHORTCOMINGS

4.1 Immature Data Model

Starting from the device model a very specific view of the control system structure has been mapped into DB structure: device classes correspond to tailored set of tables. Deviating aspects that do not fit into the scheme are modelled by relations, constraints, triggers and presented to the applications by pre-built views. The result is not very flexible, hard to extend and complicated to maintain in a consistent state free from redundancies. Numerous device attributes are scattered over template/prototype files and configuration creating scripts in an implicit format without clear and explicit relation to the data within the DB.

4.2 Unavailable Data

Configurations of archiver retrieval tools are much more demanding than those of data collector processes. In search of correlations arbitrary channel names have to be detected by their physical dimension: for a drift analysis the data sources for temperatures [C], RF frequency [MHz], BPM deviations [mm] and corrector kicks [mrad] have to be discovered. Any non-static, not pre-configured retrieval interface to the most important performance analysis tool of the facility –archived data– requires clues to signal names, meaning, functionality, dimensions, and attributes.

With the RTDB template approach the channel attributes are hidden within these files and not available to the DB. Consequently, no DB based data relations are available for correlations (comparable dimensions), projections (user, expert, device responsible) or dependencies (triggered by). There is no browsable common data source that allows coordination of consistency of configurations requiring channel attributes (that go beyond simple and clear assignments to well defined device classes). The device model is tailored for modelling applications. For the archiver retrieval it is useless continuously causing errors and inconsistencies.

5 SOLUTION ATTEMPT

In a kind of clean-up effort the existing fragments used for RTDB generation will be put into a new scheme. Sacrificing the feasibility to generate complex RTDB templates with a graphical editor, now device class tables, templates and I/O software modules (*records*) will be put together as a new DB core. In view of the extendibility for high level applications a purely DB based approach has been taken — even though for RTDB generation the framework of XML seems to be an attractive and well suited alternative.

A new entity *gadget* is introduced connecting the data spaces *Device*, *Channel* and *IOstruct*. These DB sections have to be set up consequently in 3rd normal form, i.e. all device class properties, channel attributes and I/O specifics have to be modelled in data and relations, not in table structures. In blue-print the resulting data model looks abstract and (intimidatingly) complex, but promising with respect to DB flexibility, extendibility and cleanness. Expectation that today's complicated configuration scripts collapse to simple sequences of DB queries seems to be justified.

6 SUMMARY

The most difficult problem of a comprehensive configuration management system is the determination of 'natural' data structures and breaking them down into localised data areas, dependencies and relations. For a relatively small installation like BESSY II the expected reward in stability and configuration consistency does not clearly justify effort and risk of the envisaged re-engineering task. One could argue that the present mixture of DB based and hand edited configuration is an efficient compromise. On the other hand the persistent and boring work-load due to manual configuration update requirements is a constant source of motivation to pursue this reengineering project.

7 REFERENCES

[1] R. Bakker, T. Birke, B. Kuske, B. Martin, R. Müller, Proc. of the 6th Int. Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS97), Beijing, 1997.

[2] R. Bakker, T. Birke, B. Kuske, R. Müller, Proc. of the Automated Beam Steering and Shaping Workshop, CERN, CERN 99-07, p. 97, 1998.

[3] R. Müller, R.J. Bakker, T. Birke, R. Lange, Proc. of the 7th Int. Conf on Accelerator and Large Experimental Physics Control Systems (ICALEPCS99), Trieste, p. 326, 1999.

CONFIGURATION AND DATABASE FOR THE CONTROL SYSTEM OF THE VISIR INSTRUMENT

J.F. Gournay, Y. Lussignol, CEA-Saclay, DAPNIA

Abstract

A French-Dutch consortium of institutes led by CEA/Saclay is designing and building VISIR [1] (VLT Imager and Spectrometer for the mid-InfraRed), a cryogenic multimode instrument for ground-based astronomy at wavelengths of 10 and 20 μm. It will be mounted in 2002 at the focus of the third 8-meter telescope of the ESO VLT observatory in Chile.

In order to control the different parts of the instrument, to interface the telescope and to perform observations, a suitable control system is being developed. It must adhere to common solutions, which are used for all the VLT instruments. The VLT software [2] makes extensive use of configuration files, parameter files and data dictionaries to configure the instrument database, to perform observation and to store astronomical data.

The overall layout of the control system will be described, the use of the VLT software will be discussed and a particular emphasis will be put on the benefit of the extensive use of a well-defined and structured environment for configuration and data representation.

1 INTRODUCTION

VISIR is an imaging and spectrometric instrument at the focus of one of the telescopes of the VLT built by ESO on Mt Paranal in Chile. VISIR will collect infrared radiation essentially produced by dust in various environments, such as circumstellar disks, a domain seldom studied so far. These disks can harbor newly formed planets. Another aspect covered by VISIR is the phenomena related to stars formation. VISIR is composed of an imager and a spectrometer operating in the thermal infrared. Photons are detected by a photo-conducting array of 256x256 pixels cooled down at cryogenic temperatures. Wavelength selection is performed with filters. For the spectrometer part of the instrument, dispersion is achieved with a grating. One key element in the VISIR design is an innovative type of cryomechanism, which is able to actuate optical devices with a very high accuracy and positioning repeatability. The instrument will be installed in 2002. Fig. 1 shows the outer part of the cryostat, its diameter is 1.20m and the total weight of the instrument will be 1000 kg.

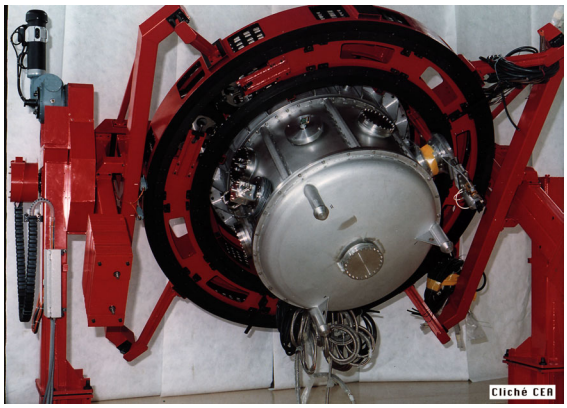


Fig. 1 VISIR mounted on the integration support

For the control of the instruments, ESO provides a complete precisely defined solution to all institutes and sub-contractors responsible of the development of an instrument. This approach is completely justified: the VLT observatory will consist of 12 instruments for the 4 telescopes plus 3 others for the VLT interferometer program. All these instruments come from various European laboratories and will be managed locally in Chile by a limited number of ESO people. Then, it is mandatory for the local team to deal with hardware and software standard solutions for day-to-day operation, maintenance and possible repairs.

The different hardware elements to build a control system for an instrument are the following:

- A HP workstation running UNIX (WS).
- A VME crate with a 68K or a Power PC CPU running vxWorks (LCU).
- An Ethernet network linking the workstation and the LCU.
- A set of 5 VME modules to achieve motor controls, analog IOs, binary IOs, timing synchronization and serial interfaces to other devices.
- A set of specialized modules to perform detector read-outs.

Software is also completely defined; the VLT Common Software consists of layers of software in the

UNIX and in the vxWorks environments. On the UNIX side, this software is based on RTAP provided by HP. This package consists mainly of a database system and an interprocess communication layer. A compatible database and communication protocol was developed by ESO in the vxWorks environment.

The infrastructure to perform the control and acquisition of VISIR consists of one WS, two LCUs and a front-end acquisition system. The first VME is used to interface the imager and the star simulator (an assembly of devices to simulate stars with standard sources). The second one is used to interface the spectrometer. These two LCUs with the associated electronics will be mounted on each side of the cryostat. Two non-standard VME modules are used for the specific needs of the gratings control: a 16 bit DAC module and a 16 bit LVDT readout module.

2 DATABASE

2.1 Overview

The VLT common Software relies on a real-time database distributed on the WS and the LCUs. This database system supports an object-oriented organization: the points are the true objects describing the system and, in general, are instances of classes. The database includes also some other object-oriented concepts like class inheritance and overloading. The points contain a list of attributes, which contain the data.

Each database used in the control system (one for each LCU and one for the WS), is defined by ASCII files: different files are used to define the classes and the point instances. These files look like C++ include files and a special tool is used to generate loadable modules from them. Unfortunately no tool is provided to build easily a database. Furthermore, the process of generating a database is slow: it takes 1 minute for 50 points on a rather powerful workstation (HP 9000/785 with 1GB RAM). So another policy was adopted to easily generate the big complex motor databases.

2.2 Structure and configuration of the motor database

Each instrument contains several motors of various types: VISIR has 13 stepper motors with limit and reference switches. They are used to rotate filter wheels, to introduce diaphragms or lenses, to move tables, etc. The software handling of these motors is quite sophisticated: the database definition of one motor is based on a generic class. This class has more than 100 different attributes of various types and lengths. To handle efficiently all these parameters, a dedicated ensemble of panels is used to initialize the motor configuration. When the LCU is booted, a skeleton database instantiating a motor of the proper type is loaded. Then, the configuration previously defined by the panels is loaded. It is an elegant way to hide the complexity of the motor database and to save time in avoiding rebuilding again and again the database on the WS.

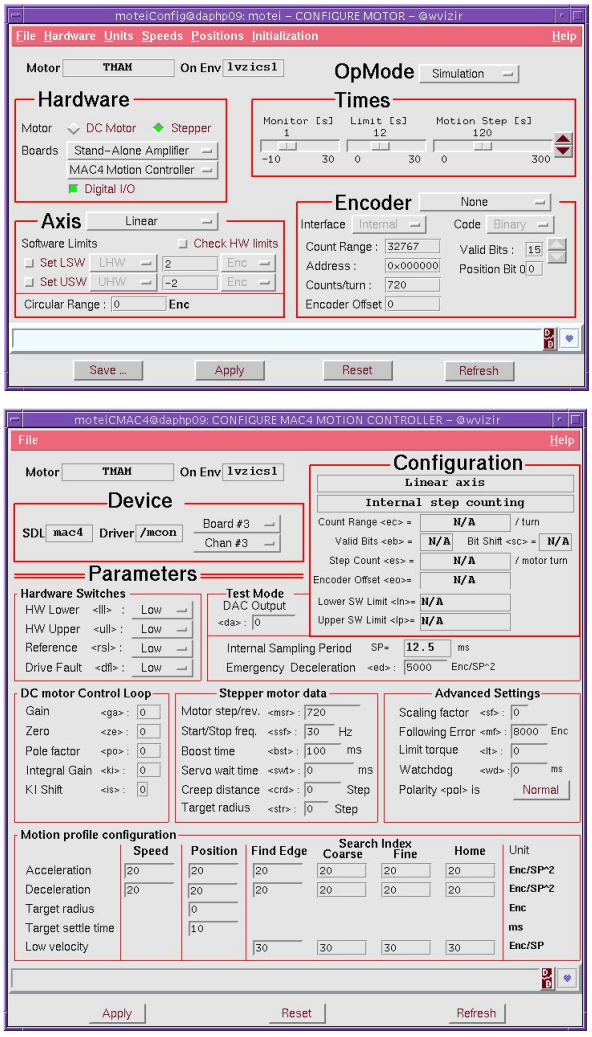


Fig. 2 Panels for motor configuration

3 DATA FLOW SYSTEM

3.1 Overview

In such a facility operated with a reduced team and hosting foreign scientists, well defined and coherent parameters and data specifications are fundamental. To achieve this goal, a complete data flow system starting from the observation preparation and ending in the astronomical data and log files are provided by ESO. Practically, data dictionaries are used in order to define all parameters used in a given context, e.g.. to control the instrument or to interface the telescope or to perform an acquisition.

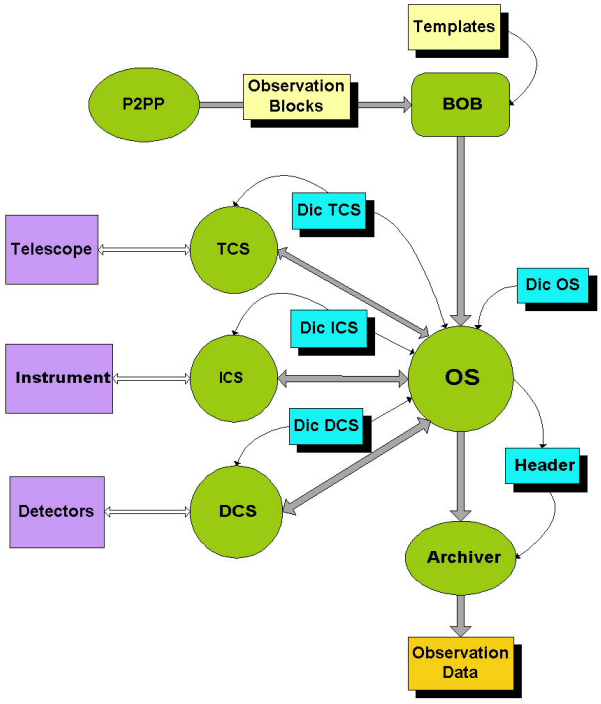


Fig.3 Software Architecture

The main software building blocks are shown on Fig. 3. The Observing Software (OS) is the central part of the system. It receives commands from the Sequencer (BOB). These commands are grouped together in so-called observation blocks (OBs) prepared by astronomers with a dedicated tool (P2PP). OS coordinates the exposure in sending commands to the telescope via TCS, to the instrument via ICS and to the detector via DCS. Finally, OS receives astronomical data from DCS and is responsible for archiving these data. For each subsystem a dictionary must be supplied. The dictionaries contain all the keywords, their context, their format, their description etc…which will be used throughout the system. All instruments of the VLT share the telescope dictionary, the detector dictionary and a large part of the instrument dictionary. Finally, the astronomical data are archived with a set of keywords associated with their values in order to identify the data. Then, all operational steps (setting the telescope and the instrument, performing the acquisition, storing the data) are based on these dictionaries giving a great coherence to all the different control and acquisition systems of the VLT. Fig. 4 shows the generic OS software interface, which controls the subsystems and handles exposures. This generic interface will be later enhanced for the specific VISIR needs.

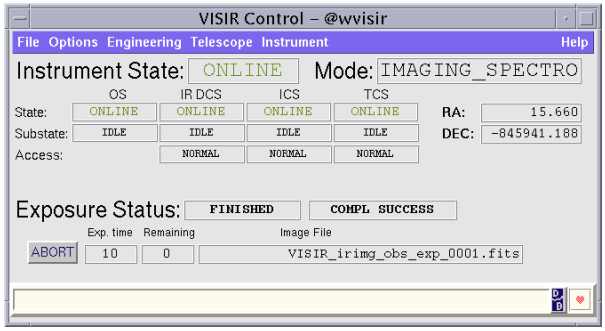


Fig. 4 Generic OS user interface

4 SUMMARY

The VISIR control software is being done with the hardware and software framework provided by ESO for the whole VLT. To overcome the complexity of the database, especially for the mechanisms, tools are provided for dynamic configuration. Dictionaries are used for instrument configuration, for telescope and detector settings and for identification of data. This policy gives a good coherency to all instruments and telescope control systems and eases preparation and performance of observations.

REFERENCES

[1] P.O. Lagage - The final design of VISIR, the mid-infrared imager and spectrometer for the VLT - Proc of SPIE vol. 4008 (2000), Optical Telescope Instrumentation and Detectors, Masanori Iye and Alan F. Moorwood Editors, pp 1120-1131.

[2] G. Raffi - Control Software for the ESO VLT - Proc. of ICALEPCS'91, KEK, Tsukuba, pp. 202-207.

WEDT005
physics/0112049

FRONT-END ELECTRONICS CONFIGURATION SYSTEM FOR CMS

P. Gras, CERN, Switzerland, University of Karlsruhe/IEKP, Germany
W. Funk, CERN, Geneva, Switzerland
L. Gross, D. Vintache, IReS, Strasbourg, France
F. Drouhin, UHA, Mulhouse, France

Abstract

The four LHC experiments at CERN have decided to use a commercial SCADA (Supervisory Control And Data Acquisition) product for the supervision of their DCS (Detector Control System). The selected SCADA, which is used for the CMS DCS, is PVSS II from the company ETM. This SCADA has its own database, which is suitable for storing conventional controls data such as voltages, temperatures and pressures. In addition, calibration data and FE (Front-End) electronics configuration need to be stored. The amount of these data is too large to be stored in the SCADA database [1]. Therefore an external database will be used for managing such data. However, this database should be completely integrated into the SCADA framework, it should be accessible from the SCADA and the SCADA features, e.g. alarming, logging should be benefited from. For prototyping, Oracle 8i was selected as the external database manager. The development of the control system for calibration constants and FE electronics configuration has been done in close collaboration with the CMS tracker group and JCOP (Joint COntrols Project)[1].

1 INTRODUCTION

The supervision of the CMS detector control system (DCS) has to control classical "slow control" items such as power supplies, gas systems, etc. as well as FE (Front End) configuration, which comprises a lot of different parameters.

The supervision level has to provide hardware access, alarming, archiving, scripting, graphical user interface and logging features. In the following the prototype system developed for the tracker of CMS will be described.

[1] The four LHC experiments and the CERN IT/CO group has merged their efforts to build the experiments controls systems and set up the JCOP at the end of December, 1997 for this purpose.

2 DESCRIPTION OF THE FRONT-END CONFIGURATION CONTROL SYSTEM

PVSS II, a commercial SCADA product from the Austrian company ETM was used for the supervision level. Because the amount of data for the FE configuration is quite big, an external database (Oracle 8i) was used for the storage of the FE configuration data itself.

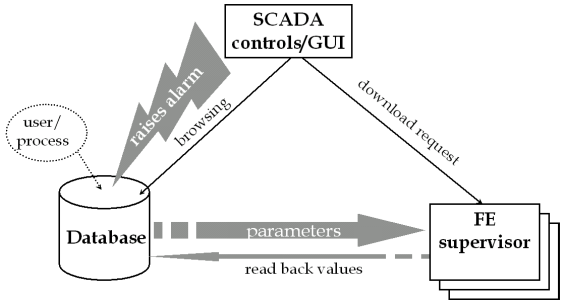


Figure 1: FE configuration mechanism.

2.1. Configuration mechanism

The FE configuration system is comprised of 3 actors as illustrated in Figure 1: the database, the SCADA and FE supervisor(s). The database stores the parameters, the SCADA controls the operation and

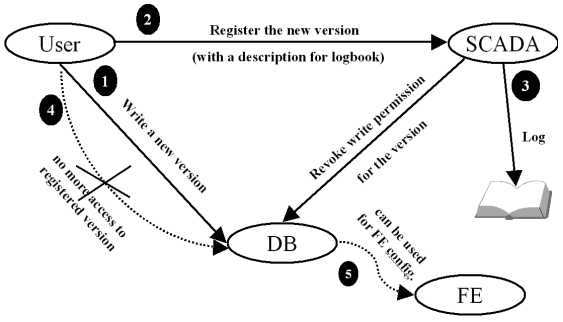


Figure 2: Access control registration mechanism.

provides the user interface, the FE supervisor(s) accesses the FE. The SCADA system as well as the database can be distributed over many PCs on different platforms. In order to transfer the data in parallel to the electronics, the system can have several FE supervisors.

On user request or during an automatic procedure (e.g. start of run, error recovery…) the SCADA sends a download command to the FE supervisors. Then the FE supervisors fetch the data from the database and download them into the FE electronics. It is possible to ask each FE supervisor to read back the configuration from the electronics and send it back to the database. Then, if some values are outside limits, an alarm summarizing the differences will be sent to the SCADA.

2.2. SCADA library and user interface

For the user interface, PVSS II panels have been developed: for version creation, for version registration, for database browsing, etc. These panels can be used as complex widgets.

The database-browsing panel called DBNav is a generic user interface for Oracle 8i databases. It is able to discover the structure of the database and display it in a tree.

These panels are based on two underlying PVSS II script libraries, which can be used directly to develop custom scripts or panels.

2.3. Access control

In order to keep the history of the data used for a configuration, versioning of the stored parameters and a registration mechanism have been developed. FE parameters, and also calibration constants, may be calculated by some process, which is independent of the SCADA. Before a configuration set may be used for a run, it should be registered using the SCADA. At registration time, the SCADA logs the description of the new configuration, and revokes the write permission on the registered configuration set. This write permission revocation is done using the database access control. In this way, all versions used once for configuration will be kept unchanged for later analysis. Figure 2 describes this registration mechanism.

2.4. Database model

Each electronic device type is represented by a table. This table contains the device parameters. In the example of the CMS Tracker described in section 3.2, the database table named APV contains all the parameters of the APV readout chips. A device can be part of a higher-level device (e.g. a chip is part of a board). Such membership relation is specified in the database by a standard relational database "references constraint"[2] between the device and the subsystem. A "controlled by" relationship or any N-to-1 relationship is represented by such a constraint.

The parameter versioning is taken care of by a specific table, typically called "version", which contains the list of all available versions. A version is identified by two numbers: the major and the minor version ids. The version table is composed of at least five columns: one for the major id, one for the minor id, one for the version creation date, one for the description and one which specifies if the version has been registered. Each row of device type tables contains the values of one device for a specific parameter version. The row includes the version ids, which refer ("references constraint") to the version table. Actually if a device contains versioned parameters and version-independent parameters, the device type table can be split into two tables: one for the versioned parameters and one for the version-independent parameters.

Finally, in order to use the alarm mechanism, the device type table must have a "device_type" column which specifies if the row contains the set value, the minimum allowed value or the maximum allowed value.

The "value_type" column and the version table are optional and are not needed if alarming or versioning is not required.

3 FE CONFIGURATION SYSTEM USAGE

3.1. Electronics-specific part

Two parts are specific to the front-end. The first part is the database content: typically each device will have a table, which will contain its parameters. A general database scheme is given as a template.

The other specific part is the "driver". The "driver" is the part that fetches the data from the database and downloads it to the front-end electronics. It receives commands from the SCADA[3]. This part needs to know how to access the specific front-electronics hardware. It can get the data from the database using a standard interface like JDBC, as it has been done for the Tracker (see next part) or with more Oracle-specific interface like OCI or Pro*C or in XML format (provided as standard by Oracle 8i).

[2] In relational database jargon the constraints are the rules that are defined in order to keep the database consistent. Each time the database is altered, the database manager checks that these rules are not violated.

[3] A simple interface to receive this command is provided in C/C++ and in Java.

3.2. Use of the FE configuration system for the CMS tracker readout electronics

The Tracker FE has in its final design about 80,000 APV readout chips of which each has about 20 parameters. Therefore each version of parameters will contain several Mbytes. With the expected number of versions we arrive at the order of GBytes.

The Tracker is organized in modules. Each module has 2 to 6 APVs, 1 PLL[4] chip and 1 channel multiplexer, called an APVMUX.

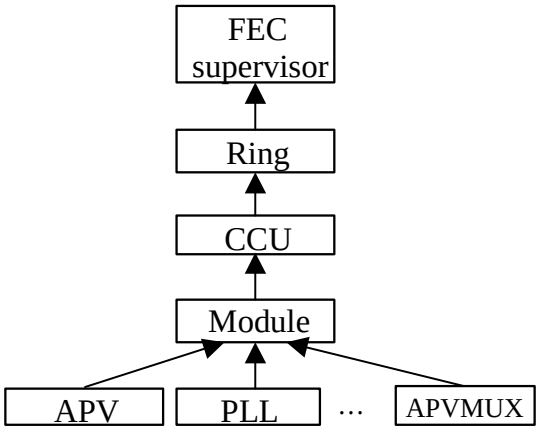


Figure 3: FEC electronics control hierarchy.

Chips, called CCUs, control the APVs. CCUs[5] communicate through a Token Ring controlled by a FEC[6][4] board. In current prototypes the FECs are PCI cards hosted by a PC. The hierarchy of all FE electronics is shown in figure 3.

This hierarchy is reflected in the database through the "references constraints" as described in Section 2.4. For each item in Figure 3 one table is defined in the database.

The setup has been used successfully in the tracker beam test, which took place at CERN in October 2001. Figure 4 shows the FE configuration in the beam test DCS context. During this beam test, PVSS II was also controlling a HV power supply and was monitoring humidity and temperatures on the detector. A PLC was interlocking the high voltage depending on the detector temperature. On interlock the PLC was notifying PVSS II in order to generate an alarm. An electronics logbook using Oracle database with a user interface in PVSS II and a web interface has also been developed for this beam test. Finally a communication between the DAQ and SCADA has been implemented in order to synchronize them.

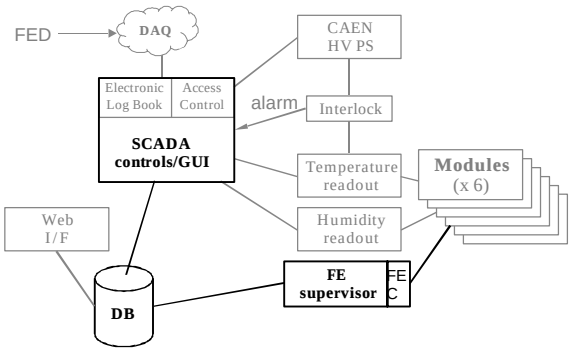


Figure 4: Tracker DCS setup. The FE electronics configuration system is represented in bold. The SCADA communicates with the DAQ, which receives data from the FEDs (Front End Drivers). For simplification the CCU ring between the FEC and the modules is not shown.

4 CONCLUSIONS

The FE electronics configuration system, which has been developed and applied to the CMS Tracker, has been designed in a sufficiently generic manner to be used in other detectors. A prototype using this system for the CMS ECAL detector is under development. During a beam test it has been proven that this system fulfills the principal requirements. In the future some more effort will be needed to measure the performance and to tune it. Tests will also be undertaken to verify the scalability of the system.

ACKNOWLEDGEMENTS

We are deeply indebted towards the CERN Oracle support group for their competent assistance in setting up and running our database.

REFERENCES

[1] F. Drouhin et al., The Control system for the CMS Tracker Front-end, IEEE No. TNS-00028-2000

[2] J. Dirnberger et al., A Prototype of the Control System for the CMS Tracker Front-End, Icaleps'01 TUCT004

[3] Specifications for the Control and Read-out Electronics for the CMS Inner Tracker, A. Machioro, http://www.cern.ch/CMSTrackerControlxx

[4] The Tracker Project Technical Design Report CERN/LHCC 98-6, CMS TDR 5, 15 April 1998

[4] Phase Locked Loop
[5] Communication and Control Unit.
[6] Front-End Controller

Wednesday, 28 November 2001

WEAP - Posters

WEAP002
hep-th/0111071

PORTING OF EPICS TO REAL TIME UNIX, AND USAGE PORTED EPICS FOR FEL AUTOMATION

T.V. Salikova
Budker Institute of Nuclear Physics, 630090, Novosibirsk, RUSSIA

Abstract

This article describes concepts and mechanisms used in porting of EPICS (Experimental Physical and Industrial Control System) codes to platform of operating system UNIX. Without destruction of EPICS architecture, new features of EPICS provides the support for real time operating system LynxOS/x86 and equipment produced by INP (Budker Institute of Nuclear Physics). Application of ported EPICS reduces the cost of software and hardware is used for automation of FEL (Free Electron Laser) complex.

PORTING OF EPICS TO REAL TIME UNIX

The aim of the porting [1] was the reduction of cost of hardware and software required for creation of distributed control system which is built on base of personal computers with processors Intel x86, which has best relation of the cost per performance. INP has the rich experience of development of the special equipment for the particle accelerators, which is used for creation of FEL [2], and CAMAC devices made in INP is applied for automation of FEL. LynxOS/x86 [3] was chosen as low-cost UNIX real time operating system that supports POSIX standards. Besides, free software of GNU and XFree86 projects works at LynxOS platform.

Release 3.13.2 was chosen as robust variant of EPICS [4] for FEL control system, but the porting was started with release 3.13.0.beta12 [5] in 1998. Now I do not use R3.14, which supports Operating System Independent (OSI) features, by two reasons: first, I have not time for the preparation OSI package for LynxOS and testing of it. I am afraid that R3.14 requires several corrections of the device support routines and CAMAC driver. Second, FEL control system would be fully installed by the end of this year.

The porting of EPICS has two stages:

a) support of LynxOS/x86 at OPerator Interface (OPI),
b) libraries emulates VxWorks [6] calls which is used by Input Output Controller (IOC). Writing of CAMAC driver that supports synchronous and asynchronous requests.

The inclusion of LynxOS/x86 support at OPI level is trivial procedure. It required the preparation of configuration files. Channel Access (CA) package of these EPICS versions is built on 4.3BSD socket system, then LynxOS/x86 2.5 and later versions uses 4.4BSD. Only R3.13.3 give opportunity of CA compilation as 4.3BSD and 4.4BSD. It requires little corrections in the routines of CA library working with *ifreq* structure [7].

The general requirement of support LynxOS at IOC level is conservation of EPICS source without modification. For this purpose, the list of VxWorks calls was prepared, list contains all calls used by IOC. On base this list, libraries were created that was fully emulated VxWorks calls. Multitask mechanism of VxWorks is reconstructed by multithreads features of LynxOS easy.

1) Ring buffer subroutines library *rngLib.h* is emulated by the set of routines which manages FIFO circular buffer, which is described *RING* structure such as VxWorks uses. Routines of linked list library *lstLib.h* was fully copied from VxWorks.
2) General semaphore library *semLib.h*, binary semaphore library *semBLib.h*, counting semaphore library *semCLib.h*, mutual-exclusion semaphore library *semMLib.h*. For emulation of different types of semaphores was chosen counting semaphores of LynxOS. All semaphore identifiers is described by *SEMAPHORE* structures that is placed in doubly-linked list. This list is processed by emulated routines as *semDelete*, *semTake*, *semGive*, *semFLush* and other. *SEMAPHORE* structure contains information about type of semaphore, priority, counter, timeout and additional information about threads that works with this semaphore.
3) Task information library *taskInfo.h*, task management library *taskLib.h*. The set of these calls is emulated by POSIX draft 4.9 multithreads features of LynxOS. LynxOS thread is equivalence VxWorks task. All task identifiers *tid* is placed into ring buffer, that is dispatched by special *pthread*. This *pthread* controls state of all spawned tasks.

4) Watchdog timer library *wdLib.h*. Watchdog timer mechanism is emulated by the creation of POSIX interval timer for the calling process [8]. The extended *WDOG_ID* structure describes POSIX timer and his queue.

The problem of symbol table subroutines library is not solved, *iocCore* is compiled with libraries that emulate VxWork features, libraries of record support routines and of CAMAC device support routines fully. The necessary entry points of record support routines and device support routines is enumerated in *src/db/iocInit.c* file. Sequencer does not works of this reason. The IOC structure is shown on Fig. 1.

iocCore has the possibility of launching of X11 subwindow with several IOC test routines: *dbl, dba, dbpr, scanppl, scanpel, scanpiol* (the list of routines will be extended).

On the analogy of VME/VXI, CAMAC, BITBUS and other, the support of CAMAC_PPI bus is added to *src/dbStatic/link.h* and *src/dbStatic/dbStaticLib.c* files. CAMAC_PPI interface is ISA card with six channels for the connection of crate controllers. This IOC supplement serves CAMAC records, which has next capabilities: the maximal number of NAF for record initiation is 5, the maximum of NAF for record processing is 9. The chain of NAF consists of following information: N – the crate number, A – subaddress, F – function, data, cycle counter for waiting Q signal and time interval in ticks for delay in this cycle.

Due to CAMAC_PPI supplement in *src/dbStatic/dbStaticLib.c*, Database Configuration Tool *dct313* supports the preparation of records working with CAMAC devices. *dct313* allows preparation of a block of NAF. CAMAC devices is serviced by the set upward of 20 device support routines that were written for optimization of exchange with equipment.

USAGE PORTED EPICS FOR FEL AUTOMATION

The distributed control system of FEL is built on private subnet providing minimal traffic. All control computers is connected to “Ethernet segment” which attached to the gateway. There are personal computer under supervision of Linux with IP-masquerade. All control computers has the access to Internet, but there is not external access to them.

The high performance Pentium allows one to join OPI and IOC functions in one computer, the average performance is equal to 4000 records per second at Pentium-III 550MHz, which simultaneously services OPI and IOC features of Radio Frequency (RF) control system. Test executes *get* and *put* requests to *ai* and *ao* records in *iocCore*, servicing RF database consisting of 483 records. Those are usual debugging conditions for the adjustment of FEL subsystem.

At present time, FEL complex can be divided into four autonomous subsystem, each having its own IOC:

a) RF system joins three cavities and three generators of injector and sixteen cavities and two generators of microtron. RF IOC services 481 records.
b) Injector IOC controls electron gun and solenoids, lenses of injector and beam transducers. Database contains upward of 150 records.
c) IOC of microtron magnetic system controls about 300 parameters of power supplies of bend magnets, solenoids, quadrupoles and undulators.
d) Diagnostic IOC services diagnostic equipment and checks temperatures in 120 points of microtron.

The adjustment of RF IOC and Injector’s IOC is being finished. Now last two IOC is designed.

OPI programs is compiled with libraries of Graphical User Interface (GUI) which is built on free software: XFree86 4.1.0, OpenMotif 2.1.30, TCL/TK 8.4 tool kits, which supports modern video cards and monitors as well as new features of graphical libraries. The software choice was defined by possibility of upgrade of video resources in future.

OPI programs of FEL control system is built by two means:

a) control panels for operator-physicist is set of subwindows that is spawned by widgets as *menu_pull* at menu bar or *PushButton*. Subwindows contains tables and graphs of input data. Set of *scale*, *SpinBox* and other widgets is used for output data. FR console is created in this way.
b) control panel of program for government of injector (also another subsystem of FEL) has mnemonic diagram of elements of FEL injector. It is similar to automations symbols on panels of industrial automation applications made by National Instruments [9]. Each element on panel is linked with its subwindow. Clicking on a mnemonic element will spawn of subwindow, control panel of this element. Also color palette is used to indicate the alarm status. If device fails, then corresponding element in the mnemonic diagram is painted in red.

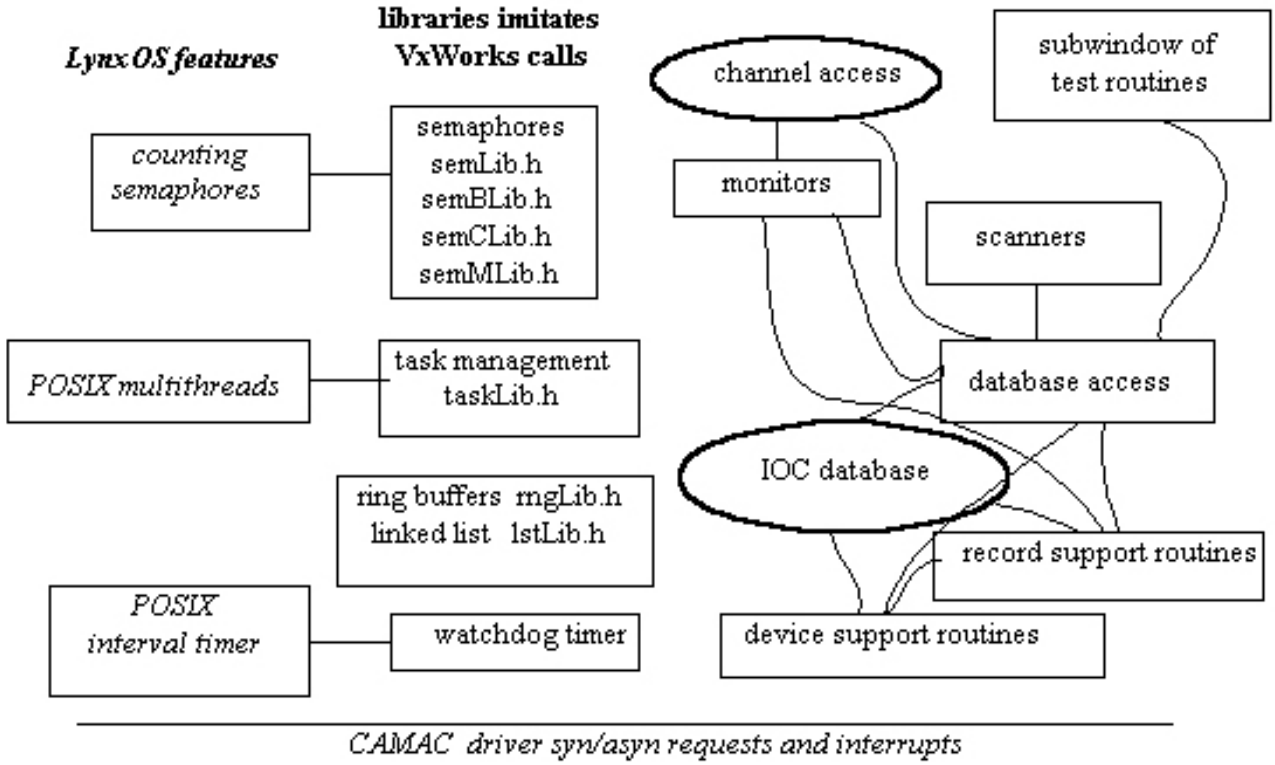


Fig.1 The structure of *iocCore*.

CONCLUSIONS

The EPICS tool kit provides environment for development of the FEL control system in the modern state of computer technologies. The ported version of EPICS allows utilization of hardware made by INP, which reduces cost of hardware used for FEL automation. The average performance of IOC at Pentium III (550 MHz) is equal to 4000 records per second, which is enough to make a good control system.

REFERENCES

[1] A.D. Oreshkov, T.V. Salikova, John Madey, Ying Wu, “Project for developing a low-cost and high performance version of EPICS”, 1998, preprint INP 98-19, Novosibirs, Russia.

[2] N.G. Gavrilov et al. " Status of Novosibirsk high power electron laser project". Free-electron laser challenges. P.G. O'Shea, H.E. Bennett, eds. 13-14 Feb. 1997, San Jose, California. SPIE, 1997. Page 185-187. (Proc. of SPIE -Intern. soc. vol. 2988).

[3] Web site of Lynx http://www.lynuxworks.com

[4] Web site of EPICS collaboration http://www.aps.anl.gov/epics

[5] M.R. Kraimer. “EPICS Input/Output Controller application developer’s guide”. Argonne National Laboratory APS, Jully 1997, release 3.13.0.beta11.

[6] “VxWorks programmer’s Guide”, 1995, Wind River Systems Inc.

[7] G.R. Wright, W.R. Stevens, “TCP/IP Illustrate, Volume 2. The Implementation”, Addison-Wesley, 1995, pp 114-125.

[8] B.O. Gallmeister “POSIX.4: Programming for the Real World”, O’Reilly 1995, pp 463-474

[9] “The measurement and Automation”, catalog 2001, NI, pp 148.

WEAP003
physics/0111024

THE EVOLUTION OF THE DELTA CONTROL SYSTEM

E. Kasel, B. Keil, D. Schirmer, D. Zimoch
Institute for Accelerator Physics and Synchrotron Radiation,
University of Dortmund, Germany

Abstract

Since January 1999 the Dortmunder Elektronen Testspeicherring Anlage (DELTA) control system has changed from a non-standard hand-made system to the standard Experimental Physics and Industrial Control System (EPICS) [1]. The complete 80 MeV LINAC and transfer-line, the asymmetric 5.5 T multipole wiggler magnet (SAW) and main parts of the 1.5 GeV electron storage ring are now under EPICS control.

Additional hardware upgrades like a fast 100baseTX/FX Ethernet network, Linux-PCs, PowerPC-IOCs, and reengineered device I/O-interface cards have been installed. Moreover, a self-made VME DSP-board is in the final test phase to control the booster ramping process. Generic stream device drivers for the IEEE/GPIB and CAN fieldbusses have been developed and have already been in use for one year [2].

The number of high level software tools for data acquisition and evaluation as well as for remote device control and complex software feedback is increasing continuously. The availability of a wide variety of interfaces to the EPICS Channel Access (CA) protocol makes it possible to include state of the art software platforms and architectures like the CORBA middleware, Java, Delphi, Tcl/Tk etc. very easily. Furthermore, the integration of complex simulation, design and modeling tools together with database archiving and Web-based accelerator monitoring is in progress.

1 INTRODUCTION

During the last years the self-made DELTA control system has been replaced by EPICS. The change had to be done step by step because storage ring operation had to be ensured during the migration. In summer 2001 this migration was finished. The whole facility, the 80 MeV linac, the 1.5 GeV booster, the storage ring and the Superconducting Asymmetric Wiggler (SAW), is now handled by EPICS.

Combined with the migration of the control system exchanges of VME computer types (from Force- and Microsys-CPUs to Power-PC) and upgrade of the internal Ethernet network to 100 MBit and fiber optics had been made.

2 REACHING EPICS

The DELTA control system has the usual three level soft- and hardware structure consisting of fieldbus systems which are linked via 28 VME-IOCs and local network to the front-end computers for high level operator interfaces. Altogether, there are 46 computers in the control network.

Because many devices had to be accessed by serial protocols, we developed a generic driver which can be configured by an plain text file. Due to the modular structure (see fig. 1) of this driver, new EPICS records and also new fieldbus systems can be applied easily. The flexibility of the driver leads to its use for 672 of the 4767 records (process variables in EPICS) in the DELTA control system (see table 1).

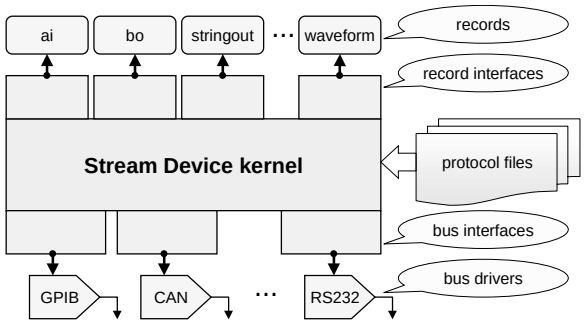


Figure 1: Structure of Stream Device Driver

HP-Workstations as operator interfaces are replaced by Linux-PCs. At the moment, there are still several workstations in use, but mainly as display computers for programs running on the PCs.

For the reengineering of the display panels, the EPICS Megawidget (EMW)-GUI-builder was developed so that even complex graphical user interfaces (GUI) can be generated without programming knowledge. The whole front-end programming was done on the basis of Tcl/Tk. Even extensive applications like beam based calibration (BBC) routines and online optic measurements are written with this toolkit.

Table 1: Records in the Delta Control System

Device Type	Number of Records
Soft Channel	1909
Stream	672
Delta DSP	561
CANbus	497
Delta-FG	490
Desy CPS	244
PPIO Chopper	222
Others	172
Total	**4767**

3 ORBIT DRIFT FEEDBACK

On a slow time scale, DELTA shows significant orbit drifts. Two main reasons could be detected: thermal effects and field drifts of the Superconducting Asymmetric Wiggler (SAW). Thermal effects are beam current related and caused by the heat load from synchrotron radiation. The changing temperature leads to deformations of the vacuum chamber. This causes displacements of the quadrupole magnets [3].

While the Superconducting Asymmetric Wiggler (SAW) is operative, its slowly increasing field error is the dominating reason for orbit drifts. This field error is caused by different current losses in the three separate circuits of the SAW [4].

Both effects could be successfully suppressed by a EPICS-based feedback system implemented in Tcl/Tk on a Linux PC. Orbit correction is performed once a second in both planes. The machine operator can choose from a variety of different correction algorithms. All correction methods work on measured beam response matrices, which have proved to be a better basis than calculated models [5].

Currently implemented methods are:

- most effective corrector
  This chooses the steerer that will minimize the RMS orbit error.
- most efficient corrector
  This chooses the steerer that needs the smallest change of steering to correct the rms orbit error.
- weighted correction
  This chooses a steerer that will correct the rms orbit error while reducing the absolute steerer strength.
- local bumps
  Three consecutive steerers are used to create a local orbit bump that reduces the largest orbit displacement.
- SVD (singular value decomposition)
  SVD is a numerical algorithm to 'invert' non-square or singular matrices. Applied to the beam response matrix, it allows to use all steerers to reduce all orbit displacements simultaneously.

The feedback system uses 43 beam position monitors, 30 horizontal and 26 vertical steerer magnets. The range of the steerers is ±3.1 mrad in the horizontal plane and ±1.1 mrad in the vertical plane. The resolution is $2.4 \cdot 10^{-4}$ of the maximum value. Due to the serial communication interface of the steerers, corrections can only be done slowly.

A correction rate of once a second has emerged to be entirely sufficient to stabilize the orbit on the required time scale. Removal of high frequent orbit oscillations is beyond the intention of this application and needs a separate set of fast steerers and faster hardware for acquisition and processing of orbit data.

4 THE BOOSTER DSP NETWORK

The 1.5 GeV booster "BoDo" is actually a ramped storage ring, with an energy range of 35 to 1500MeV and typical ramp cycle times between 2 and 7 seconds, depending on injection and extraction energy.

In May 2001, VME low level software and GUIs were changed from a self-made control system to EPICS, still using the old hardware: simple synchronized digital function generator VME boards (FGs) with external DACs for ramped power supply control, digital I/O VME boards, and CANBus modules with digital or 12 bit analog I/O that provided time stamp precision of about 100ms and readout frequencies of some Hz.

Since October 2001, all ramped booster magnet power supplies (dipole, 6 quadrupoles, 2 sextupoles, steerers), rf power, and all booster diagnostics (beam position measurement, tune control, beam loss measurement, power supply currents, ...) are handled by VME DSP boards developed at DELTA ("DeltaDSP"). New self-made standalone eurocards with DC-isolated temperature-stable 16-bit ADCs and DACs and fast interfaces to the DeltaDSP boards were installed, with a minimum distance to power supplies and signal sources. Up to 8 DAC cards and 128 ADC cards can be connected to each DSP board, with data transfer rates up to 32 MByte/s for DAC cards (16 bit parallel bus, 10 m max. cable length) and 10 MBit/s for ADC cards (serial daisy chain, 50 m max. cable length). All booster diagnostics data is now available at ADC sample rates between 10 kHz (BPMs) and 100 kHz, with 300 ns time stamp precision.

Each DeltaDSP board has two 32 bit floating point DSPs (Analog Devices ADSP-21062), 3 MByte fast SRAM program memory, up to 64 MByte DRAM data memory and 1 MByte VME dual ported memory. External devices can be connected to four synchronous serial interfaces (32 MBaud max.) or two CANBus master interfaces (1 MBaud max.). Furthermore, a general-purpose flatwire differential I/O port is provided. Reconfiguration of an FPGA connected to this

port provides flexible high-speed interfacing to different accelerator devices like DACs, DDS function generators or switched power supplies with inputs for digital current control via pulse width modulation.

Up to 255 DeltaDSP boards can be connected to a multiprocessor cluster using "DeltaNet", a novel FPGA-based 160 MBaud real-time field bus developed at DELTA. DeltaNet provides real-time data transfer with predictable bandwidth and bus access time, software-independent trigger distribution and accelerator-wide clock synchronization that tolerates transmission errors. Each DeltaNet node is connected to its next neighbour node by one fiber optic cable (max. length 2km node to node, 200km overall). The synchronous hardware design allows future DeltaBus implementations with much higher baud rates. ICs for a 1.25 GBaud implementation are already available.

The DeltaNet bus protocol and interface are completely implemented in hardware (FPGA), including 32 bit CRC checksums and state machines for power-up, bus arbitration and error handling. This solution provides maximum speed (1.5 μs package routing time, independent of data size) at minimum software overhead. Package transfer between DSP and DeltaNet FPGA uses DMA. All DeltaNet nodes receive all data packages transmitted by all other nodes.

The DSP boards have a generic EPICS driver/device support, i.e. the same driver/device support is used for DSP boards with different firmware and FPGA configurations.

Before installing the DSP system, all parameters of the booster ramping process had to be optimized manually, which usually took many hours. Since the control system could not provide most diagnostics data with sufficient time resolution and precision, reasons for bad booster performance were hard to detect.

Since the DSP system makes all diagnostics data available to all DSP boards via DeltaNet with high speed and precision, it is now possible to automate machine operation by implementing intercommunicating distributed real-time control loops up to some 10 kHz bandwidth, e.g. fast orbit feedback, digital power supply current control and real-time tune and beam optics correction [6]. Furthermore, the DSP system can be adopted for a fast global orbit feedback in the main storage ring.

5 DEVELOPEMENT OF AN AGENT SYSTEM

An agent system consists of several independent programs which can communicate with each other and which perform their task highly autonomously. The main advantages of an agent system are the high flexibility and the possibility of adding new agents to meet additional requirements of the system [7]. Both features are needed in the DELTA accelerator facility: on the one hand easy handling and if possible automatic control for routine operation is required, on the other hand we need a flexible toolbox for accelerator experiments.

The knowledge base of the planned agent system is split into two parts: the actual machine values are handled by EPICS and for the more static data, like magnet positions, an Oracle database was installed. Most of the agents will access at least one of these sources. In addition, each agent has the ability to ask the other agents for information via TCP socket connection.

For example, the alarm handler can provide information about actual errors. The display-agent can ask for these errors and determine the physical position of the underlying hardware with the help of the static database. This combined information can be displayed on a map representing the machine and so the responsible technician can find the faulty devices (avoiding confusion about different naming conventions used by the technical staff and the control system).

At the moment the design requires six agents: a manager-agent for debugging and control of the running agent system, an alarm handler which can detect faults and display errors in natural language, a status-agent for valuation and display of the actual machine state, a display-agent to provide information about hardware positions, help texts, naming conversions and technical terms, a program handler which supports the locating and execution of programs of the control system, and last but not least an expert system for semiautomatic control of routine operation.

REFERENCES

[1] D. Schirmer, et al., Standardization of the DELTA Control System, Proceedings of the 7th ICALEPCS 1999, Trieste, Italy
[2] www.delta.uni-dortmund.de/controls/pub/doc
[3] G. Schmidt et al., Position Monitoring of Accelerator Components as Magnets and Beam Position Monitors, DIPAC 2001, Grenoble
[4] D. Schirmer, et al., First Experience with the Superconducting Asymmetric Wiggler of the DELTA Facility, EPAC 2000, Vienna
[5] D. Zimoch, Ph.D. thesis, to be published
[6] B. Keil, Ph.D. thesis, to be published
[7] G. Weiss, Multiagent Systems, MIT Press 1999

WEAP005
physics/0111080

UPGRADING OF THE BEAM DIAGNOSTIC SYSTEM OF U-70 BEAM TRANSFER LINES

V. Kovaltsov, A. Loukiantsev, A. Matyushin, V. Milyutkin, I. Romanov, V. Seleznev,
A. Sytin (IHEP, Protvino, Russia)
M. Clausen (DESY, Hamburg, Germany)

Abstract

The beam diagnostic system of U-70 beam transfer lines (beam profiles, intensity and beam losses measurements) was designed in the beginning of the 80s based on an 8-bit microprocessor, SUMMA hardware and a home made serial communication link. Because of maintenance problems the decision was taken to upgrade the hardware and software parts of the system.

The main implemented features of the modernized system are:

- at the console level – Sun SPARC Station (Solaris 2.5.1) and PC (Linux, Windows NT, eXceed);
- at the front-end level – VME crates with MVME 167 CPUS running VxWorks and microcontroller based VME modules;
- at the application level – the process control and data acquisition toolkit EPICS.

The architecture of the upgraded system, the first experience of its usage in U-70 accelerator run and possibility of integration the beam diagnostics into the common beam lines controls are discussed.

1 INTRODUCTION

Created in the beginning of the 80s and still in use the diagnostic system of the transfer lines of ejected beams [1,2] provides the measurement of:

- Profiles and position of beams in both fast and slow ejection modes. Profilometers are placed in 20 locations of the beams complex, have 16 channels in each of two orthogonal planes with a space resolution of 1, 2.5, 5, 8 and 10 mm;
- Intensities in the range of 10^{10}–10^{13} protons/cycle for fast ejection and in the range of 10^{7}–10^{13} protons/cycle for slow ejection with accuracy of several percents in three cuts;
- Relative allocation of proton beam losses around the transfer lines in a dynamic range of 10^{7}–10^{5} protons/cycle for both the fast and slow ejection modes. Totally 16 detectors were installed on beam transfer lines (BTLs).

The system included PDP-11/40 (RSX-11M) compatible support computers and home made CAMAC-like crates housing programmable controllers and a set of I/O modules connected with the specialized analog electronics placed close to appropriate detectors [1,3]. Remote access to diagnostic information was implemented by communication boards placed in a separate crate.

The structure of the upgraded diagnostic system of BTL is shown in Fig. 1. It is based on personal computers running Linux, VME crate controllers MVME 167 under VxWorks, a set of microprocessor I/O modules and the software tools of EPICS, distributed over the network. A package like eXceed running under Windows on PCs could allow PC console users access to the beams diagnostics information stored and represented in the different forms using EPICS graphical applications. A Sun Station (Solaris 2.5.1), not shown in Fig. 1, is used for development of different software (drivers, etc.) for MVME 167 IOCs. The modification does not affect specialized analog electronics.

2 VME MODULES

The application of the VME standard has lead to development of microprocessor based boards such as the timing generator VTG, the fast scanning 12-bit ADC VSA and the 16-bit ADC VMA [4], aimed to solve real time tasks of beam profile and intensity measurements.

All mentioned modules share a common architecture, a standard map of address space and uniform assignment of bits of the status and control register (CSR). They are based on the microprocessor chip DS80C320 and have 64 KB of dual port memory (DPM) allowing access from both the microprocessor and host processor on the MVME 167. The address space of the modules has three logical areas: 1) The control area used for reading from and writing to the CSR and interrupt vector registers, 2) the parameter area containing the operational parameters of modules, and 3) the data area storing the data acquired by the boards. The definition of the module's operational mode and reading of the results is realized through the DPM, while the operational control of modules is carried out through the CSR register.

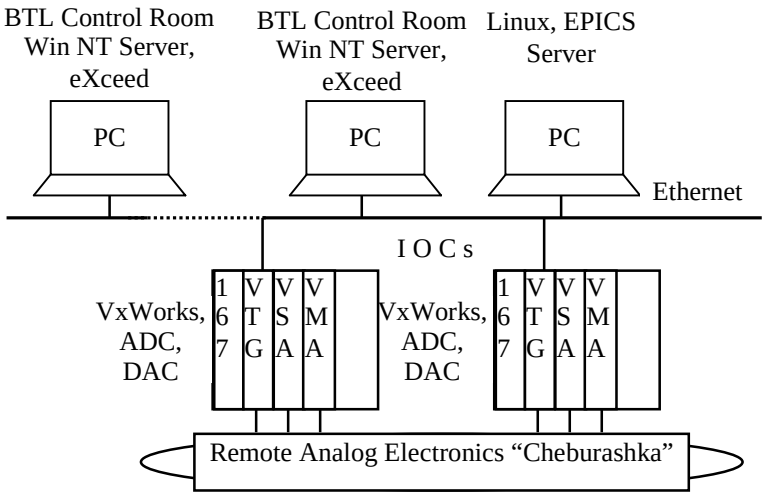


Figure 1: General view of the hardware/software structure of the beam transfer lines (BTL) diagnostic system

The module software reuses common subroutines for all types of modules, namely INIT (initializing of hardware), CONFIG (updating all controlled parameters), and MAIN (event queuing). The other subroutine – EXECUTE – is module specific and treats events from the queue. Such approach facilitates simplicity of software maintenance and future developments for new modules.

The timing generator module – VTG – executes main management and synchronization functions in the systems dealing with the measurement of beam profiles and intensities. When receiving the clock pulse, it implements hard-coded sequences of operations such as permit or disable the integration, control of channel multiplexor, start and stop analog-to-digital conversions.

The VSA and VMA modules are normally looping in waiting for the external trigger, handling the single measurement of input signals for all their channels (16 per VSA, 32 per VMA) and archiving data in their own DPM. An incoming stop pulse breaks the modules out of this loop, allowing the host processor to extract data from the DPM.

3 IOC SOFTWARE

The software of an input-output controller (IOC) in the diagnostic system is based on the EPICS package [5,6] and is running under VxWorks [7]. A corresponding driver has been developed to support all types of modules used. A number of EPICS record types are involved including analog input (**ai** class), analog output (**ao** class) and data array input (**waveform** class) [8,9]. The records of **ao** type are used to specify the parameters of the VTG module, whereas the records of **ai** class ensure reading the current state of modules in the system. The beam profile data is represented by records of **waveform** type.

The driver developed ensures the continuity of profile and intensity measurement. It handles interrupts from the VTG board, keeps all acquired data and finally prepares the signals and zero level measurements in the next accelerator cycle. It supports all used hardware modules because of their tight interdependency. For example, obtaining the interrupt from VTG, it scans all ADC units in the system, reads their data, loads a new configuration and starts all modules. The execution of such time consuming procedure would be much less efficient when running different drivers in the same IOC.

4 DATA REPRESENTATION

The EPICS package includes graphical applications [7] implementing the man-machine interface in control systems. We use MEDM as well as DM2K which are graphical display builders and real time managers. The StripTool program is also rather useful, it permits to watch data of arbitrary selected record groups with the possibility of historical data review.

A hierarchy of displays has been developed for representing 1) the general map of U-70 beam transfer lines, 2) a selected beam transfer line with a set of profilometers installed on it, 3) an arbitrary selection of one or several profilometers. Profilometer data contains the distribution of beam profiles in two directions, the position of the beam's center of gravity, the full charge accumulated by the profilometer, and the beam width satisfying to controlable threshold condition. Some MEDM displays are shown in the figures 2, 3.

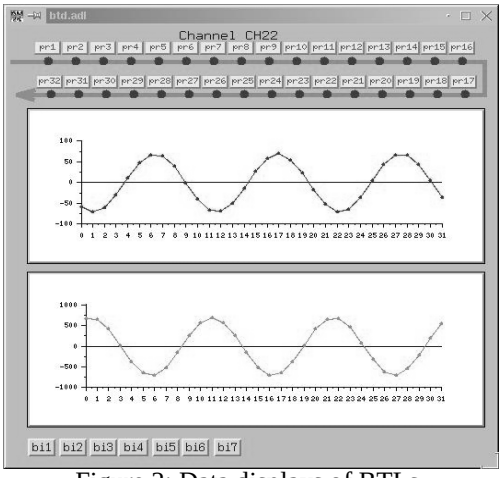


Figure 2: Data displays of BTLs

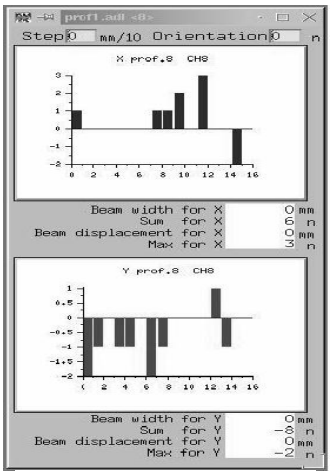


Fig.3: Profilometer display

5 CONCLUSION

The upgraded diagnostic system of U-70 BTLs supports measurement and visualization of beam profiles and their intensities for two channels. The software for diagnostics operates within the framework of EPICS tools.

Further extension of the diagnostic system will also include measurements of beam losses. All the equipment needed for this project is underway.

Next year the diagnostic system will be integrated with the extracted beam control system.

REFERENCES

[1] V. V. Gotsev, V. N. Gres, Yu. P. Davydenko, et al., "Monitoring System of Ejected Proton Beams of IHEP Accelerator", IHEP Preprint 88-106, Serpukhov, 1988 (In Russian).

[2] Yu. P. Davydenko, Yu. P. Romankov et al., "Multifold of Beam Profile During Slow Extraction", IHEP Preprint 90-13, Serpukhov, 1990 (In Russian).

[3] V. Gotsev, V. Gres, et al., "Controlling the UNK transfer line beam diagnostics", Proceedings of ICALEPCS'93, Berlin, Germany, Oct. 18-23, 1993.

[4] Yu. Bardik, E. Kallistratov, A. Makhnachev, A. Matioushine, G. Obukhov, "Microcontrollers applications for IHEP accelerator control", Proceedings ICALEPCS'95, Chicago, Illinois, October 30 – November 3, 1995, p. 980.

[5] W. McDowell et al., "EPICS Home Page", "http://www.aps.anl.gov/asd/controls/epics/EpicsDocumentation/WWWPages/EpicsFrames.html".

[6] L. Dalesio et al., "The EPICS Architecture: Past, Present, and Future", Proceedings ICALEPCS'93, Berlin, Germany, October 18-23, 1993, p. 179.

[7] VxWorks Programmer's Guide, WindRiver Systems, 1994.

[8] J. Anderson, M. Kraimer, "EPICS IOC Record Reference Manual", ANL, APS, December 1, 1994, DRAFT APS Release 3.12.

[9] M. Kraimer, "EPICS IOC Application Developer's Guide", ANL, APS, November 1994, APS Release 3.12.

[10] V. Alferov, Y. Bordanovski, et al., "U-70 Proton Synchrotron Extraction Beam Lines Control System Modernization", IHEP, Protvino, Russia (This conference).

[11] A. Matioushine, A. Sytin, P. Vetrov, S. Zelepoukine (IHEP), M. Clausen, W. Ebenritter, B. Schoeneburg (DESY), "A Configurable RT OS Powered Fieldbus Controller for Distributed Accelerator Controls", Proceedings ICALEPCS'97, Beijing, China, November 3-7, 1997, p. 321.

WEAP006
hep-ph/0111252

CONTROL SYSTEM OF THE BEPCII

J. Zhao, C.H. Wang, X.C. Kong, G. Lei, S.F. Xu, Q. Le
IHEP Beijing, 100039 P.R. China

Abstract

Recently the Chinese Academy of Sciences has chosen BEPCII as the future development of the Beijing Electron Positron Collider (BEPC), i.e. upgrade of both the machine and detector. The luminosity of the machine is expected to increase to $1.0x10^{33}cm^{-2}s^{-1}$. The project will be started at the beginning of 2002 and finished within 3-4 years. The BEPC control system was built in 1987 and upgraded in 1994. According to the design of the BEPCII, a double ring schema will be adopted and a number of new devices will be added in the system. The existing control system has to be upgraded. The BEPCII will be distributed architecture and developed by EPICS. We are going to apply the standard hardware interfaces and mature technologies in the system. A number of VME IOCs will be added in the system and the feildbus, PLCs will be used as device control for some kind of equipment. We will keep the existing system in use, such as CAMAC modules and PC front-ends, and merge it into EPICS system. Recently the development of the prototype is in progress. This paper will describe the system design and development issues.

1 BEPCII PROJECT

The BEPC was constructed for both high energy physics and synchrotron radiation research. BEPC accelerators consist of a 202 m long electron-positron linac injector; a storage ring with a circumference of 240.4 m, and 210 m transport lines. There are two interaction points on the storage ring and a detector, the Beijing Spectrometer. The Beijing Synchrotron Radiation Facility has 9 beamlines and 12 experimental stations. As an unique e+,e- collider operating in the τ-charm region and the first SR source in China, the machine has operated well for over 10 years since it was put into operation in 1989.

The upgrade to the BEPC, known as BEPCII, will increase its luminosity to $1.0x10^{33}cm^{-2}s^{-1}$ with the major upgrades in both the machine and the detector. Running at the resonance peaks of J/ψ and ψ', the BEPCII could provide data samples of J/ψ and ψ´ with good statistics for many important physics [1]. To reach the goal of the higher luminosity, the double ring schema will be adopted and much new equipment will be installed in the BEPCII, such as superconducting RF cavities, superconducting magnets, new BPMs and beam feedback system. The design goals of the BEPCII are in Table 1.

Table 1: The design goal of the BEPCII

Beam Energy range	1- 2.8GeV
Beam energy	1.55 GeV
Beam current	1.116A
Luminosity	$1.0x10^{33}$ $cm^{-2}s^{-1}$ @1.55GeV
Injection from linac	Full energy injection
Dedicated SR operation	250ma @ 2.5GeV 150mA @ 2.8GeV

2 CURRENT SYSTEM

The current control system of the BEPC was built in 1987, which was transferred from SLAC New Spear system and upgraded in 1994. A VAX4500 computer with CAMAC hardware controls most of the equipment on the storage ring such as the magnet power supplies, the RF cavities and the vacuum equipment. There are about 300 CAMAC modules and 4,000 channels in the system. Several subsystems are PC-based systems including the beam diagnostic system, the injection power supply and the linac system. Two VAX workstations serve as the console and all of the machines are connected with 100Mbps Ethernet. On the software side, the system is an in-house made database driven system. The applications for accelerator operation with its Operator Interface (OPI) are closely linked to the real-time database. However, the accelerator commissioning model of the BEPCII is different from BEPC, so the accelerator commissioning programs have to be transferred from other laboratories and modified. In this case, the existing real-time database does not match the new transferred applications and the old OPI written in the FORTH language will not be used in future. The control system has to be upgraded.

When the system is upgraded, we will utilize the existing equipment of the system, such as the CAMAC modules and PC-based subsystems. The standard hardware interfaces should be applied in the system so that it could be an open and standard system. With regard to the software development environment, EPICS will be a good choice for it is a mature software

package and widely used in the world. The cost-performance of the system should be considered for the limited budget.

3 SYSTEM DEVELOPMENT

3.1 Software Engineering

Project management is very important for the development of a control system. The BEPCII control system has to be delivered on time and within the budget and it should meet the requirements of accelerator physicists. The standard of software engineering will be used in the project, which is based on the software life cycle reference model. It includes several phases: the user requirement phase, system analysis and software requirement analysis, system design, coding and testing, installation and commissioning. The documentation and review for the major phases will be done. A software tool "Project 2000" will be used to manage the project BEPCII. And we are going to produce a Chinese template of the documents for the project management that will be used by our users and developers.

Now the user requirement phase is in progress. A detailed system design will be delivered next March, and a prototype of the control system will be built at the same time. From the beginning of 2003 we will spend two years for system construction, and one year for system installation. We hope the BEPCII will finish its commissioning in December of 2006.

3.2 Development Tool

To develop a large scale control system, a SCADA (Supervisory Control and Data Acquisition) tool kit should be used. Currently there are many commercial SCADA products in the world market and most of them are running on the PC and Windows NT platform, which mainly support PLC hardware. The EPICS can be considered a non-commercial SCADA tool, which was first developed by LANL and ANL and is widely used in the accelerator area. We are going to integrate the BEPCII control system by the EPICS after evaluating the commercial SCADA products and EPICS [2]. The benefit of using EPICS is that a lot of applications for accelerator commissioning could be shared; it strongly supports the VME hardware and it is easy to get technical support from many HEP institutes in the world, such as KEK-B, SLAC, APS, LANL and BESSYII. Some independent and slow control systems could be developed by a commercial SCADA product.

4 SYSTEM ARCHITECTURE

The BEPCII control system will adopt a distributed architecture, called the "standard model". Logically, the system is structured with three levels, which are the presentation layer, the process control layer and the device interface layer [3].

As shown in Figure 1, several PCs, the cost-effective equipment, and SUN workstations will be used as the operator console at the presentation layer with the EPICS/OPI, which is a friendly graphic man-machine interface. There is a server machine in the top layer installing an Oracle database, which provides data logging and analysis service, the accelerator commissioning support, and general computing resources [4].

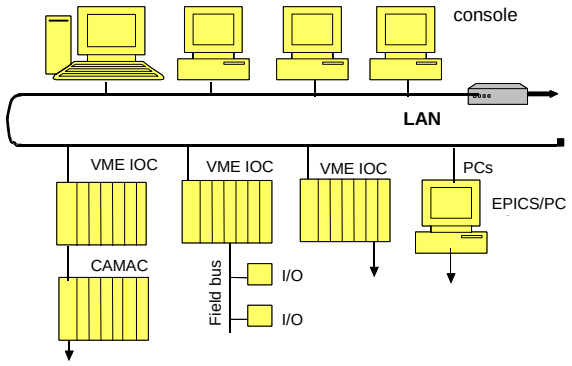


Figure 1: System architecture

In the process control layer, the PowerPC micro-processors running a VxWorks operating system will be used as EPICS IOCs. In some existing subsystems the PCs are still used as the front-ends. The VAX4500 computer will be eliminated. The majority of real-time tasks will run on the front-ends, and the raw data are stored in the IOC real-time database. The standard 100Mbit/sec switch Ethernet with TCP/IP protocol is used as a LAN, which provides access to the distributed computers.

The device interface layer provides an interface to the hardware, either as separate modules or as intelligent controllers. The fieldbuses will serve data exchange with PLCs, intelligent controllers, and remote I/O modules.

5 SUBSYSTEMS

There are seven subsystems in current BEPC control system: the magnet power supply system in the storage ring and the transport lines, the radio frequency system, the vacuum system, the injection system, the beam diagnostic system and the linac system.

To upgrade the control system and meet the requirements of the BEPCII, the new parts of the system will be built with EPICS and some existing parts should be merged into the EPICS system. (See Figure 1)

The BEPC has a single ring and the BEPCII will adopt a double ring schema to obtain higher luminosity.

The number of the magnet power supplies will be increased on the storage ring (see Table 2) and the superconducting RF cavities, new BPMs and beam feedback system will be added in the BEPCII. This equipment will be controlled by ten EPICS IOCs which consists of VME crates, PowerPCs, DSPs and VME I/O modules.

Table 2: Number of magnet power supplies on SR

Power supply for	BEPCII	BEPC
Bending magnets	6	1
Quadruple magnets	144	21
Correctors	160	64
Superconducting Mag.	10	0
Sextuple magnets	4	4

The vacuum equipment and magnet power supplies of the Linac accelerator will be controlled by PLCs and remote I/O modules, which are connected to the EPICS IOCs via fieldbus. ControlNet and CANbus are the candidate of the fieldbuses [5].

There are several PCs for beam diagnostic system running applications developed by Labview. The EPICS/Portable Channel Access will be installed on these PCs to merge the existing system to the EPICS system.

Some parts of the control system still use CAMAC I/O modules. We are going to add VME IOC in the local control area and connect the CAMAC system to the EPICS system with VME-CAMAC interfaces.

For slow controls, such as cooling water system, we are going to develop the independent system with commercial SCADA products, because it supports PLCs, various sensor and controllers with industrial standards [6].

6 APPLICATIONS

There are two kinds of applications in the system. One is the application of device control; the other is the application for accelerator commissioning. For the device control, we have to configure our own IOC database first. If needed, there are some I/O drivers that have to be developed for the special hardware interface. The graphic man-machine interface will be developed by the EPICS tool DM2K. A lot of applications for device control should be developed by EPICS tools or written with C/C++, Tcl/tk or the SNL language of EPICS. The accelerator commissioning software from other laboratories such as KEK-B or PEP-II is being considered for use and it has to be modified based on the physical requirements of the BEPCII.

7 DATABASES

There are two kinds of databases in the system, one is a distributed real-time database running in EPICS IOCs to store raw data; the other is a relational Oracle database, which will be installed on a server machine to store static and dynamic data including system configuration data, machine parameters, historical and alarm data etc. And both of the two databases provide a user interface on the Web page and publish the running information. We must first make signal naming conventions used in the databases and create the database structures.

8 TIMING SYSTEM

The timing system is to synchronize all of the relevant components in the accelerator of the BEPCII complex. Since the RF frequency of the BEPCII will be changed from 200MHZ to about 500MHZ, the current timing system has to be upgraded. Except for the hardware based fast timing system, the software-based timing system might be considered in some synchronization area.

9 CONCLUSION

The BEPCII project will be started next year. In order to build the control system with EPICS and develop basic applications, a prototype system should be created first. After making a great effort, we think the upgraded system can meet the physical requirements of the BEPCII and it will be an advanced, flexible and reliable system.

10 ACKNOWLEDGEMENT

The authors would like to thank all of the people and friends who have given us a lot of help and advice in the system design stage.

REFERENCES

[1] “Feasibility Study Report on BEPCII ”, Edition1, IHEP April 2001
[2] “EPICS Release R3.14”, APS Argonne.
[3] Tadahiko Katoh, “Design and Construction Accelerator Control System” KEKB control group, 28 Aug. 2000.
[4] J. Zhao et al, “Preliminary Design of the Control System for SSRF and BTCF” ICALEPCS’97 p56 – p58, Nov. 1997 Beijing, China.
[5] http://www.sns.bnl.gov/epics/cnet/
[6] A. Daneels, W. Salte “What is SCADA” ICALEPCS’99 p339-343 Oct. 1999 Trieste, Italy.

WEAP007
physics/0111105

A PROTOTYPE OF UPGRADING BEPC CONTROL SYSTEM

C.H. WANG, X. C. KONG, G. LEI, S.F. XU, Q. LE, and J. ZHAO
IHEP, BEIJING, 100039, P.R. CHINA

Abstract

We will use EPICS toolkit [1] to build a prototype for upgrading BEPC control system. The purposes are threefold: (1) Setup a network based distributed control system with EPICS. (2) Study some front-end control technology. (3) Merge the existing control systems into EPICS. The hardware architecture consists of three physical layers. The front-end layer consists of VME crate, PowerPC, VME I/O modules and interface boards (VME-CAMAC and Fieldbus). The back-end layer is a SUN workstation, which is used for application development. The two layers are connected via the middle layer which is 100M Ethernet. In order to preserve our investment in the hardware, the existing CAMAC hardware will remain. The CAMAC will be connected to the VME via CAMAC serial highway [2]. The operator interface will be developed with EPICS DM2K. High-level accelerator control applications will be studied as well with EPICS channel access and archive access.

1 INTRODUCTION

BEPC consists of a 200 m long linac and storage ring with a circumference of 240.4 m. It can produce some extremely important high energy physics and synchrotron radiation.

The storage ring control system was transplanted from SPEAR in 1987. The upgrade of this system in 1994 was to replace a couple of very old systems, which were either already obsolete or unsupported (VAX750, Grinnell, VCC). The upgraded system has worked reliably for us.

But this system after the upgrade still has the following shortcomings: (1). it is heavily dependent on the VMS operating system and CAMAC, (2). it is not open to the option of non- CAMAC hardware such as VME, (3). There is a communication bottleneck because real-time data reside in shared memory on a single computer, and (4). the operator interface is intimately bound to the application programs and is primitive by today's standards.

Because the commissioning of BEPCII will be different from BEPC, we are unable to use the FORTH-based OPI and the existing real-time database. Therefore, we will modify and adopt the control system used at other labs for use with BEPCII. Except some subsystems, the control system hardware and software architecture will be upgraded.

The linac control is different from the ring. It uses PC with remote I/O via RS422 to control the devices. The console in the ring only can access some dada such as the Phase in the linac via the network.

2 REQUIRMENTS AND OPTIONS

2.1 Requirements

Our motivation for doing the upgrade of BEPC is to meet BEPCII [3] requirements. BEPCII will construct double rings in the current tunnel. BEPCII will add some high voltage switch power supplies in each ring. There are about 400 magnet power supplies distributed around the rings and transport lines of BEPCII. So, we need to build new power supply control system on the rings. In order to preserve our investment, the CAMAC in the transport line will remain. We will take a power supply as an example in the prototype and implement the control of the power supply with EPICS. For the requirements of the power supply control are:

- Current setting synchronously without beam loss.
- the fastest time period is 30ms/per step setting.

2.2 Options

Several proposals were discussed on how to meet requirements. The proposals submitted consisted of:

- VME + Fieldbus + Intelligent Device controller.
- VME + direct I/O.

We did some market survey for the fieldbus such as Canbus, DeviceNet, ControlNet and Profibus. We think that Canbus only has two-layer protocol supported with 1Mbps speed; it's hard to develop software. SSRF (Shanghai) already developed the DeviceNet communication software between IOCs and device controller. Although ControlNet is faster than DeviceNet, they both have a problem with signal synchrotron processing. So, it is suitable for slow control such as vacuum system. The power supplies on the rings can be directly controlled by VME I/O modules.

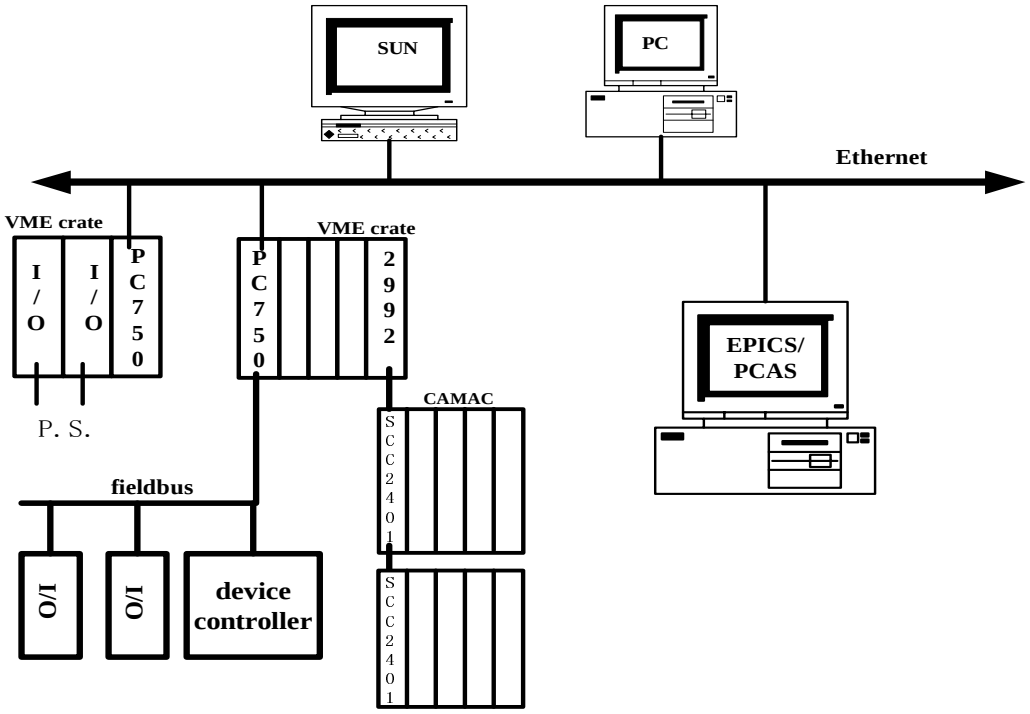


Figure 1: the hardware architecture of the prototype

We are also considering ControlNet for the vacuum control because EPICS already supported ControlNet. We can get the ControlNet driver from the vacuum control group of SNS/BNL.

3 ARCHITECTURE OF THE PROTOTYPE

The prototype consists of a Sun workstation, a PC and VMEbus systems as shown in figure 1. The Sun workstation is used as development. The PC is used as OPI. There are three ways to control the devices. One is VME-fieldbus interface controller and a device controller. Another is VME direct I/O module. Another more is VME-CAMAC interface module. In addition, we will install EPICS/PCAS on the PCs running LabView for merge the PC systems into EPICS.

Our main goals for the prototype are: (1) Setup hardware and software architecture with EPICS, (2) Study key technology of the control system integration, (3) Merge the existing PCs and CAMAC into EPICS.

4 PLANS

We plan to construct EPICS Real-time database with CapFast and ASCII files according to the BEPCII database name convention. We will build new operator screens with DM2k. We will do EPICS applications on the prototype control system for the power supplies and measure the precision and update rate of acquisition signals. We also will develop the application for the vacuum with VME-fieldbus. Meanwhile we will develop high-level accelerator control applications.Finally, we will test new IOCs on some subsystems. The power supply system and vacuum system will be first. Then, other subsystems can also be implemented with EPICS soon.

5 ACKNOWLEDGEMENTS

We would like to thank all teachers of EPICS seminar and EPICS workshop in Beijing in September for their constructive suggestions and helpful discussions.

REFERENCES

[1] http://www.aps.anl.gov/epics/index.php.
[2] Karen S. Nolker, Hamid Shaoee, Willaim A. Watson III, Marty Wise, “The CEBAF Accelerator Control System: Migrating from a TACL to an EPICS Based System”.
[3] “Feasibility Study Report on BEPCII”, Beijing, April, 2001.

WEAP008
physics/0111200

UPGRADE OF THE SDL CONTROL SYSTEM

S.K. Feng, W.S. Graves, Y.N. Tang
Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

The Source Development Lab (SDL) at BNL consists of a 230 MeV electron linac and a 10 m long wiggler for short wavelength Free Electronic Laser (FEL) development. The original control system [1] was based on the one in use at the National Synchrotron Light Source. In 2000 the control system was upgraded to be based on EPICS to facilitate software interfaces for extensions at all levels. The FEL began commissioning in May 2001. The upgraded control systems have been stable, reliable and easy to operate. An overview of the control systems, including for the linac, the waveform digitizer controls, video system/image processing and radiation monitor/alarms will be presented.

1 INTRODUCTION

In SDL, the challenging physics requirements and different favorite approaches for implementations among physicists, users, and programmers initiated the upgrade of the control system to be based on EPICS to facilitate software interfaces for extensions at all levels. As of today, the mixed hardware platforms and software tools of the SDL control systems are integrated homogeneously (see Figure 1).

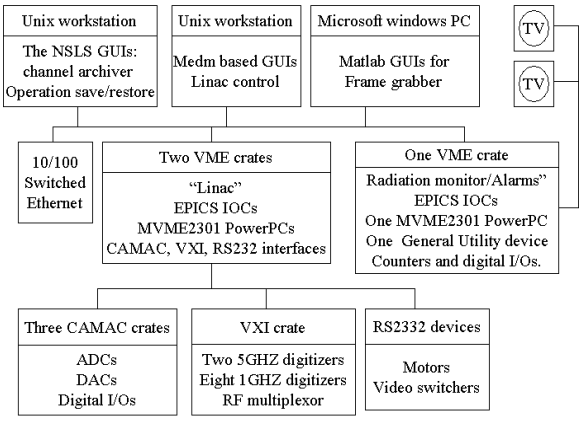


Figure 1: The Hardware/software Illustration of SDL Control System

The VME Input/Output Controllers have interfaces to both CAMAC and VXI crates, and RS232 devices. The software for an RF multiplexor (MUX) and fast VXI waveform digitizers is developed to measure/record/display many voltage waveforms.

2 IMPLEMENTATION

2.1 The waveform digitizers

In the VXI crates, there are two 4-channel 1 GS/s waveform digitizers used for general purpose data acquisition and one 2-channel 5GS/s waveform digitizer used for beam position monitor (BPM) processing. The VXI waveform digitizer control and display, consisting of a MVME2301 PPC/VME-MXI-II/MXI-VXI-II and VXI scopes hardware and EPICS IOC core in low level control, is developed with the MEDM tool and Cartesian plotting to effectively present the required information on the workstation (see Figure 2).

The requirement of the SDL application is to refresh the display of the ten waveforms simultaneously on the workstation at a rate close to 10HZ for each channel. The challenge is to overcome the limited data transfer capability between the CPU and the waveform digitizer, and to transfer 1K bytes of voltage data in floating point format for each waveform (e.g. 80K bytes of data for the ten channels) across the network to display on the workstation without compromising the performance requirement.

Theoretically, the data move performance of the VME-MXI-II and MXI-VXI-II pair tops 8-10 MB/s at single cycles. However, the benchmark of data move between the MVME2301 PPC and VXI waveform digitizers ranges 16-30 KB/s via the Fast Data Channel (FDC) protocol at single cycles. That limits the readouts of each waveform data (2K bytes) per scope channel to be 8-15 HZ in the low level control.

At an early stage of device driver development with EPICS, the ten channels of scopes performed at 2 Hz of waveform display refresh rate when the records were scanned periodically. After further enhancement of real-time programming by employing EPICS event scan and scope driver interrupts, the ten channels of scopes, while running simultaneously under average network loads, achieved an average of 10 Hz of waveform display refresh rate on two channels and 6-7HZ of waveform display refresh rate on the other eight channels.

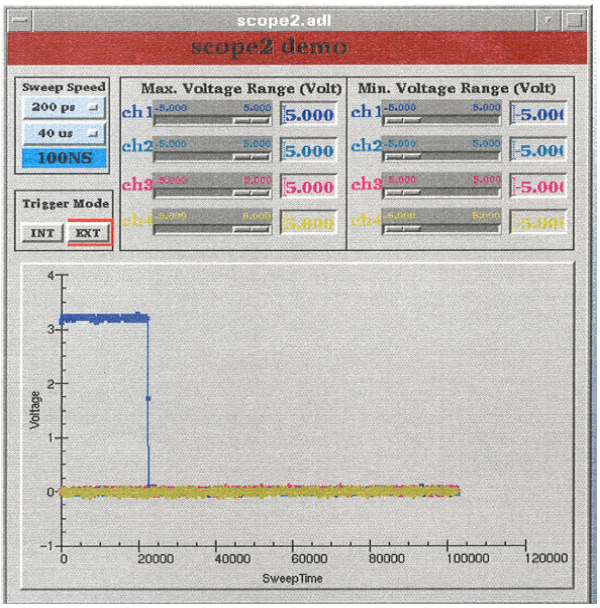


Figure 2:waveform digitizer control/display demo

2.2 Linac control systems

The linac control systems consist of two VME IOCs controlled by MVME2301 PowerPCs in low level control. Most of the ADCs, DACs and digital I/O for the linac are carried using legacy CAMAC equipment. Although the VME/CAMAC driver was obtained from the community of the EPICS collaborations, the EPICS device support routines for our CAMAC equipment were developed in house. Stepper motors and video switchers are controlled via RS232. Once device support routines are developed, new records of the supported device can be added to the system by editing the database without the need of changing the low level device drivers and support routines, thanks to the object-oriented structure of the EPICS layers. The EPICS save/restore program embedded in the IOC facilitates the restore of the output parameters in case of an IOC reboot.

The high level linac control is implemented with the MEDM tool of EPICS to provide friendly Graphical User Interfaces (GUIs). Figure 5 shows the main page of the linac GUI control. The motif-based channel archiver (the history program and its graphic display) and save/restore programs for various control operations, which were in use at the NSLS were modified to include the EPICS channel access to communicate with the linac IOCs.

2.3 Video System and Image Processing

The primary beam diagnostics for SDL are YAG:Ce scintillators that intercept the electron beam and produce light that is imaged by Cohu 4910 CCD cameras. The RS-170 video signals are fed from approximately 60 cameras through video matrix switchers to monitors and a PC based frame grabber for image processing. The Matrox Meteor II/MC 8 bit analog frame grabber is controlled by Matlab MEX C function calls and includes a GUI interface (Figure 3). The frame grabber and analysis software can set and read EPICS process variables utilizing an ActiveX interface.

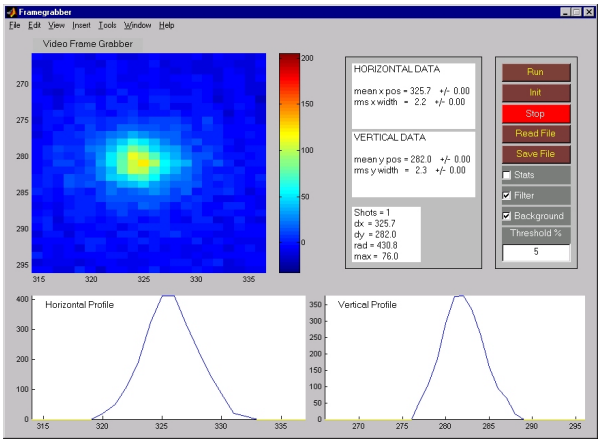


Figure 3: Matlab GUI to PC-based frame grabber

The frame grabber and associated Matlab beam analysis programs provide near real-time diagnostics capability that has been very useful during the commissioning process. Figure 4 shows an emittance measurement that is performed by Matlab function calls. The beam sizes are read from the frame grabber and magnet settings are read from the EPICS database. These values are then used to calculate the emittance and beam Twiss parameters. This enables fast lattice development and optimization of beam parameters.

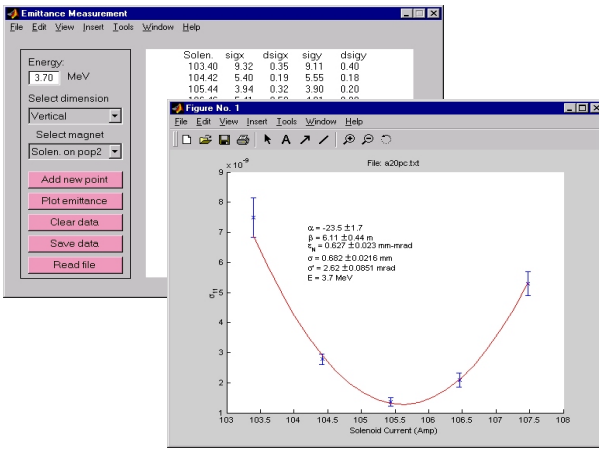


Figure 4: Automated emittance measurement using Matlab and Epics.

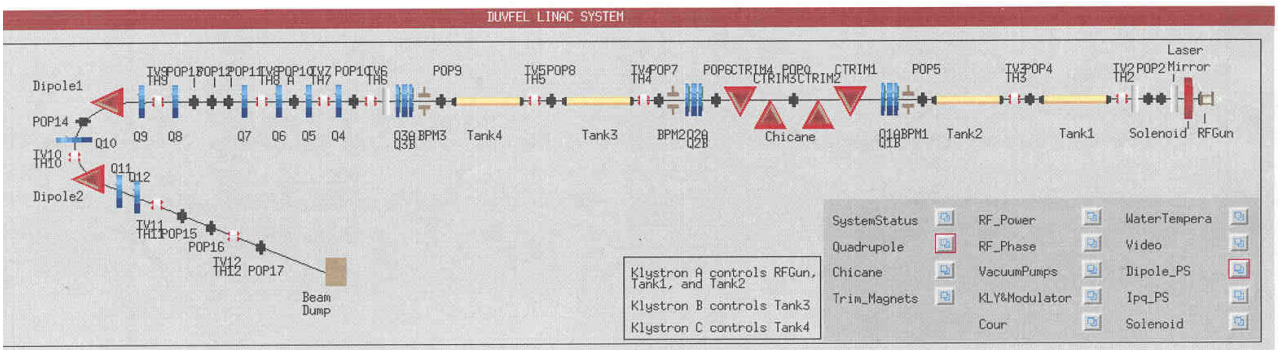


Figure 5: The main page of the SDL control system

3 WORK IN PROGRESS

3.1 Radiation monitors/alarm status

The radiation monitors/alarm status IOC consists of one MVME2301 PowerPC, three 5 - channel counters, digital I/Os and a general utility board for TV message display [2]. The radiation monitor is required for radiation safety to monitor and display radiation levels and alarm status on the TVs. The system is implemented to the same specification as those of the alarm/radiation system which is in use at the NSLS. The advantages of using the TV display instead of an X window display are, 1) the information is real-time independent of the network status, and 2) it eliminates the need of point and click and shuffling the screen. Thus, there is no need to train users who are not familiar with X-windows environment.

We decided it is worthwhile to rewrite the same specifications to be based on EPICS to facilitate software interfaces for extensions at all levels.

4 FUTURE PLANS

For the further development of the SDL control system, we would like to use the VME64x crate, IP carriers and IP modules.

For the image processing, we would like to use the same PC-based frame grabber to be controlled under the LINUX operating system as one of the EPICS IOCs, which facilitates high level GUI remote controls via a workstation.

The workstation for VxWorks development and high level GUI control will be replaced by a high-end PC running the LINUX operating system due to concern for cost and performance.

5 CONCLUSIONS

The SDL control systems have been stable, reliable and easy to operate. We have been able to utilize the pre-existing hardware to minimize costs. We benefited from the object-oriented structure of the EPICS layers to reduce the efforts of maintenance and expansions without compromising the performance of low level real-time control (e.g. the implementations of the waveform digitizers). The upgrade is satisfactory that we will continue to develop and expand our existing control system to increase functionality, ease of use and reliability of the facility.

6 ACKNOWLEDGEMENTS

The authors wish to extend their thanks to J.D. Smith and Eric Blum of BNL and all the other people who have helped with the integration of the SDL control system. This work performed under DOE contract DE-AC02-76CH00016.

REFERENCES

[1] W.S. Graves, S.K. Feng, P.S. Pearson, J.D. Smith, “Innovative Aspects of the SDL control System”, Proc. PAC97 (Vancouver, 12–16 May 1997).

[2] S. Ramamoorthy,J.D.Smith,“GPLS VME Module: A Diagnostic and Display Tool for NSLS Micro Systems”, Proc. PAC99(New York City, 29 March–2 April 1999).

WEAP009
physics/0111067

INTEGRATION OF A NEW KNOBBOX IN THE PSI CONTROL SYSTEMS ACS AND EPICS

A.C.Mezger, D.Anicic, T.Blumer, I. Jirousek
Paul Scherrer Institute, Switzerland

Abstract

Operation of the PSI cyclotrons has always relied heavily on manual control. Tuning of our high intensity beams for minimum losses is an art still depending on personal skill. An ergonomically good design of the knobbox is therefore essential for good operation. The age of the previous design, and the request to implement the ACS knobbox in the SLS control system based on EPICS, was an opportunity for a redesign. The new knobbox is an autonomous node in the system. It is based on industrial PC/104 hardware with LINUX as the operating system. CDEV and ACS-communication provide the interface to the respective control systems.

1 INTRODUCTION

For the operation of the PSI accelerators the Man-Machine Interface (MMI) to the power supplies, motors, etc. consists of a panel presenting knobs, buttons and displays, the so-called knobbox (Fig. 1). The functionality of this interface has been optimized for the handling of our accelerators over the last 20 years. It has the ability to make small increments to a device through buttons, as well as continuous variation with knobs. Many other useful functions have been implemented for keeping the old value when a device is hooked, setting back this old value as well as the possibility to set a new value, switching the power supplies off and a HP-like calculator.

Figure 1: Knobbox MMI

2 HARDWARE EVOLUTION

The first generation of this hardware was implemented through CAMAC modules directly driven by a PDP control computer. The next generation saw the hardware controlled by a CAMAC auxiliary crate controller (ACC, CES 2180) and communicating with the control computer through the parallel CAMAC branch [1].

In the early nineties our control system evolved to a distributed system with front-end and back-end computers, all connected through an Ethernet network (Fig. 2).

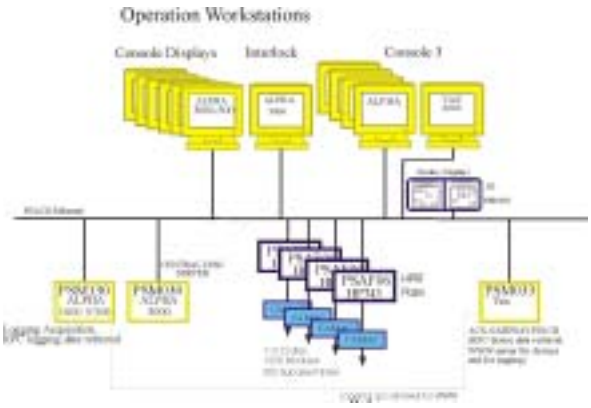


Figure 2: Control System layout

In this architecture, the front-end computers perform the data acquisition on the field bus (VME and CAMAC) and pass the results to the back-end (demanding) computer [2]. To integrate the knobbox into this architecture, a direct connection to the ACC via an Ethernet interface has been implemented. This made the crate standalone and its connection to the parallel CAMAC branch was removed.

3 NEW DEVELOPMENT

The knobbox is a necessary tool for the operation of our accelerators and cannot be replaced easily in all its functionality by modern software. Our Operators are accustomed to it and it is also used in the SLS (Swiss

Light Source) control system. We realized that the hardware associated to the knobbox is getting older, is expensive and furthermore that CAMAC is not always the preferred solution. We looked therefore for a new hardware solution where we could keep the same MMI.

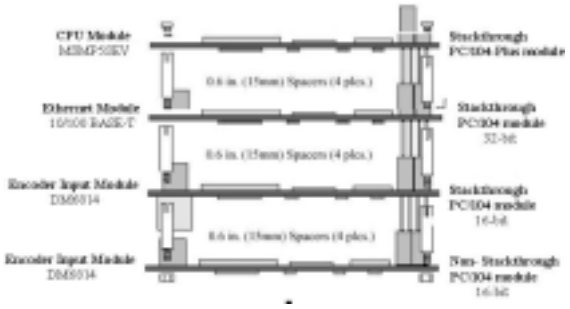


Figure 3: Typical module stack for the knobbox

The solution we have adopted uses an industrial embedded PC/104 system (smartCore-P5, a 266 MHz Pentium PC) with the necessary IO-cards: Up-Down Counters for the knobs, digital IO and Ethernet 10/100 (Fig. 3). These PC/104 cards are much smaller than ISA/PCI-bus cards found in PC's and stack together which eliminates the need for a motherboard, back plane, and/or card cage. Power requirements and signal drive are reduced to meet the needs of an embedded system. Because PC/104 is essentially a PC with a different form factor, any popular operating system can be used for this system. We have chosen LINUX.

The 10-year-old displays of the original knobbox have been replaced with quarter-VGA electro-luminescent displays, giving the possibility to drive the two displays through the TFT connector of the PC/104. The small character display situated in the middle of our knobbox is driven through the parallel port of the PC/104. To simplify the handling of the keyboard and lights we used a field-programmable gate array from XILINX interfaced to the IO-cards. This layout is shown in Fig.4.

4 SOFTWARE IMPLEMENTATION

Because of the new architecture of the knobbox, the software has had to be completely rewritten (the old software was written using FORTRAN and was based on CAMAC hardware). The new software is completely written in C and does its IO through the IO-ports of the PC/104 system. The software uses X-WINDOWS for displaying on the "quarter-VGA" screens. To support these screens the PC/104 BIOS had to be modified; this was done by the Company that supplied us the PC/104 system. The data-acquisition is implemented in the standard way as defined by our control system, however CDEV (CDEV is a C++ framework for developing portable control applications developed at Thomas Jefferson Laboratory) has also

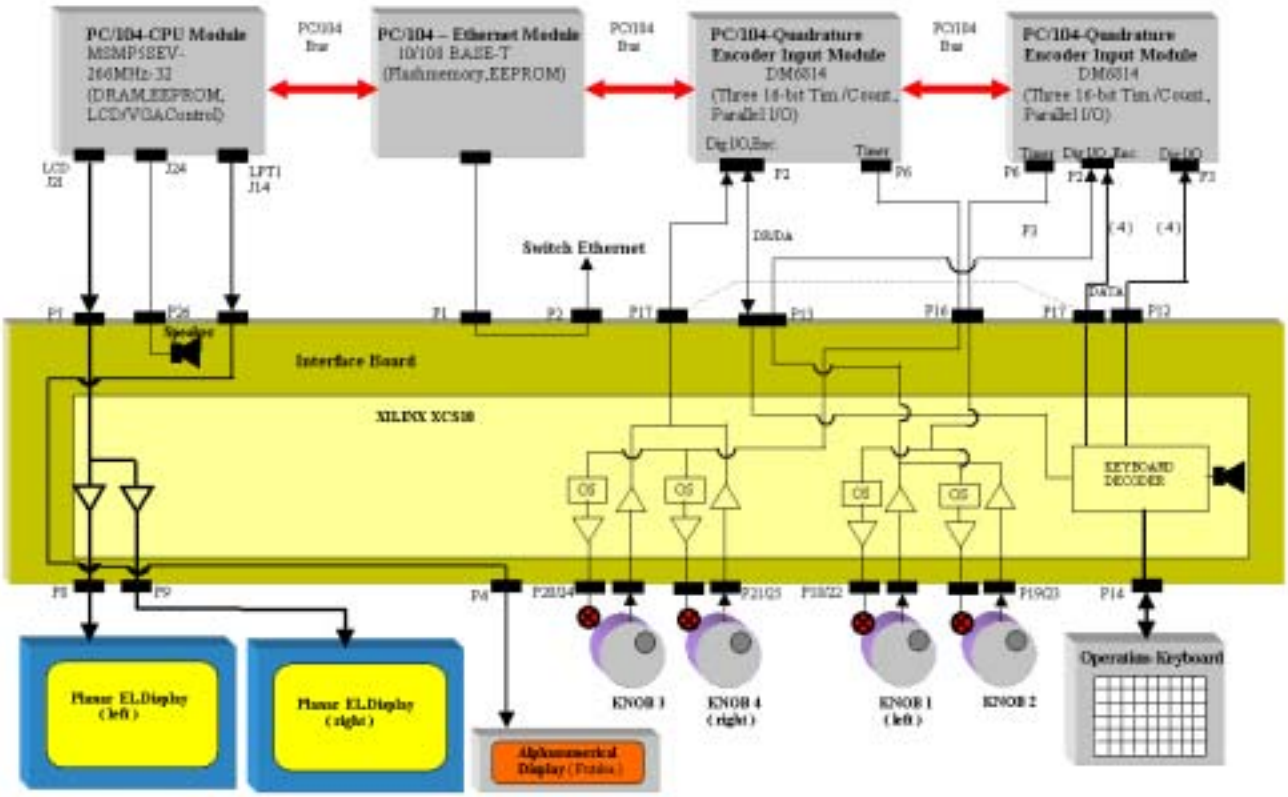


Figure 4: Knobbox block diagram system.

been implemented. This extension to the knobbox allows it to be used directly with any system using CDEV. This requirement is important in the case of SLS or when more than one control system interface has to be addressed.

For the development of our knobbox, a small laptop disk running Linux and the necessary software was connected to the PC/104. Recently we changed to a diskless system booted from a server so that several knobboxes can be installed with only a small effort.

Two knobboxes are in the test phase and will soon be installed into our Control room so that we can test the knobbox under real conditions.

5 ACKNOWLEDGMENTS

The authors would like to thank the following people for their contributions to the hardware of the upgraded knobbox: D.Buerki, W.Hugentobler and G.Janser.

6 REFERENCES

[1] The SIN Operator-Console: Functionality and Performance by A.C.Mezger, T.Blumer, I.Jirousek and L.Sekolec. International Conference on Control Systems for Experimental Physics, Villars sur Ollon, 1987

[2] Status report of the PSI accelerator control system. T.Blumer, D.Anicic, I.Jirousek, A.Mezger, L.Sekolec and Z.Sostaric. Nuclear Instruments and Methods in Physics Research A 352 (1994) 150-157

WEAP012
physics/0111189

THE BATES PHASE AND AMPLITUDE MONITORING SYSTEM

D. Cheever, T. Ferrari, X. Geng, R. Page, T. Zwart
Bates Linear Accelerator Center, Middleton, MA 01949-1526I.

I. INTRODUCTION

The MIT Bates LINAC is a 1 GeV electron accelerator used to perform precision measurements of the structure of nuclei and nucleons. The accelerator is a pulsed RF structure operating at 2856 MHz. Twelve CPI klystrons are situated along the length of the 200 m. LINAC. Each klystron is capable of delivering 6 MW of RF power for 30 us at a repetition rate of 600 Hz. Reliable operation of the machine requires that the relative phase of the 12 klystrons must be maintained to better than 1°. Similarly, the relative amplitude stability of each klystron must be maintained to better than $10^{\wedge -3}$.

To better meet these requirements we have constructed a RF phase and amplitude monitoring (PAM) system. This system samples a small fraction of the RF immediately downstream of each of the twelve klystrons. The complete system includes several main components: RF phase and amplitude detectors, analog signal conditioning, VME based digitizing and EPICS software for monitoring and calibration. The functionality and performance of each of these system components is described in detail below.

II. HARDWARE

Each of Bates' 6 transmitters house a pair of output klystrons and a drive klystron. These signals are available attenuated to under +30dBm via directional couplers on each klystron output wave-guide and reference line taps. The RF signals are patched to a chassis shown in Figures 1 and 2 containing off-the-shelf 2.856 GHz RF components as listed in the appendix [1]. The output of these devices is impedance matched, fanned out, and routed to custom gated integrators and a custom video matrix switch. The integrators integrate and hold a gated portion of the pulsed signals and reset immediately before the next gate pulse, shown in Figure 3. The integrator outputs are low pass filtered at ~30Hz creating –10V to 10V ~DC signals that represents the peak amplitude or phase within the gate and are routed out of the transmitter gallery over multi-conductor twisted pair cabling to the 3 patch rooms where the non-synchronous VME ADC digitizes the signals. An Argonne Timing Board is used to set the relative timing of the reset and gate pulses, triggered by each transmitters RF "start" pulse. The VME DAC produces the ramping signals that shift the phase over 360º via the phase shifter to allow the necessary calibration. The pulsed phase and amplitude signals are also routed to the Bates video switch, enabling the control room to examine the signal for phase or amplitude ramps

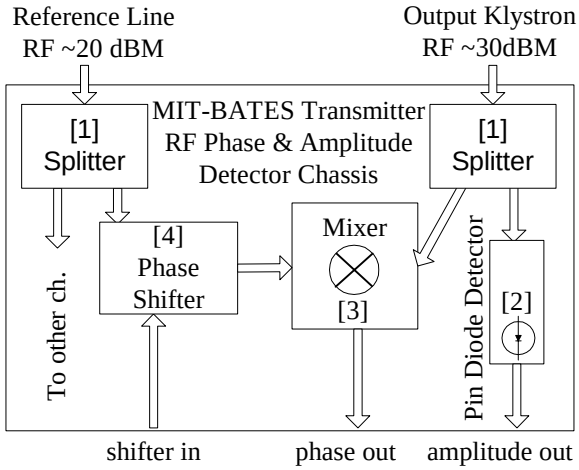


Figure 1. *RF Detection Hardware architecture*

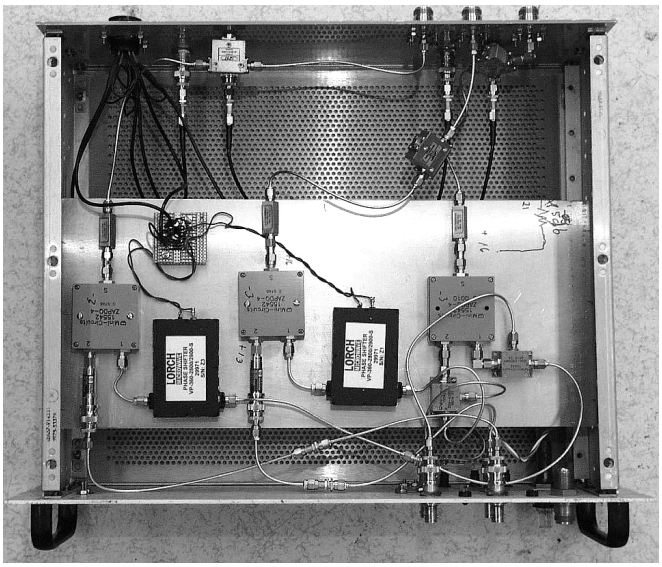


Figure 2. *RF chassis for two Klystrons*

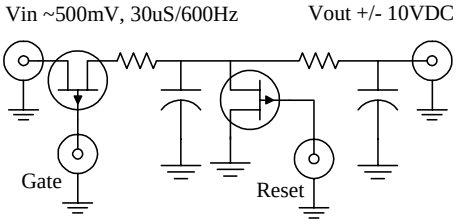


Figure 3. *Integrating Sample and Hold*

within a pulse. Shown on the following page as Figure 4 is the front view of the RF detector chassis, integrators and signal fan-outs.

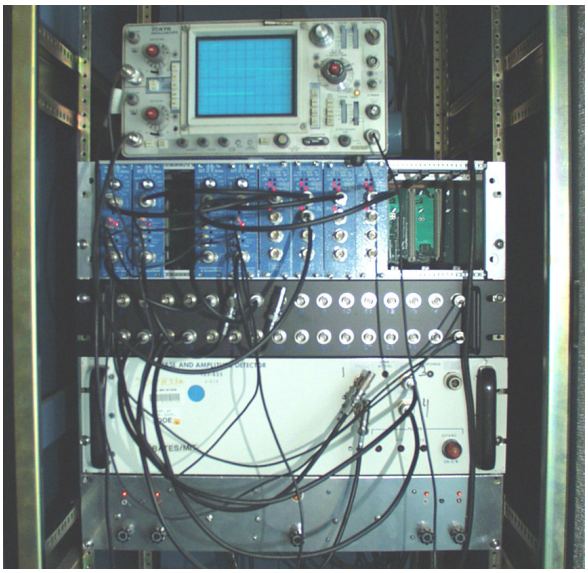


Figure 4. *Front view including integrators*

Specifications for the hardware are long term stability of displayed +/- 0.25º phase and +/- 0.1 Amps amplitude. Not shown is the VME crate that houses the ADC and timing board. These boards share a crate with other systems as is common. The integrator output is directly patched to the ADC via a custom BNC to ribbon cable interface board. The specifications were exceeded with the help of continuous calibration and special filtering as described in the software section.

III. SOFTWARE

The software of the PAM systems is comprised of a database and a sequencer program. The software functionality is described as follows.

1. Phase calibration

Due to the use of passive mixers gain errors in the phase signal exist as the amplitude of the klystron signal is changed or drifts. To null these errors a full 360º phase sweep is performed at timed intervals. The sequencer program initiates a 0 to 10V DAC ramp that causes the phase shifter to swing the integrated phase signal thru a major part of the ADC input. Performing an arcSine with respect to peaks of the resulting sinusoid (the mixer peaks when differential phase of the two inputs are 0º and 180º) adequately nulls all common errors including RF cable temperature differences, chassis temperature drifts, and signal amplitudes. This calibration scheme eases the complexity and cost of temperature controlling the entire RF cable run and RF components. Calibration also extracts the slope of the phase shift, ensuring that drift are indicated in the right sign.

2. Zeroing

The end user of the PAM systems output is a machine operator. They are interested in drifts from a setting that earlier in the day resulted in good beam. Therefore they are interested in changes, not absolute data. The sequencer program allows for both phase and amplitude zeroing. The resultant variables are displayed as both strip charts and bar charts, with the former excellent at showing trends, and the latter useful as an aid in returning the phase or amplitude of an individual Klystron to its previously known good “zeroed” value. These are shown below, and on the following page in Figure 7. Note the user has the ability to select specific klystrons for either phase or amplitude zeroing. Amplitude zeroing is simply the change from the last saved value, while phase zeroing causes the sequencer to proportionally change the phase shifter voltage until the ADC reads back the 0º point in the arcsine.

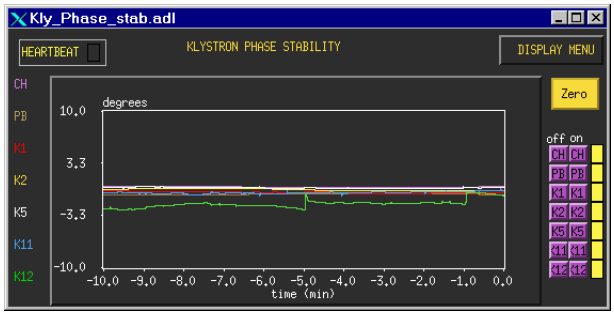


Figure 5. *PAM phase drift display*

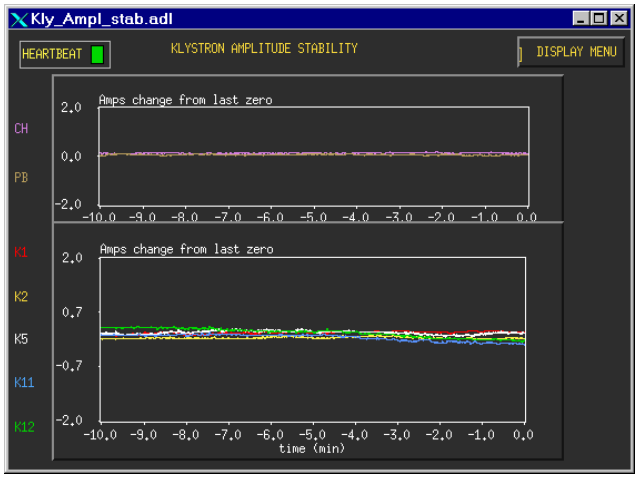


Figure 6. *PAM amplitude drift display*

3. Glitch handling

Due to RF noise burst pickups a software glitch filter was necessary to meet these specifications as analog low pass filtering at the ADC and software smoothing proved

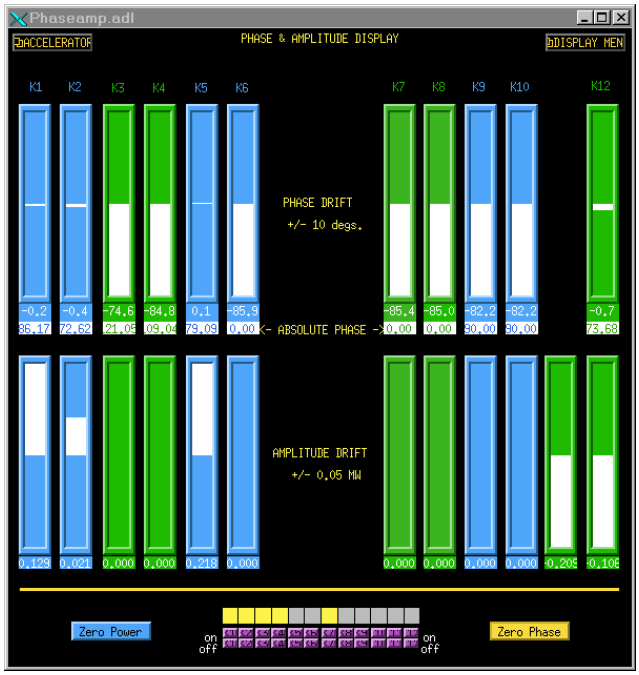


Figure 7. *PAM bar chart*

unable to adequately eliminate the error contributed by the RF transients. The glitch filter is comprised of three additional records in the database, two "calc" and an "ao". The filter "histrograms" in the sense that it replaces the ADC raw value with the previous if it exceed a "glitch" threshold, but passes the value if it is under a "change" threshold. Software smoothing (moving average, as "SMOO" in an "AI" record) must be done in an additional subsequent "calc" record. The performance is shown below in Figure 8. The glitches are seen as >10 count events and are never greater than one acquisition long. The upper trace is the output of the glitch filter with a "glitch" setting of 5, and a "change" setting of 3. The smooth line is the output of the final "calc" record, with smoothing set to 9 old values and 1 new value (0.9).

IV. CONCLUSIONS

The PAM system has been installed and operational over the last 3 ½ years in various forms. The current system has proven to have the resolution and stability to allow for the accurate monitoring of phase and amplitude drifts. Operationally the PAM system allows an operator to re-adjust either the amplitude or phase of the commonly lone drifting klystron rather than blindly re-tuning the beam via sequentially re-phasing all operating klystrons.

Over the past year PAM has proved enormously useful for catching step discontinuities in the unregulated high voltage applied to the klystrons, for monitoring and correcting phase drift of one or more klystrons and in establishing a clear correlation between a slow 3° phase drift on all klystrons and the performance of the primary

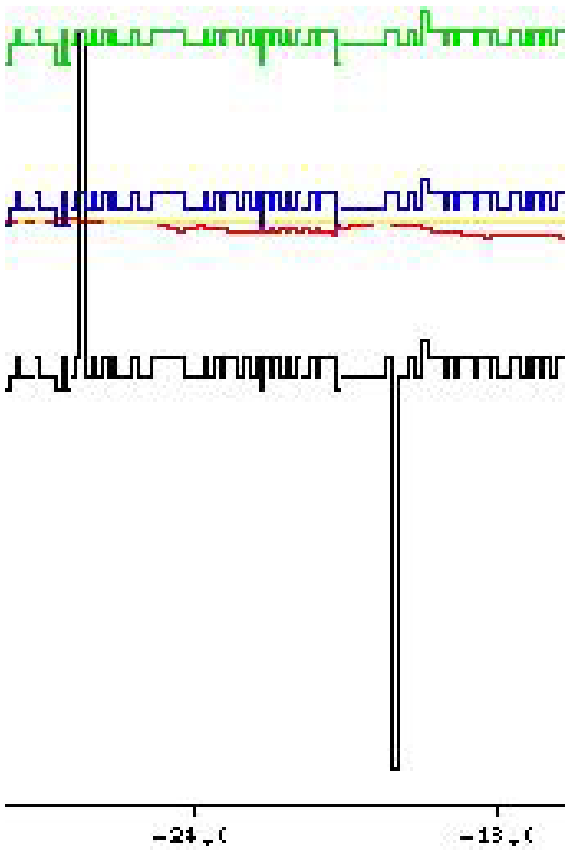


Figure 8. *Glitch handling.*

water cooling system. This PAM information is presented to the operations group and accelerator scientists through the usual set of EPICS tools, MEDM strip charts, bar charts and archived data. This information has enabled the operations group to much more easily set and restore the phasing of the machine. It has allowed them to more effectively troubleshoot malfunctioning components of the RF system.

Future work at Bates for RF PAM systems will include implementing the EPICS alarm handler. The addition of 11 new digitally controlled phase shifters (one is already in use) will allow automatic phasing of the machine and potentially closed loop control of the RF phases for further suppression of drift and improved machine performance.

V. REFERENCES

1. RF Detector Chassis Components

[1] Mini-circuits ZAPDQ-4	Splitter
[2] Hewlett Packard 8473D	Detector diode
[3] Mini-circuits ZEM-4300MH	Mixer
[4] Lorch VP-360-2800	Phase shifter
Lorch 5BP7-2850/100-S	B.P. filter
S.M.T. SM22052	Isolator

WEAP013
physics/0111022

THE SNS CRYOGENIC CONTROL SYSTEM: EXPERIENCES IN COLLABORATION [1]

W. H. Strong, P. A. Gurd, SNS ORNL, Oak Ridge, TN 37830 USA
J. D. Creel, B. S. Bevins, TJNAF, Newport News, VA 23606 USA

Abstract

The cryogenic system for the Spallation Neutron Source (SNS) is designed by Jefferson Laboratory (JLab) personnel and is based on the existing JLab facility. Our task is to use the JLab control system design [2] as much as practical while remaining consistent with SNS control system standards. Some aspects of the systems are very similar, including equipment to be controlled, the need for PID loops and automatic sequences, and the use of EPICS. There are differences in device naming, system hardware, and software tools. The cryogenic system is the first SNS system to be developed using SNS standards. This paper reports on our experiences in integrating the new and the old.

1 PROJECT DESCRIPTION

The SNS project is a partnership involving six DOE national laboratories: Argonne, Brookhaven, Jefferson, Lawrence Berkeley, Los Alamos, and Oak Ridge. The project is described in more detail on the SNS website and elsewhere in these proceedings [3, 4].

The cryogenic system provides the refrigeration necessary to supply the 2.1° Kelvin liquid helium required for operation of the SNS accelerator superconducting cavities. Once the cryogenic refrigerator system is placed in service, it is expected to run continuously. Shutting down the cryogenic system is very expensive due to the loss of beam time and helium. Cryogenic refrigerator systems operate best with a fairly constant load. There are very few independent control loops; a change to any one control variable affects the entire system. The Cryogenic Control System (CCS) provides the coordinated control algorithms to support the complex operation of the cryogenic refrigeration system.

2 CONTROL SYSTEM OVERVIEW

The top-level Integrated Control System (ICS) for the entire SNS project is based on the Experimental Physics and Industrial Control System (EPICS) suite of monitoring and control software. For the cryogenic systems, EPICS provides the operator interface for control and monitoring, alarming and archiving of selected signals, and routines to analyze real-time and historic data. All high level control algorithms, control loops and automatic sequences reside in EPICS. EPICS routines constantly monitor the state of the cryogenic system and update control outputs to keep the refrigerator system stable and the load constant. Programmable Logic Controllers (PLC) are used to interface signals from sensors and actuators to the EPICS Input/Output Controllers (IOC). PLCs perform low level interlocks, control loops and sequences.

3 DESIGN COLLABORATION

JLab is designing and procuring the cryogenic plant and cryomodules including all mechanical equipment, instrumentation, and control devices. The control system at JLab uses EPICS. The SNS CCS duplicates the JLab system as much as practicable.

The SNS Integrated Control System Group at the Oak Ridge National Laboratory (ORNL) is producing the CCS from the sensors and actuators up. This includes cables, signal conditioning, PLCs, VME based IOCs, boot servers, operator interface workstations, network hardware, and all control room equipment.

Los Alamos National Laboratory (LANL) provides general EPICS support. LANL also provides the RF control system for the cryomodules and the RF signals needed by the CCS for use in heater control.

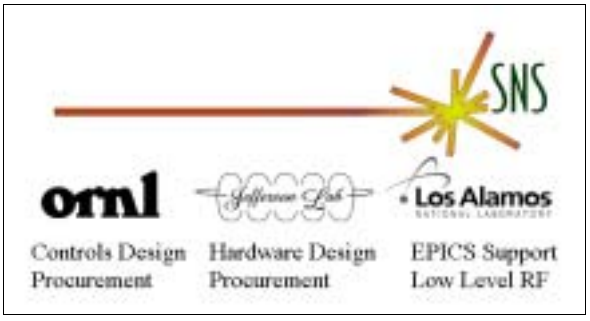


Figure 1: Collaboration on the SNS Cryogenic System.

Due to advances in hardware and software, there are differences in the CCS implementation between JLab and SNS. JLab has used their experience with an operating cryogenic system to provide invaluable

advice and guidance in the selection of hardware and software components for use by SNS.

4 HARDWARE

4.1 Cryogenic Control Room Hardware

The SNS Control Room will be used for control of all accelerator systems. A small local control room will be included in the Central Helium Liquefier building for operation of the cryogenic system. It will house the PC-based computer workstations, network hardware, and dedicated instrumentation for efficient control of the cryogenic refrigerator system. Two workstations will be used for operator interface functions and IOC boot servers. These two machines will run Linux and contain all source code and libraries necessary to operate and maintain the CCS in the event that the communication link to the SNS integrated control system server is lost. They will be configured such that if the primary boot server fails, the backup can assume the role of boot server with minimal operator intervention. These machines will provide the resources required for archiving, alarming and coordination of system control among the 10 cryogenic system IOCs. Each workstation will have 3 monitors for operator use.

A third Linux workstation with a single monitor will be provided for engineering and software development activities. These workstations will be identical so that they are interchangeable.

A Windows based workstation with a single monitor will be provided for PLC software development and maintenance.

While the hardware is different, this control room configuration is functionally very similar to the cryogenic control room at JLab. The design is based on the existing JLab cryogenic control room with updates to incorporate improvements suggested by JLab control system experts.

4.2 Cryogenic Control System Network

The SNS ICS network is divided into several subnets. The CCS subnet is separated from and independent of the remainder of the network. It includes redundant central switches and interconnecting fiber optic cables [5]. Experience at JLab determined that separation of the CCS network from the remainder of the network was necessary to provide the high availability required of the CCS.

Some heat load parameters needed for optimum operation of the refrigerator system originate outside the CCS subnet in the Low Level RF (LLRF) subsystem. If communication with LLRF is lost, the CCS inhibits beam operation and automatically takes the facility to a safe operational state.

4.3 Signal Interface Hardware

The major differences between the JLab and SNS CCS are in the interface hardware. VME/VXI crates with a MVME2101 power PC are used for the SNS IOCs. Custom utility modules provide crate information such as power supply status, crate temperature, etc. Custom timing modules are used to time synchronize all IOCs to a master clock so that all data can be time correlated. Some VME input modules are used for special sensors:

- Highland Technology V460 for silicon diode cryogenic temperature sensors
- Highland Technology V550 for linear variable differential transformer (LVDT) position sensors

Where possible, SNS uses the VME version of JLab CAMAC modules. Since no VME version of the helium liquid level module is available, custom signal conditioning is required at SNS.

Allen-Bradley ControlLogix PLC standard input and output modules are used to interface most sensors and actuators that require analog and discrete signals.

4.4 Local Control Hardware

Local control must be provided for some process valves so that the cryogenic system can continue to operate in the event of loss of an IOC. For the JLab CCS, local control is provided by custom valve control modules in the CAMAC crates. The SNS ICS provides Allen-Bradley PanelView terminals directly connected to the PLCs to fulfill this requirement. The PanelView at SNS provides additional capability by displaying many of the process parameters that would be used in the valve control loops in the IOC.

5 SOFTWARE

SNS global controls have mandated a number of software standards to be used in all components. These include the use of EPICS, device naming conventions, the use of the Oracle database and operator screen color conventions.

5.2 EPICS

The use of EPICS makes available a large body of software and support. Since JLab also uses EPICS, some JLab software such as the LVDT device driver (used for position measurements) can be used directly. Other software can be substantially copied; the V460 device driver (used to measure cryogenic temperatures) was adapted from the LVDT driver. The CPID record, used extensively in the CCS, is a part of EPICS base created by JLab for their control system. EPICS facilitates the SNS collaboration; SNS collaborators at Los Alamos National Laboratory developed the

Ethernet Industrial Protocol driver that provides the communication link between the IOC and the PLC.

5.3 Device Naming and Databases

A common naming scheme is crucial to integrate signals among the systems and collaborating laboratories. SNS cryogenic controls personnel have used a Microsoft Access application as a step toward the development of a project-wide database. The lessons learned in the process of creating EPICS databases from this application are being incorporated into the project-wide ORACLE technical database. Examples of all cryogenic system device types have been included in the test database. So far, the records created have very few links to other records. Graphical tools will be required to create and maintain highly linked databases, for example those involving calculations or PID loops. Because many of the cryogenic signals and calculations are adapted from JLab, test versions have been created using the same tool that JLab uses, CapFast. A general SNS solution to this problem is still required. Alarms and archive configuration files will be produced from the project-wide database. In the interim, these must be produced from our local database.

5.4 Display Manager

SNS has selected the Extensible Display Manager (EDM), which is maintained at ORNL. A script was created to translate the JLab operator screens, which run under MEDM, into EDM. After the screens were translated, the SNS color and font standards were applied. The screens were updated to reflect the SNS plant hardware design and incorporate improvements suggested by JLab. Finally, the JLab device names were replaced by the SNS standard device names. Although the final versions are substantially different from the original JLab screens, this process provided a good starting point for much of the operator interface. This process has worked for the main compressor screen and first and second stage compressor screens. Some features have been added to EDM for SNS: symbolic colors, symbolic font declarations, undo and a striptool widget, as well as widgets to display parts of SNS-standard names.

5.5 Control Algorithms

Both SNS and JLab use EPICS or routines accessed through EPICS for all high-level control loops and algorithms. An example is the cavity heater control algorithm.

A cryogenic RF cavity operates at a constant pressure. Cryogenic refrigerator systems operate best with a constant load. The flow, temperature, and pressure in the common primary return line from the cryogenic cavities are factors in the load. The return pressure, which must be held constant for proper RF cavity operation, is closely related to the heat absorbed by the helium liquid in each of the 81 superconducting cavities and the return flow. One of the major dynamic heat sources is RF loss into the cavity. When RF to a cavity is turned off, power to a resistance heater in the cavity is increased to keep the heat load constant. All cavities share a common return line, and a change in one cavity affects all other cavities to some extent. If too much total heat is applied, the return pressure will increase and the return flow will try to increase. The return flow control loop will react to keep the flow constant. Coordination of all heaters is required to keep the total heat load and return flow balanced. An algorithm running on the boot server in the cryogenic control room provides the coordination of heater control across all cavities. JLab, LANL, and SNS personnel are working together to develop this control algorithm and provide all required input parameters.

6 SUMMARY

Collaboration among several partners over great distances is working. The SNS project benefits by having access to the experience of the personnel at the partner laboratories.

REFERENCES

[1] Work supported by the US Department of Energy under contract DE-AC05-00OR22725
[2] Marie S. Keesee and Brian S. Bevins, "The CEBAF Control System for the CHL", Advances in Cryogenic Engineering, Vol. 41, pp. 649-654, Plenum Press, New York, 1996
[3] SNS Website http://www.sns.gov/
[4] Dave Gurd, TUDT002 “Management of a Large, Distributed Control System Development Project”, ICALEPCS 2001
[5] Bill DeVan, TUAP055 “Plans for the Spallation Neutron Source Integrated Control System Network”, ICALEPCS 2001

SYNCHROTRON CONTROL SYSTEM OF HIMAC

E. Takada, NIRS, Chiba, 263-8555, Japan, e-mail: takada@nirs.go.jp
T. Kondo, AEC, Chiba, 263-0014, Japan

Abstract

HIMAC, Heavy Ion Medical Accelerator in Chiba, has been in operation for about 8 years. The control system of HIMAC synchrotron is now undergoing renewal and upgrade. As a medical accelerator in use, reliability and availability of the beam must not be compromised in HIMAC. We decided to change most computers of the system in a node-by-node way, over a couple of years. Among the replaced are: Power-supply Controllers, RF-system controller, Beam Transport, and Timing System controllers. The steps taken and to be taken are reported.

1 INTRODUCTION

HIMAC accelerator complex has successfully provided carbon beam for clinical study on the cancer treatment since June 1994; cumulated number of patients treated by irradiation of HIMAC carbon beam exceeds one thousand in July, 2001 [1]. It leads to the planning and realization of a few particle therapy facilities in Japan [2].

HIMAC accelerator complex has been required to supply ion beams, not only for an ion therapy, but also for wider range of biological and physics experiment using various ions from H to Xe. As such, synchrotron control system has to meet needs of developing and tuning new beams, which becomes harder and more complicated than the original scheme had assumed. Initial design of the system was reported in the earlier conference of the series [3].

2 HARDWARE REPLACEMENTS

The figure on the last page of the report shows the present configuration of the HIMAC synchrotron control system. Relevant elements are described in the following.

2.1 Operation Terminals with Touch Panel

As reported in the last conference for injector linac system, the terminals in synchrotron control system were also replaced by PC's, and their performance is satisfactory [4]. (The problem mentioned there turned out to be a bug in OS.)

2.2 Power supply Controllers

Power supplies for synchrotron magnets were grouped and controlled by VME-bus board-computers for trapezoidal pattern cycle. The cpu board of the controllers had been VME147 and were replaced by VME162 in a similar situation as in the operation terminals. It is also planned to upgrade further with a Power PC family board.

The installation of DPI/O and DTO boards [5] are now under way.

2.3 PLC for power supply status control

Operational status (ON/OFF, Faults, and interlocks) of power supplies were controlled and monitored by a dedicated PLC, HIDIC-S10α, made by HITACHI. We have added a gateway PC in order to provide an additional access to the PLC from the independent system other than the original control main computer.

2.4 RF pattern memory control

The RF acceleration voltage, frequency curve, and ferrite-bias current must be controlled in accordance with the main magnet excitation. Another VME computer was prepared to facilitate the function, and was called RF pattern memory. It was, however, connected via GPIB to the FA computer (Toshiba, G200E), which then communicated with the main computer of the system, and took care of RF power and ring vacuum devices as well, in the beginning. At the occasion of the replacement, we have separated the RF pattern memory and connected it directly to the synchrotron control LAN.

2.5 PLC for RF power and vacuum control

As mentioned above, event-critical part of the function of the FA computer was separated and the rest of the functions can be carried out by a PLC reasonably. Therefore, the former was replaced by the PLC with LAN interface.

2.6 PLC for Beam lines into and out of the ring

Beam transport lines of injection and extraction regions were also controlled by the FA computer as for

RF system. Although the number of I/O points is larger in BT case and BPM data processing demands faster cycle than, e.g., vacuum data monitoring, it is still the case that PLC can perform the control function. Thus the replacement is under way for both upper and lower rings.

3 ADDITION OF NEW CONTROLLERS

There are devices that have not been foreseen but added after the commission of the machine. RF-KO system for beam intensity control, and Electron Cooling system (for the lower ring only) are among them.

3.1 Electron Cooling System

At present, the electron cooler is installed and used for accelerator study in the lower ring. Its control system is independent from the main system to avoid possible interference with routine therapy operation. However, once the EC operation procedure is established and incorporated into future treatment irradiation, the control system will also be merged into the main system.

3.2 Beam Intensity Controller

Beam intensity control by applying lateral RFKO field at injection energy level as well as flat top has been developed and constructed for both therapeutical respiration gating and experimental study of radiobiological irradiation. The system will replace the present RFKO system for respiration-synchronizing gate.

4 SOFTWARE ENHANCEMENT

Synchrotron rings deliver carbon beams for cancer therapy, at present, of 290, 350, and 400 MeV/u. In order to help increase total throughput of treatment, and to help reduce the load of daily operation, a few enhancement of operational tools has been done.

4.1 Energy Changing Sequence

Accumulated experience established the energy changing procedure for treatment condition where the reproducibility of beam is well observed. The original procedure provided many adjustable parameters and took time and cautious tuning on the part of operators. With the adoptation of established procedure into the control sequence, the time needed for the change has become about 5 minutes, reduced to nearly a quarter of the previous condition. An extension of similar trimming of the procedure for injection part is now in preparation.

4.2 Demagetization Sequence

In the COD correction, excitation of the steering magnets (STV and STH, for short) may differ with different beams, such as for the therapy use beam and one for cooling study. The residual magnetic field of the STVs, e,g., however, can be problematic since it causes occasional intensity decrease. It was nullified by de-magnetizing the STVs as expected, but the procedure was tedious and time-consuming. Therefore, the procedure was made into the sequence from the main computer, and an operator can now conduct it by simple cell-touch. The enhancement contributed to keep the beam intensity level for treatment with comfortable margin.

4.3 Parameter-Editing Station

Operational parameters were stored in the database, which was supposed to be accessible only from the control stations. As the machine was providing beams to the users continuously, it was found that the preparation and/or review of parameters through the operator console is inconvenient. Therefore, an addition of extra workstation for parameter editing and relevant tools was done. Since the tools were basically common to the existing one at console, and with handy enhancement such as comparing two files of any rings, it proved fruitful for operational improvement.

5 FARTHER PLAN

The replacement of present main computer, VAX4000/300, has been considered, both because of increased load from new devices and operations and because of aging hardware. To succeed existing software application with least extra work and most reliability, we picked an Alpha system of the same manufacturer as for the old one. Performance comparison of the new and old system was carried out. Typical result shows the following (for starting-up of the application system memory):

Alpha XP900	3.46 sec. (average)
VAX4000/300	156.35 sec. (average)

The result suggests a major improvement in terms of the response time of the system to the operator action. We now plan to replace the old VAX4000 by the new machine in next March.

The static VAR compensator (SVC) will need closer control to use cooled beam, which is under study. The backbone for the synchrotron control should be upgraded for faster multi-path one from the present 10BASE5.

6 ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the colleagues of Accelerator Physics & Engineering

Department of NIRS, and to the fellow crew members of Accelerator Engineering Corporation.

REFERENCES

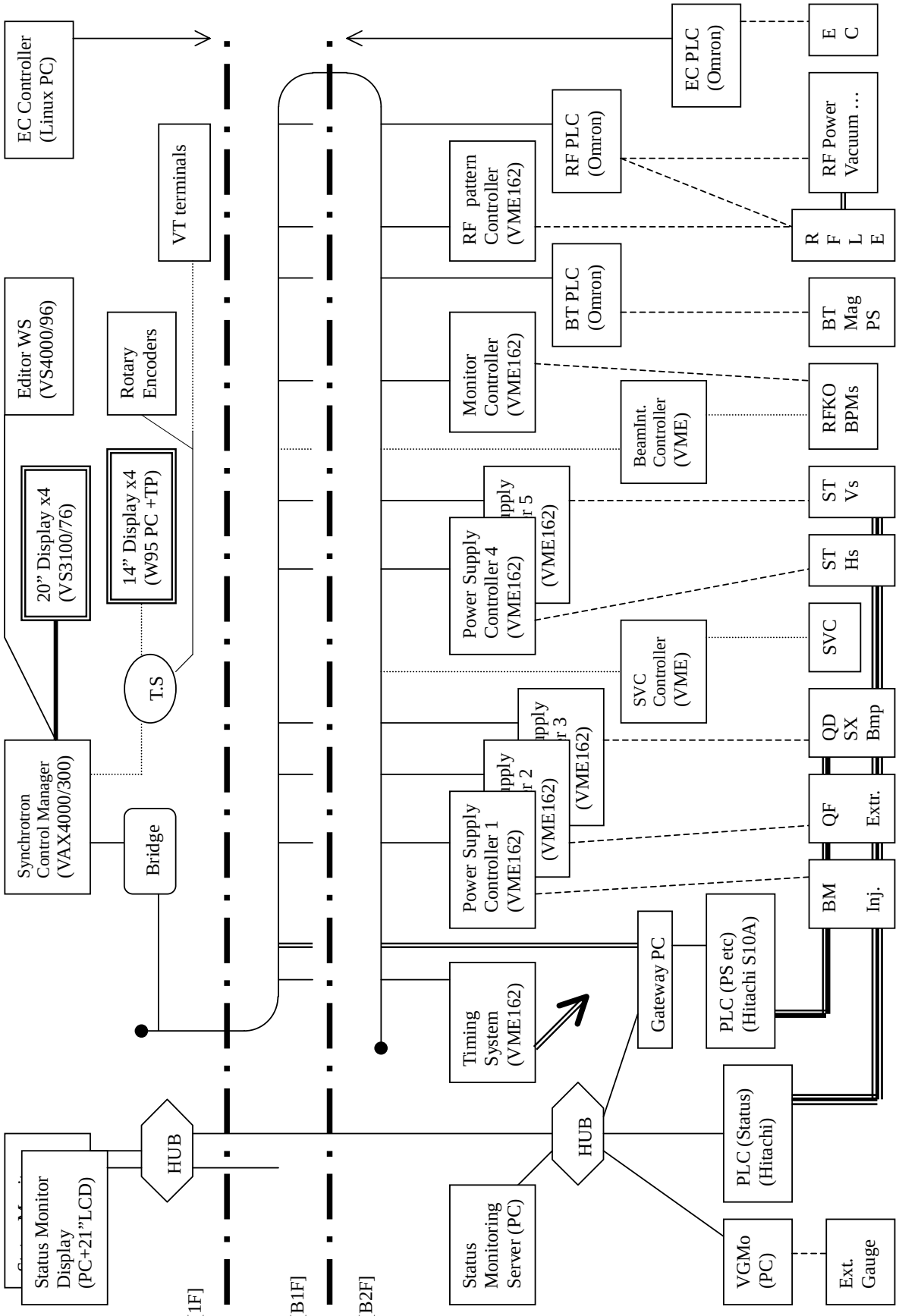
[1] E. Takada *et al*., "Present Status of HIMAC", pp. 187-189, Proc. of SAST'01, Osaka, October 2001. C. Kobayashi et al., "Operation of HIMAC and cancer therapy", pp. 214-219, Proc. of WAO2001, CERN-2001-002.

[2] S. Yamada *et al*., "HIMAC and Medical Accelerator Projects in Japan", APAC'01, Beijing, September 2001.

[3] E. Takada *et al*., "The HIMAC synchrotron control system", Nucl. Instr. & Meth. A352, 40-42 (1994).

[4] E. Takada *et al*., "Present status of control systems of HIMAC accelerator complex", pp. 683-685, Proc. of ICALEPCS'99, Trieste, 1999.

[5] N. Araki *et al*., "Design of a Synchrotron Control System with DPO Series for Advanced Therapy Operations", T3B-6, ID157, Proc. of ICALEPCS'97, Beijing, 1997.


EC Controller (Linux PC)
VT terminals
Editor WS (VS4000/96)
Rotary Encoders
20" Display x4 (VS3100/76)
14" Display x4 (W95 PC +TP)
T.S
Synchrotron Control Manager (VAX4000/300)
Bridge
HUB
Status Monitor Display (PC+21"LCD)
[1F]
[B1F]
[B2F]
Timing System (VME162)
Gateway PC
PLC (PS etc) (Hitachi S10A)
Power Supply Controller 1 (VME162)
Power Supply Controller 2 (VME162)
Power Supply Controller 3 (VME162)
Power Supply Controller 4 (VME162)
Power Supply Controller 5 (VME162)
SVC Controller (VME)
BeamInt. Controller (VME)
Monitor Controller (VME162)
RF pattern Controller (VME162)
RF PLC (Omron)
EC PLC (Omron)
BT PLC (Omron)
BM
QF
QD
SX
Bmp
Inj.
Extr.
SVC
ST Hs
ST Vs
RFKO BPMs
BT Mag PS
RF Power Vacuum ...
R F L E
E C
Status Monitoring Server (PC)
PLC (Status) (Hitachi)
VGMo (PC)
Ext. Gauge

WEAP017
physics/0111154

EVOLUTION OF THE SPS POWER CONVERTER CONTROLS TOWARDS THE LHC ERA

J.C.L. Brazier, Brazier Systems & Consultants Ltd, UK
A. Dinius, CERN, Geneva, Switzerland, P. Semanaz, B & S International, France

Abstract

By the end of the nineties, the power converter control system (Mugef) of the CERN proton accelerator (SPS) had undergone a complete modernization. This resulted in newly developed hardware for function generation, measurement and I/O in a VME environment, under the LynxOS real-time operating system. This has provided a platform on which extensions can be developed for future operation in the Large Hadron Collider (LHC) era. This paper describes some of these extensions, in particular a fast Surveillance and Interlock system for monitoring the power converter output currents. This will be mandatory for the safe operation of the SPS transfer lines TI2 & TI8 to LHC and for similar applications in the future. The strategies employed to cope with various failure modes of the power converters and the timely activation of the interlock are outlined. The new SPS controls infrastructure now under development, will give rise to new modes of operation for the Mugef systems. Integration with the proposed middleware must be undertaken in a structured evolution, while retaining compatibility with the current usage.

1 BACKGROUND

Magnet circuits in CERN's SPS are driven by power converters. These power converters are controlled by equipment known as a Mugef [1] that:

- Turns the power converter on and off and reads back its state.
- Using an analogue to digital converter (ADC), digitises the analogue voltage (±10V) from the output current transducer (DCCT) every millisecond.
- Produces a (±10V) reference voltage for the power converter to generate the magnet current, using a digital to analogue converter (DAC). The output of the DAC follows a pre-loaded function.

One Mugef crate can control up to 64 power converters and connects to the control system via a proprietary command/response middleware (SL-Equip).

1.1 Mugef Description

The present Mugef is described in [1]. It consists of a VME chassis with a PowerPC-603e. The controlling software runs under the LynxOS real time operating system. Special purpose cards perform the reference voltage function generation and the analogue acquisition of the reference output and DCCT output. A single card provides on/off/reset and status read-back for all the power converters. A CERN standard timing card provides synchronisation and SPS machine timing events. A System Arbiter Controller (SAC) card completes the hardware.

The controlling software consists of a set of processes connected with message queues and shared memory segments. There is a process controlling each of the hardware elements (function handling, acquisition, status control, etc.) which includes the hardware driving, as all the cards are directly mapped into the process' memory. A timing process translates machine timing events into internal action triggers. These are interpreted by an action sequencer that maintains a list of actions (loading functions, doing measurements etc.) Finally, there is a process that accepts commands from the rest of the control system and initiates immediate actions ('*Now*'). This software is general purpose and has been ported to other applications within the SPS control system [2].

1.2 Operational Experience

Some 24 Mugef systems were brought into operation in April 1998 and have worked well since. Reliability has been good, with several months between reboots.

One of the initial design criteria was that the system should be amenable to extensions and modifications. Several of these have been absorbed successfully, most notably, the ability to accept dynamic corrections to reference output function data. This was included for experiments in closed-loop control of the beam tune [3]. This entailed receiving a stream of correction data via UDP over a private network from beam position equipment around the ring. These presaged the closed-loop control needed for LHC.

The following chapters will detail two crucial extensions, namely a dynamic surveillance system

(known as Channel-64) and the modification of the controlling software to mate with the proposed SPS control system middleware.

2 CHANNEL-64

Surveillance of the power converters in operation has been, in the most part, the province of the control system application programs. These perform both occasional analogue measurements of the output current and repeated requests for power converter status. Whilst this is adequate for many cases, particular circumstances such as beam ejection require rigorous testing of the magnet current at the time of ejection, on a cycle-by-cycle basis. Here it is necessary to ensure that all the critical power converters in the transfer line are working *within their expected tolerance* just before the beam is extracted. Implicitly, this also means that the power converter is not in fault and therefore a status check is not needed. If any of the critical tolerances are exceeded, then extraction must be prevented, otherwise material damage to the vacuum chamber and magnets could occur.

2.1 Channel-6 Heritage

The Channel-6 system, (named after the 6th interlock chain in the SPS control room), was designed in the mid 80's, using discrete analogue electronic components to provide the above functionality. A total of 18 power converters had to be monitored in two buildings. Remote control of the Channel-6 system was done via standard SPS multiplex and associated software.

The system verified that the monitored power converters had reached their nominal current 20 ms before the extraction should take place. On a per power converter basis, two comparisons using analogue comparators were made, just prior to extraction. Using the timing system to trigger the process, the DCCT output signal was compared with a reference voltage supplied by the Mugef system. Any difference exceeding the set tolerance then opens an interlock chain, thus disabling the extraction kicker magnet power converter. This prevents extraction of the beam.

Since this hardware was reaching its end of life and the same functionality is needed for the LHC transfer lines, the opportunity existed to provide the same functionality, using an extension of the Mugef system. Specifically, advantage could be taken of the fact that:

- The Mugef could digitally compare the actual ADC output values with the theoretical function values, rather than using a hardware-synthesized reference and analogue comparison.
- The analogue acquisition memory already contains data, at the milli-second rate, prior to the comparison.
- A choice of which combination of power converters to include is fully variable up to all 64 available.
- Different comparison algorithms can be used in different circumstances.
- The precision of the comparison can be increased from 2% to 0.1%

These advantages lead to a very cost effective solution, both for upgrading the original Channel-6 equipment and also for the new transfer lines to LHC.

With any safety/interlock system, great care needs to be taken to ensure the integrity of the system. Permissioning must be positive, ie. the system has to enable the output only after each good comparison. The design must ensure that a previous GO situation is not erroneously replicated to succeeding cycles. This is taken care of by both software and hardware clearing the permission at the start of each machine cycle.

2.3 Hardware Needed.

No additional hardware is needed if the power converters monitored by the Channel-64 system are only sending a message to the Central Alarm Server (CAS) when the set tolerances are not met. However for beam extraction, some additional hardware is needed. However, instead of adding an I/O board, the existing SAC board is used for performing the necessary I/O tasks for the Channel-64 hardware. Since the internal memory on the SAC board is not used, a small daughter board has been developed to replace the memory chips. Outputs are connected via a flat cable to the Channel-64 display and interlock chain chassis. A maximum of sixteen individual interlock chain chassis can be driven from the SAC board.

2.4 Software Additions

The surveillance is triggered by a machine event occurring >= 10 mS prior to the required output action. Typically, some 3-10 mS of earlier data has been used. Using a tolerance table, pre-loaded with allowable values for each selected channel, a comparison is made with the measured data. Timing test have revealed that the entire process takes approximately 4 mS for all 64 channels.

A result bit-mask is returned of all the channels out of tolerance. This is sent to the hardware in the form of go/no-go and the appropriate output is/is-not enabled.

2.4.1 Data Filtering Strategy Routines

The original Channel-6 system simply averaged over time and compared the result with the tolerance using analog comparators. Whilst this is acceptable with stable ADC values, it will give erroneous results on a

ramp. In this case, point by point comparison with the reference may be better.

In certain cases, power converter ripple and noise may be a problem. Simple averaging will mask the noise, whereas point comparisons will not. A variation to the point comparison strategy is not to allow single spikes to affect the result. This is accomplished using a median filter. Ripple outside a given tolerance could also be detected by yet another data treatment.

Because different strategies are valid in different circumstances, a mechanism has been incorporated to allow different *data treatment subroutines* to be incorporated with the minimum of effort.

Preliminary tests have proven the validity of the system using the strategies outlined above. Detailed results can be found in [4].

2.5 Software Only Solutions

Since the software is available on all the Mugef systems, regardless of whether they have the attached hardware, this becomes a useful tool for monitoring power converters in operation. To this end, several new commands have been added to allow the surveillance results to be logged or read directly.

2.5.1 Alternative To on/off Status Checking

In general, status checking only confirms that the required power converters are switched on. It does not confirm that their output current is within tolerance. By comparison the new surveillance provided by Channel-64 implicitly provides *all* this information. Therefore, many application programs could be simplified/speeded up by augmenting traditional status checking with this new method.

3 NEW SPS CONTROL SYSTEM

The current power converter application programs interact with Mugef via an infrastructure based on the SL-Equip call, a simple, single-user, synchronous, command response protocol. This may be superseded in the near future by a new middleware-based control software, with which the Mugef must necessarily integrate.

This middleware, is part of a project to upgrade of the SPS software [5] to turn the SPS into a multi-cycling injector for LHC. This works by defining management *contracts* for cycle, settings and status control. A *'Device Server'* in the equipment subscribes to as many contracts as it can implement. Callback routines in the Device Server are activated as part of the contract, for example on the addition/removal of new machine cycles.

The new control system and device servers are being developed now but, will have to be implemented on a phased basis. During this transition, both the new and old models of control will have to co-exist. To enable this, a 'Bridge Server' is being constructed which will respond to the new middleware calls and generate the appropriate SL-Equip calls to the Mugef. Initially, this will implement the '*state contract'*, which allows a power converter to be turned on and off. Other contracts, like 'settings' and 'cycle', will be added in due course.

When all the legacy applications have disappeared, the bridge can be disposed of and its functionality absorbed into the basic structure. Clearly, this will be a major software change (approx. 40%) entailing a complete rewrite of the Mugef front end. However, the original design of the Mugef software allows such changes to be made with relative ease, as previous examples have shown.

4 CONCLUSION

The hardware and software employed in the Mugef was designed from the start to be flexible enough to incorporate changes, as requirements for power converter control evolved. During its lifetime, the whole SPS control system model will also change. This paper shows the system fulfilling such requirements and rising to the new challenges of the LHC era.

REFERENCES

[1] A.H. Dinius et al, "Evolution in Controls Methods for the SPS Power Converters", ICALEPCS'95, Chicago, Nov. 1995.

[2] J.C. de Vries, et al, "Real-time and control software for the new orbit measurement system of the CERN SPS, ICALEPCS'99, Trieste, Nov, 1999.

[3] A. Dinius, L. Jensen, P. Semanaz, "Measurement of the transfer function of the main SPS Quadrupoles, SL-Note-98-047, 20th July 1998

[4] A. Dinius, D. Hundzinger, "Tests préliminaires du système "Fast Interlock" pour les lignes de transfert TI2 et TI8", CERN Note, April 2001.

[5] M. Jonker et al, "Upgrading the SPS operational software for the LHC era: The SPS-2001 software project", ICALEPCS'99, Trieste, Nov. 1999.

WEAP022
physics/0111032

FRONT END CAMAC CONTROLLER FOR SLAC CONTROL SYSTEM

M. J. Browne, A. E. Gromme, Stanford Linear Accelerator Center, Stanford, CA 94025, USA
E. J. Siskind, NYCB Real-Time Computing, Inc., Lattingtown, NY 11560, USA

Abstract

Most of the devices in the SLAC control system are accessed via interface modules in ~450 CAMAC crates. Low-cost controllers in these crates communicate via a SLAC-proprietary bit-serial protocol with 77 satellite control computers ("micros") within the accelerator complex. A proposed upgrade replaces the existing Multibus-I implementation of the micro hardware with commercial-off-the-shelf ("COTS") personal computers. For increased reliability and ease of maintenance, these micros will move from their current electrically noisy and environmentally challenging sites to the control center's computer room, with only a stand-alone portion of each micro's CAMAC interface remaining in the micro's original location. This paper describes the hardware/software architecture of that intelligent front-end CAMAC controller and the accompanying fiber optic link board that connects it to the PC-based micro's PCI bus. Emphasis is placed on the hardware/software techniques employed to minimize real-time latency for pulse-to-pulse operations that control accelerator timing, acquire data for fast feedback loops, and change device settings to close those loops. The controller provides the sole interface between the COTS computing/networking environment and the existing CAMAC plant. It also supports higher bandwidth commercial byte-serial crate controllers and legacy BITBUS hardware.

1 INTRODUCTION

The SLAC control system currently utilizes a central database server connected via a proprietary broadband network to 77 front-end microcomputers or "micros" distributed throughout the accelerator complex. Each micro is equipped with a list-processing interface that uses a proprietary 5-megabit/second bit-serial protocol to access 1-4 strings of CAMAC crate controllers. The total number of CAMAC crates is around 450. Approximately 10% of the micros also have BITBUS masters that control PEP-II magnet power supplies.

The database server-to-micro network connection consists of three 1-megabit/second logical segments, with each segment carried in 6 MHz of bandwidth in one of two closed circuit TV ("CCTV") cables via RF modems. Message transfers via this network are asynchronous with the 360 Hz pulsed operation of the SLAC LINAC; the maximum packet length is far longer than the 2.78 millisecond inter-pulse period. Pulse-to-pulse sequencing of accelerator operation is controlled by a 360 Hz broadcast of 128 bits of "trigger pattern" information from the timing control micro. A second, beam-synchronous network carried in another 6 MHz CCTV cable channel distributes this broadcast. Each receiving micro forwards timing information for the next three beam pulses from within the trigger pattern to accelerator devices via an F(19) operation to all slots in all CAMAC crates containing timed devices. Low-latency point-to-point 2-megabit/second baseband links comprise a third, beam-synchronous network. These collect fast-feedback sensor data into a common micro for each feedback loop, and distribute commands to the micros controlling devices that close each loop.

The circa-1980 micro hardware architecture uses the Multibus-I backplane. (A small number of front-end VME/VXI systems that control PEP-II RF stations and other non-pulsed devices are beyond the scope of this article.) Although the processors in these systems have been upgraded to 80386s or 80486s, the performance and density increases in CPU, memory, I/O bus, and networking technology driven by Moore's Law over the past decade have bypassed these systems because of their reliance on an archaic hardware architecture. In addition, the proximity of the micros to high-power pulsed electrical systems has resulted in large amounts of noise pickup in backplanes and communications links, particularly from kickers and klystron modulators. Finally, the micros' remote locations expose them to extremes in temperature and humidity, as well as to large concentrations of airborne dust/dirt.

Preparations for replacing the micros with low-cost COTS personal computers have been underway for three years. The functions of all three existing front-end networks can be combined into a single modern network based on COTS technology such as switched gigabit Ethernet. Quality-of-service primitives within this COTS network ensure adequate real-time latency for beam-synchronous communications supporting trigger pattern dissemination and fast-feedback loops.

Deployment of COTS micros and networks requires the development of the hardware and software of an interface between the micro's PCI bus and the existing CAMAC (and BITBUS) accelerator hardware.

Reliability and maintainability considerations suggest locating the micros in the controlled environment of the control center's computer room. The interface hardware thus is realized as a stand-alone Front End CAMAC controller ("FECC") that is near the actual CAMAC plant and is connected via optical fibers to a PCI Link ("PCIL") board on the micro's PCI bus.

2 REAL-TIME CONSIDERATIONS

In the current architecture, sequences of CAMAC operations required by beam-asynchronous threads of execution in the micro are organized into "packages" that require no more than one millisecond for interface execution. Micro software prevents a new package from being passed to the hardware until processing of the previous package has been completed. Beam-synchronous software activated by 360 Hz trigger interrupts queues multiple packages to the hardware without awaiting the completion of previous packages. The maximum latency with which the synchronous software acquires the CAMAC interface after an interrupt is thus one millisecond out of the 2.78-millisecond inter-pulse period. The synchronous software executes the F(19) pulse-to-pulse timing data broadcast, acquires data for fast-feedback loops, and performs other data acquisition and control functions that must occur on a particular beam pulse in order to properly coordinate with actions in other micros on the same pulse or in the same micro on previous and subsequent pulses. Once fast-feedback loop input data are acquired, they are passed to sequences of loop-processing threads of execution within the same micro via inter-thread mailboxes, and to threads in other micros via the third network. These fast-feedback threads have high priority for use of the CPU in their respective micros, but must still spin-wait for the CAMAC interface to be idle before passing a loop-closing package to the hardware. If a lower priority beam-asynchronous thread is allowed to regain control of a micro's CPU while fast-feedback threads await data from another micro, an additional millisecond of latency may intervene before the CAMAC interface can be used to close the loop if the asynchronous thread requests the execution of its own package.

The FECC design addresses these concerns by including separate pipelines for beam-synchronous and beam-asynchronous operations in both the hardware and software. The processing of beam-synchronous operations take precedence whenever these two classes of operations compete for a common resource, and any ongoing asynchronous operation is preempted with a latency no greater than a few microseconds. This design requirement is imposed on the FECC's CAMAC cycle generation hardware, the PCIL's PCI block transfer hardware, the inter-board link, and all software executing on Analog Devices 40-MHz SHARC digital signal processors that add intelligence to both boards.

3 HARDWARE FEATURES

The first-generation hardware featured FECC support for four SLAC serial CAMAC cables, PCIL support for a 33 MHz/32 bit PCI, and separate half-duplex 125 megabit/second inter-board data links for beam-synchronous and beam-asynchronous messages. This design employed the Xilinx XC4000XLA family of field programmable gate arrays ("FPGAs"), with three FPGA parts in the PCIL and two in the FECC. The second-generation FECC2 supports four rings of IEEE-standard type L-2 CAMAC crate controllers and a BITBUS system in addition to the four strings of SLAC CAMAC crate controllers. The matching PCIL2 uses a 66 MHz/64 bit PCI, while the second-generation link uses gigabit Ethernet hardware components to support full-duplex 125 megabyte/second transfers. The PCIL2, employing a single XCV1000 FPGA, is currently being debugged. The FECC2's FPGA logic, now nearing design completion, is intended to reside in an XC2V3000. Both FECCs use CAMAC packaging for power, cooling, and mechanical support, but are unconnected to the CAMAC dataway. A PowerPC processor embedded in a Virtex-II FPGA replaces the SHARC in the planned third-generation PCIL3.

In addition to its on-chip memory complement of 64k 32-bit data words and 40k 48-bit program words, the FECC2's SHARC has 512k words of external zero wait state 32/48-bit data/program memory in six 12-nanosecond 4-megabit asynchronous static RAM chips, expandable to 2048k locations with 16-megabit parts. The PCIL2 and FECC2 each have a single 64-kilobyte boot PROM, and can download additional software from the host micro's memory. Operation of the CAMAC, link, and PCI I/O hardware employ Direct Memory Access ("DMA") operations. The combined I/O transfer rates in both boards exceed the capacity of the SHARC's bus and memory. I/O operations thus access special DMA memories attached to each board's FPGA. These employ 133 MHz synchronous static RAM parts operating with the link's 125 MHz clock. The PCIL2 uses one rank of two 32-bit wide parts to achieve 1000 megabytes/second of bandwidth; the FECC2 has four interleaved 32-bit parts providing 500 megabytes/second of bandwidth. Each current part contains one megabyte of storage, and can be upgraded to future 2-megabyte components. Cycles in these memories are allocated via a time-slicing algorithm. The PCIL2's memory allocates 9/16 of the cycles to the PCI interface, 2/16 to each of the link transmit and receive interfaces, and 3/16 to SHARC access to the memory. The FECC2's allocation is 4/16 to each of link transmit and receive, 4/16 to IEEE-standard

CAMAC, 1/16 to SLAC CAMAC, and 3/16 to the SHARC. The FECC2's BITBUS hardware has very low bandwidth and employs programmed I/O transfers from the SHARC rather than DMA hardware.

The PCIL's PCI interface performs block transfers at up to 533 megabytes/second, and has hardware support for accessing host memory buffers in paged virtual memory as well as in a linear address space. There are separate control registers for beam-synchronous and beam-asynchronous operations, with an independent pair of read/write data FIFOs for each class. Beam-synchronous transfers acquire the common cycle-generation logic from an ongoing asynchronous transfer with a latency of a few PCI clock ticks.

The link hardware employs a 16-deep ring buffer of pointers to transmit buffers and a similar ring of receive buffer pointers for each transfer class. Buffer contents are segmented into a stream of 608-byte cells, with each cell carrying a 32-byte hardware header and 320 bytes of message payload. The cell is composed of 32 interleaved Reed-Solomon (19,11) codes blocks, each of which carries one header byte, ten payload bytes, and eight bytes of forward error correction code ("ECC"). A beam-synchronous message acquires access to the link from an asynchronous message cell stream with a maximum latency of one 4.864-microsecond cell length. A cell carrying a trigger pattern in its payload has the highest priority for link access. Fields in the cell header permit hardware implementation of independent flow control for each receive buffer, cell receipt acknowledgement, and receipt timeout followed by a cell retransmission sequence, as well as remote register manipulation. The ECC scheme is tolerant of a noise burst more than a microsecond long in every cell. The link protocol design was funded by an SBIR grant for technology for the Next Linear Collider. However, the cell parameters were tailored to the characteristics of the SLAC LINAC klystron modulator pulse, which is most noisy during rise and fall times that are less than a microsecond long and are separated by a 5-microsecond flattop.

The CAMAC hardware for beam-synchronous and beam-asynchronous operations each comprises separate DMA memory interface units and transfer execution units for every one of the eight (i.e. 4 SLAC + 4 IEEE-standard) serial CAMAC cables. Beam-asynchronous block transfers are interrupted in favor of synchronous operations with a maximum latency of one CAMAC cycle. With typical round-trip cable propagation delays this is approximately 10 microseconds with the SLAC protocol and 6 microseconds in the IEEE-standard system. The IEEE-standard hardware also employs an enhanced block transfer protocol used in Kinetic Systems model 3952 crate controllers. This protocol supports pipelined block transfers at the full CAMAC crate dataway rate of 1 microsecond/word. If desired, the hardware can perform a "recovery cycle" to reinitialize a memory pointer register in a CAMAC module with properly incremented data before resuming an interrupted beam-asynchronous operation.

4 SOFTWARE FEATURES

The SHARCs on both boards employ custom real-time kernels to provide interrupt-driven CPU scheduling and serialization of access to shared I/O and memory resources. The FECC's kernel supports the full C execution environment, including separate run-time stacks for each thread of execution. The PCIL per-thread context is somewhat smaller as the kernel only supports assembly language programming. This context excludes both hardware and software stacks as well as a substantial number of processor registers. The kernel's interrupt-driven scheduling latency is 3-5 microseconds, and varies with the amount of hardware stack data that must be transferred to/from software stacks during a FECC scheduling event.

The PCIL software's function is largely limited to support for message transfers between the host micro and the FECC, and for FECC access to read/write buffers in micro memory. Forwarding of a trigger pattern from PCI-loadable PCIL registers to the FECC is accomplished without any aid from PCIL software. PCIL support for emulation of the existing CAMAC interface attempts to minimize the number of PCIL-FECC message exchanges by combining commands with write data, and by merging status with read data.

The FECC's application software consists of assembly code to support message transfers and remote micro buffer access and emulate the functions of the existing CAMAC and BITBUS interfaces (with real-time extensions), plus a much larger volume of C code that replaces the functions of the current micro's beam-synchronous software. Moving the execution platform for the micro's synchronous software to the FECC minimizes CAMAC access latency and reduces the micro's real-time load. Separate C entry points exist for initialization, processing a 360 Hz trigger interrupt, and receiving each class of incoming message.

5 ACKNOWLEDGEMENTS

This research was funded by the U.S. Department of Energy {"DOE") under SLAC's operating contract and its subcontracts. Additional DOE funding for PCIL2 development was provided by SBIR grant DE-FG02-98ER82628 to NYCB Real-Time Computing, Inc.

WEAP023
cs.DC/0111033

MODERNISING THE ESRF CONTROL SYSTEM WITH GNU/LINUX

A.Götz, A.Homs, B.Regad, M.Perez, P.Mäkijärvi, W-D.Klotz
ESRF, 6 rue Jules Horowitz, Grenoble 38043, FRANCE

Abstract

The ESRF control system is in the process of being modernised. The present control system is based on VME, 10 MHz Ethernet, OS9, Solaris, HP-UX, NFS/RPC, Motif and C. The new control system will be based on compact PCI, 100 MHz Ethernet, Linux, Windows, Solaris, CORBA/IIOP, C++, Java and Python. The main frontend operating system will be GNU/Linux running on Intel/x86 and Motorola/68k. Linux will also be used on handheld devices for mobile control. This poster describes how GNU/Linux is being used to modernise the control system and what problems have been encountered so far[1].

1 INTRODUCTION

The ESRF control systems control 3 accelerators and 32 beamlines. They have been built using the same technology and are completely compatible. They were built 10 years ago based on the state-of-the-art technology ten years ago. This included VME, 10 MHz Ethernet, OS-9, Solaris, HP-UX, NFS/RPC, Motif and C. Most of these technologies have not evolved over the last few years. In our search for better tools, support, ease of programming, and overall stability and quality we have put all our old technologies to the test. Our main criterium was which technology or tool will allow us to offer users a better control system. A better control system means one which offers more features to users without losing any of the present good features.

The result of this technology survey was 100 MHz Ethernet, VME (for the existing hardware), CompactPCI (cPCI) and PCI for new hardware, Linux as the main frontend operating system, Windows for commercially supported hardware and software, Solaris and GNU/Linux as the main desktop operating systems, CORBA/IIOP as the new network protocol, C++, Java and Python as the main programming languages.

2 WHY GNU/LINUX?

What does GNU/Linux offer that other systems don't?

1. FREEDOM! Freedom in this context means access to all the source code so that it can be compiled, understood and improved. An additonal freedom is the freedom from supplier pressure and fees.

2. Technology we know well (Unix) and which is conceptually simple to understand and program. This is an important feature in our case because we need to develop device drivers. In addition to being easy to understand it is well-documented.

3. A rich set of software packages. Almost all known sourceware packages have been developed or ported to GNU/Linux.

4. It is easy to manage in a network environment and has excellent support for all network protocols. Because our control systems are distributed over the network this played a strong role in our choice for GNU/Linux.

3 LINUX/M68K + VME

The ESRF has over 200 VME crates installed. This represents an investment of millions of Euros as well as many tens of years of work in hardware and software development. Any modernization project must take this investment into account. The modernization foresees two ways to do this: using the Motorola CPU's (MVME-162) to run GNU/Linux directly, or replacing the CPU with a bus extender which allows the VME bus to be controlled from PC running Linux/x86. This section describes the first option. The bus extender solution is discussed in the next section.

For Linux/m68k we use the Debian distribution 2.1. It can be downloaded from the Debian website[2] and is available in source code and binary format. The standard kernel (we are running kernel 2.2.10) includes the support for the Motorola CPU port (originally done by Richard Hirst[3]). We run all our Linux/m68k crates without hard disk (diskless). The root disk is NFS mounted readonly. In addition there is a RAM disk for /etc, /dev, /var and /tmp. This means crates can be switched on/off without risk of losing data nor do we have to do fsck's. We have rewritten device drivers for all our main VME cards. For many of them we subcontracted the driver writing for the first version to Richard Hirst (later Linuxcare). Maintenance and further development is now done in house. Client programs communicate with the hardware via the network using TACO/TANGO device servers (cf. below). We use the GNU tools for compiling and debugging (g++ and gdb).

[1]work supported by J.Klora, J.M.Chaize and P.Fajardo

[2]http://www.debian.org

[3]rhirst@sleepie.demon.co.uk

Our experience with Linux/m68k compared to OS-9 (the commercial operating system we were using previously) is that it is at least, if not more stable, the TCP/IP implementation is more efficient and robust and it is easy to add new features to our software using standard techniques like multithreading.

4 LINUX/X86 + BUS EXTENDERS

The modernization project of the instrument control at the ESRF using GNU/Linux supports two main hardware platforms: PCI/cPCI and VME. The former provides access to the most recent interface boards developed for a highly demanding market, and hence, with better performance/price ratios. The latter is needed for a gradual transition between the current VME instrumentation and the PCI technology. VME boards can be controlled from a Motorola MVME CPU or from a PC through a PCI/VME bus extender, both running GNU/Linux as OS.

As it was said before, the modernization project also includes the cPCI platform, which, in combination of PCI/cPCI bus extenders, notably increases the flexibility in the hardware configuration. It is well known that due to the dynamic resource configuration in the PCI specification, identical boards are only distinguishable by their slot position in the bus. Most of the drivers available for PCI boards enumerate the boards in the same order they are found by the BIOS / OS at boot time. This means that the board identification number will change when a similar one, situated before in the PCI bus structure, is removed. Moreover, if we apply the same logic to slave cPCI crates, their bus numbers will change when another is removed. To solve this problem a differentiation between physical numbers, those used by the drivers, and logical numbers used by applications is made. The mechanism responsible for making this mapping keeps track of the boards present in the system and detects any change in the bus configuration. Any non-trivial change is informed to the user, avoiding wrong addressing to the boards. The position of the boards are presented to the user in terms of chassis and slot, which are translated to PCI device numbers by hardware specific mappings.

Setups based on the PCI architecture have been mounted using both a desktop PC and an industrial PC that implements the PICMG standard. Remote VME crates are controlled through SBS Technologies PCI/VME bus extenders, and cPCI crates are directly linked to the main PCI bus by means of National Instruments MXI-3 PCI/cPCI/PXI bus adapters. These adapters expand by a large factor the amount of hardware that can be managed by a single host. Furthermore, both MVME and PCI GNU/Linux can independently control boards in the same crate, providing even more possibilities for the VME - PCI transition.

5 DEVICE DRIVERS

In order to use the same device driver codes in both systems, an interface layer was implemented to manage I/O addresses and IRQs, taken from the module parameters at load time. In the bus coupler configuration, this interface does the necessary PCI to VME address mappings during the initialization of the VME board drivers, allowing boards on different (remote) VME buses to be controlled from the same host as local. This interface also exports automatically the state and configuration of each board to the virtual file system.

In experiment automation it is often very useful to record the value of several magnitudes when an event occurs. Such an event can be generated by a hardware signal or by a software condition. To provide this functionality a buffering mechanism was developed in the kernel, named “hook” after a similar facility developed at the ESRF for OS/9 drivers, which hooks data on hardware interrupts. The values to be written in the buffer are run-time configurable by specifying the driver name, the board and the channel to be read. Each driver that can export its channels will register with the hook module during initialization. When an application wants to read one of its channels, the hook asks for the necessary actions to be done. If the actions are just simple register read/write operations (one single read is very common), they are returned in the form of a "program". Otherwise, if the process is more complicated, a pointer to a function is saved. One source of hook events is a timer provided by the hook itself, which attaches to the system software timer, and hence has a minimum repetition period of 10 ms on standard installations. Higher rates can be achieved with hardware interrupts generated by counter/timer boards like the ESRF VCT6.

Not all the boards allow a fast reading of their registers, and the system should not wait in an interrupt handler (actually a bottom half handler). This problem can be overcome with an asynchronous buffer write, as long as it is done before the next event arrives. Finally, the hook buffer can be filled in linear mode, which stops acquisitions when the end of the buffer is reached, or in circular mode for continuous measurement.

6 DEVICE SERVERS

The device drivers are the first layer in our control system architecture. The second layer is called the *device server* layer and provides transparent network access. This means hardware can be shared transparently between geographically separated parts of the accelerator and/or beamline, thereby adding a layer of flexibility which would otherwise not be available (except by recabling). The device servers at the ESRF are of two flavours. The original flavour called TACO[4] uses the ONC/RPC network protocol and is a lightweight protocol. It has the advantage that the ONC/RPC runs everywhere where NFS runs. The second flavour called TANGO[5] is based on CORBA and uses the IIOP protocol for the network layer. CORBA is slightly more heavyweight than ONC/RPC but offers more high-

[4] http://www.esrf.fr/taco
[5] http://www.esrf.fr/tango

level services. Both flavours of device servers offer synchronous, asynchronous and event-driven communication paradigms and a database for permanent storage. A large number (hundreds) of device servers have been written at the ESRF and other sites (FRM II, Lure, HartRAO). Refer to the websites for more information.

7 ADMINISTRATION

The challenge we are facing with the modernization of the control system is we must be not only able to provide the best combination of operating system+hardware, but also to be able to do the system administration of the system installed all over the site. Administration means two important things:

- quick recovery of a system after a failure.
- new release of the system.

Our present control system is based on VME / OS9 diskless systems. These OS9 systems are served by BOOTP servers which give them an identity and then downloads the kernel onto the VME crate at startup. The VME crates then all mount the same remote file system using NFS. Thus, for example if a CPU fails we have just to change it and press the ON button. This type of action takes less than 15 minutes.

With the modernized control system, the BOOTP technique is replaced with DHCP which is based on BOOTP but has more powerful features. In this way DHCP allows for dynamic allocation of the network address.

The compact PCI crates that are being installed are equipped with a hard disk. It is inconceivable that numerous systems that will be running in the future have each their own configuration. In that case it would be almost impossible to do the system administration and to provide a good service to the users of these systems.

The decision was taken to do a base system and to duplicate it as many times as it will be needed. This technique is also called "cloning". The ESRF has bought a commercial product called REMBO (for REMote BOot). This tool is able to deal with DHCP technology to give a network identity to a client that broadcasts a DHCP request. Moreover this tool can be used to make a base image of any system (Linux / Windows) and to upload that image on a computer on which a Rembo server has been installed. REMBO is a cloning tool and a backup utility as well. The administration of the Compact PCI can be improved with the capability of REMBO to do differential image. We use this feature to manage different hardware configurations starting from a base system. For example the crates dedicated to vacuum and the crates dedicated to the front end system have the same base hardware (same crate, same CPU , network card) and therefore will have the same base image (same operating system, network drivers, etc.). But since these crates do not have the same I/O boards installed a differential image for each type of hardware configuration will be made. In case of failure (crash of the hard disk) the tool will be able to rebuild the whole system from the base image and the differential one.

8 HANDHELD DEVICES

We have long had the need for handheld portable controllers for motors, and other hardware which need local tuning far from the control room. Enter the new generation of handheld devices, wireless Ethernet and Linux. We have used the iPAQ from Compaq running GNU/Linux and an X11 based client (Labview for example) to control motors remotely using a simple one-click interface. The fact of choosing GNU/Linux makes the task of network and graphical display much easier than if we were using Windows CE for example.

9 REALTIME

"Linux isn't realtime". This is true therefore we do not claim to do any realtime with GNU/Linux. However we have found that for our applications we need little or no realtime. Most realtime needs are delegated to hardware or DSP's. Where we do need soft realtime (i.e. 99% guarantee) we use interrupt routines in device drivers. We measured an interrupt response time of $< 50\mu s$ for Linux on 68k and x86. Using a driver interrupt routine we achieve a soft realtime response of $500\mu s$ for a function generator (providing we do not recompile the kernel at the same time!). For the rest of our applications *"as-fast-as-possible"* is good enough. And for that GNU/Linux on commodity hardware is surprisingly good.

10 PROBLEMS

GNU/Linux is not without problems. The main problems we have identified so far are:

- the standard GNU/Linux distributions are not easily adapted to running on diskless systems
- most commercial hardware does not have GNU/Linux drivers but Windows drivers

11 CONCLUSION

There is a viable alternative to Windows 95/98/ME, NT/2000/CE/XP for building control systems and it is called GNU/Linux! Linux is sufficiently mature for the task and even offers some advantages i.e. it is easier to program, is better adapted to distributed control and is free of commercial pressure.

WEAP025
physics/0112037

INTRODUCTION OF MODERN SUBSYSTEMS AT THE KEK INJECTOR-LINAC

N.Kamikubota, K.Furukawa, KEK, Tsukuba, Japan
S.Kusano, T.Obata, Mitsubishi Electric System and Service Co. Ltd., Tsukuba, Japan

Abstract

As an accelerator control system survives over several years, it is often the case that new subsystems are introduced into the original control system. The control system for the KEK electron/positron injector-linac has been using Unix workstations and VME computers since 1993. During the eight-year operation, we extended the system by introducing a) Windows PCs, b) PLC controllers with a network interface, and c) web servers based on modern information technology. Although such new subsystems are essential to improve control functionalities, they often cause communication problems with the original control system. We discuss the experienced problems, and present our solutions for them.

1 INTRODUCTION

The KEK linac was constructed as an injector of the Photon Factory storage ring about 20 years ago [1]. The first beam of 2.5-GeV electrons was provided in 1982. This linac now provides electron/positron beams to several rings [2]: a) 3.5-GeV positrons to the KEKB LER (KEK B-factory Low-energy ring), b) 8-GeV electrons to the KEKB HER (High-energy ring), c) 2.5-GeV electrons to the PF ring, and d) 2.5-GeV electrons to the PF-AR ring. The first control system, which consisted of mini-computers and CAMAC interfaces [3], was replaced by the present control system in 1993 [4]. The present system comprises Unix workstations and VME computers. It has been upgraded occasionally [5] and used over the past eight years.

In this article, we discuss newly introduced subsystems in recent years. They were introduced in order to improve the control functionalities, and/or to enable better maintenance capabilities. Three subsystems are described in detail in Section 2. The experienced problems between the subsystems and the original control system, and their solutions are discussed in Section 3.

2 NEW SUBSYSTEMS

2.1 Control System Overview

The present control system comprises 4–6 UNIX workstations, 27 VME computers with the OS-9 operating system, 140 PLC (Programmable logic controller) controllers, and 11 CAMAC interfaces with a network port. A home-made RPC (remote procedure call), based on TCP/UDP protocols, are used for communication between them. A simplified view of the control system is shown in Fig. 1.

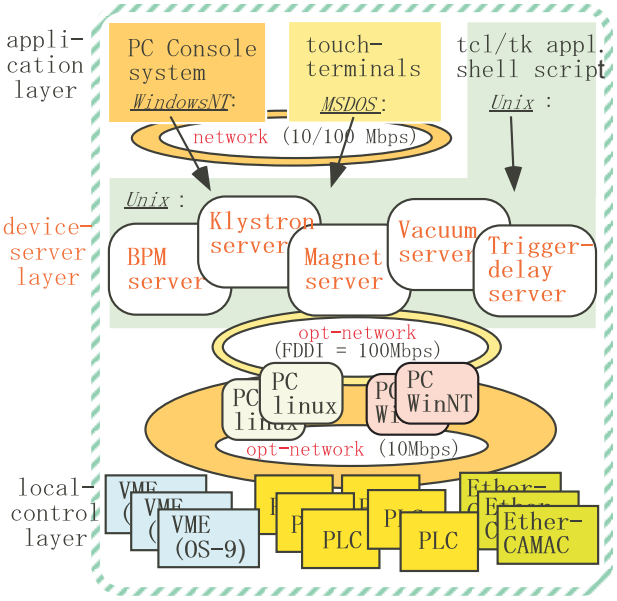


Figure 1: Simplified view of the control system.

The total number of control signals is about 6000 (in 2-byte unit). Since 1998, when the CP violation study started with the KEKB rings, the operation of the linac exceeded 7000 hours per year. The number of control transactions handled by the control system increased every year, and reached 350 transactions per sec. in June, 2001 [6].

2.2 Windows PC

Since the end of the 1980'es, we have had strong interest in using PCs. The PC-based operator's console system started with DOS PCs [7], and later reinforced with Windows PCs [8], has been successfully used for more than ten years. In the early phase, the following points were preferable for us: a) enhanced capability of 2-byte code (Japanese characters) handling, b) good development environment of graphic applications, and c) low cost.

The present console system comprises about ten PCs (Windows NT and Windows 2000). The surveillance applications for accelerator devices were developed in Visual Basic, and have been used in daily operation. The operation

log-book using MS-SQL and Access [9] is extensively used everyday with this console system.

Communication with the control system, which runs at the Unix workstations, is made by a gateway (a Windows PC). When the gateway receives a control request from a console PC by the OLE, it communicates with the appropriate device server(s) by using the RPC protocol (see Fig. 1). The gateway and the present console system have been successfully used over the past six years.

2.3 PLC

The main part of the control system for the KEK linac was renewed in 1993 [4]. However, the local controllers (shown as SBC[1] in Table 1) remained. In recent years the maintenance of these local controllers has become difficult. Thus, we decided to replace them with new controllers.

Table 1: Replacement of local controllers.

device and transition	before 1993	transition phase	present status
Klystron '97-'98	CAMAC –>SBC	VME –>SBC	Ethernet –>PLCx70
Magnet '96-'00	CAMAC –>SBC	VME –>SBC	Ethernet –>PLCx51
Vacuum '96-'97	CAMAC –>SBC	VME –>SBC	Ethernet –>PLCx18
Trigger '97-now	CAMAC –>SBC	VME –>SBC	Ethernet –>CAMACx11
BPM since '97	none		Ethernet –>VMEx19

A typical local controller should have a) a few hundred I/O points, b) simple but programmable control logic, and c) a communication path to the main control system. Among some candidates, a PLC with a direct network port (Yokogawa FA-M3) was chosen for klystron modulators, magnet power-supplies, and vacuum controllers. The VME computer was a candidate, but was not selected, because the PLC is less expensive. The replacements of local controllers since 1996 are summarized in Table 1.

It is interesting that all of the new controllers shown in Table 1 have an Ethernet port. A background fact is that world-standard field networks (CAN-bus, Profi-bus, MIL1553, etc.) are not popular in Japan, and we want to use Ethernet as a field network. The use of a standard Ethernet is preferable for long-term maintenance and cost reduction. In addition, we use optic-fiber cables for the network to local controllers in order to avoid electro-magnetic noise from the klystron modulators.

2.4 Web and Related Topics

We have recently experienced fast improvements of information technologies. The world-wide-web services at the KEK linac started in May, 1994 [10]. Up to now, we have developed many web pages to inform about the linac operation status.

a) Status of accelerator devices The web-server machine is a part of the control system. Thus, by using the CGI (Common Gateway Interface) script, it is easy to develop a homepage to show the status of any linac device. A large number of pages have already been developed.

b) Real-time display by Java and CORBA Feasibility studies of a web-based real-time display using Java and CORBA have been carried out. The measured round-trip time between a Java applet and a CORBA server (at an Unix workstation) was 50 ms [11, 12]. The server does not consume CPU resources compared with the CGI-based services. Recent updates have enabled realistic demonstration of the beam-current history at the KEK linac [13].

3 DISCUSSION

3.1 Problems with Subsystems

a) Windows PC The main language for the Windows-based console PCs is Visual Basic, while the sources at the Unix side have been developed in C language. For example, the sources for the RPC use socket (Winsock) functions at the Unix (Windows) side. Thus, the maintenances have been made independently. This fact implies that when we have some improvements at the Unix side, it always takes time for the improvements influence the Windows side.

We have operated a TCP/IP network system which contains both Unix workstations and Windows PCs. As the number of Windows PCs has increased, we experienced communication errors by two specific intervals (2 hours and 12 minutes). They were removed by changing the default settings of MS Office and Samba [15]. We also experienced an accident in which the network burst from Windows PCs occupied the network system, followed by a short-time mistake of network cable connections. At the time of the accident, the burst stopped all network modules of the PLC controllers. We modified the parameters at the network routers so as not to enhance the burst broadcasts.

b) PLC After the KEKB commissioning started in 1997, we developed various slow-feedback applications [14] in order to realize stable beam injections to the KEKB rings. By the end of 1998, the CPU capabilities of Unix workstations were found to be insufficient for increasing demands. The analysis showed that the klystron server (see Fig. 1) consumed a very large fraction of the CPU resources for network communication with PLC controllers. In the summer of 1999, we prepared on-memory cache areas to keep klystron data at the Unix workstations. Two linux PCs have been used to update the cached data by polling the PLC controllers [15, 16]. The network traffic decreased to one fourth, and the problem disappeared.

[1] Such old local controllers were controlled by Single Board Computers with micro-processors [3].

Up to now we have introduced the cached scheme even for other devices.[2]. Considering the number of control transactions [6], we can conclude that the intelligence of the PLCs was not sufficient for our case. Thus, we added more intelligent devices (PCs) between the device servers and the PLC controllers, as shown in Fig. 1.

c) Web Web presentation of the KEK linac status would consume larger CPU and network resources than dedicated applications. The considerable increase of web accesses in recent years implies that we will need more computer resources in our control system. A more serious problem is that we will need more man-power to maintain both dedicated applications and web-based services. We are eager for some tools which would enable us automatically generate web contents.

3.2 Transition of Control Architecture

We are now ready to discuss the long-term transition of the basic architecture of the control system for large accelerators. Taking into account various accelerator control systems in the past 20 years, the transitions of control standards are given in Table 2 (upper). Private expectations for the next decade (2000'es) are shown in Table 2 (lower).

Table 2: Transition of large accelerator control.

	1980'es	1990'es
console	CUI (text-base)	GUI on X (window-base)
base-machine (language)	mini-computer FORTRAN	Unix workstation C
network	dedicated network dedicated protocol	standard Ethernet on TCP/IP
local-controller	CAMAC	VME with RT-OS

	2000'es (private expectation)
console	a) GUI on Windows-PC (for operation) b) toolkit/environment (for study) c) Web/cell-phone (for announcement)
base-machine (language)	Linux, or 64/128bit Unix Java, C++
network	TCP/IP and CORBA http (for web)
local-controller	PLC with Ethernet (for simple I/O) Linux box (for intelligent controller)

The present control system for the KEK linac, which started in 1993, can be expressed as a typical standard of the 1990'es model. We conclude that our extensions (introducing subsystems in the past eight years) can be understood as an evolution toward the new standard of 2000'es.

[2]The vacuum system started to use cache in June, 2000. For magnet power-supplies, we had a Windows PC as a gateway since 1997 [15]. The BPM servers use cache from the start of the service in 1997.

4 ACKNOWLEDGMENT

The authors acknowledge Prof. A. Enomoto and Prof. K. Nakahara for kindly supervising our work. We thank I. Abe for discussions on using Windows PCs with Unix workstations. The various improvements of PLC-based local controllers have been carried out by A. Shirakawa. We also thank the KEK linac operators for cooperative and successful works to improve our control system.

5 REFERENCES

[1] J.Tanaka, Nucl. Instr. Meth. 177(1980)101-105

[2] Y.Ogawa and Linac Commissioning Group (LCG), "Commissioning Status of the KEKB Linac", Proc. PAC'99, New York, Mar.-Apr.1999, p.2984-2986

[3] K.Nakahara, I.Abe, R.P.Bissonnette, A.Enomoto, Y.Otake, T.Urano and J.Tanaka, Nucl. Instr. Meth. A251(1986)327

[4] N.Kamikubota, K.Furukawa, K.Nakahara and I.Abe, Nucl. Instr. Meth. A352(1994)131-134

[5] N.Kamikubota *et al.*, "Improvements to Realize a Higher Reliability of the KEK Linac Control System", Proc. ICALEPCS'95, Chicago, Oct.-Nov.1995, p.1052-1055

[6] N.Kamikubota *et al.*, "Growth of Control Transactions of the KEK Linac during the KEKB Commissioning", Proc. APAC'01, Beijing, Sep.2001; KEK-Preprint 2001-124

[7] K.Nakahara, I.Abe, N.Kamikubota and K.Furukawa, Nucl. Instr. Meth. A293(1990)446-449

[8] I.Abe, H.Kobayashi and M.Tanaka, "PC based Control System using ActiveX in the KEK e-/e+ Linac", Proc. PCaPAC'99, Tsukuba, Jan.1999, KEK-Proceedings 98-10

[9] M.Tanaka, I.Abe and H.Kobayashi, "Database system in the KEK Linac PC-based Control", Proc. PCaPAC'99, Tsukuba, Jan.1999, KEK-Proceedings 98-10

[10] http://www-linac.kek.jp

[11] S.Kusano, N.Kamikubota and K.Furukawa, "Real-time Display of Accelerator Status Using JAVA and CORBA", Proc. PCaPAC'99, Tsukuba, Jan.1999, KEK-Proceedings 98-10

[12] S.Kusano, N.Kamikubota and K.Furukawa, "Study of Sharable Applications Using Java and CORBA", Proc. ICALEPCS'99, Trieste, Oct.1999, p.535-537

[13] N.Kamikubota, S.Kusano and K.Furukawa, "Archive Databases as Distributed CORBA Objects", presentation at JPS meeting, Okinawa, Sep.2001

[14] K.Furukawa *et al.*, "Beam Switching and Beam Feedback Systems at KEKB Linac", Proc. LINAC2000, Monterey, CA, Aug.2000, p.633-635

[15] N.Kamikubota, "Introducing PCs to Unix-based control systems", presentation at PCaPAC2000, Hamburg, Oct.2000

[16] S.Kusano, N.Kamikubota and K.Furukawa, "Use of Linux PCs for Device Surveillance at the KEK Injector-linac", to be submitted

WEAP026
hep-ph/0111079

U-70 PROTON SYNCHROTRON EXTRACTED BEAM LINES CONTROL SYSTEM MODERNIZATION

V.Alferov, Y.Bordanovski, S.Klimov, V.Ilukin,V. Kuznetsov, O.Radin A.Shalunov, A.Sytin, P.Vetrov, V.Yaryguine, V.Zapolsky, V. Zarucheisky
Institute for High Energy Physics, Protvino, Russia

Abstract

A 70 GeV Proton Synchrotron Extracted Beam Lines Control System is described. Approximately 130 Magnet Dipoles and Quadrupoles, 20 Correction Magnets, 50 Beam Collimators, and BPM equipment spread over 1 Km have to be controlled. The old system was based on the PDP-11/40 and LSI-11 compatible computers and the MIL 1553 STD as a Field Bus. It successfully operated about 15 years. A new system includes home made Equipment Controllers based on I 8051 Processors, CAN Field Bus, FECs, Servers, Consoles connected by Ethernet. On the first stage of modernization, PDPs and LSIs are replaced with PCs connected by Ethernet. Equipment controllers are being successfully tested in the Collimators and Corrector Magnets control during a run.

1 INTRODUCTION

The Extracted Beams on the Serpukhov 70 GeV Proton Synchrotron are spread over 1 km. They include approximately 130 Magnet Dipoles and Quadrupoles, 20 small Correction Magnets, 50 Beam Collimators, BPM, vacuum, interlock and other equipment. The Power Supplies (PS) of the Dipoles and Quadrupoles are installed in the special building 500m away. Total number of the I/O signals about 1500.

The old Control System was designed in the early 80s [1]. It included PDP-11/40 compatible Host Computer, two LSI- 11 compatible FECs, and five MIL 1553 Field Buses to distribute the digital commands to the PSs. The old system was also used to connect two Beam Control Rooms with the PS Building Control Room, Digital Voltmeters and distributed multiplexers for the control of the magnets current. During the 10 years of operation, the system demonstrated high reliability, but heavy condition electronics, as well as time, caused the system to degrade. A major disadvantage was the slow (tenths of seconds) setting of regimes. A parallel access to the equipment from different control rooms also was impossible.

A new project is based on the standard 3-layer model. The backbone of the system is an Ethernet LAN connecting a Power Supplies Building with the Beam Lines area including experimental control rooms. The three layers consist of an upper level presented by PCs, a middle level by special Field Bus controllers GPFC [2], and a lower level by home-made Equipment Controllers (EC). GPFCs and ECs will be implemented on the second stage of upgrading, after testing with a small group of equipment. On the first stage, implementation of PCs connected by Ethernet was performed without changing interface electronics.

2 SYSTEM ARCHITECTURE

The layout of the control system is shown in Figure 1. There are three main groups of controlled equipment. The most numerous are Dipoles and Quadrupoles PSs in the PSB. They are connected to PCs in the PSB Control Room by four MIL 1553 Field Buses. A CAMAC Crate houses relevant Bus Controllers. Field Bus is used for distribution of regime digital commands to DACs of PSs. Control of regimes is provided by means of distributed Analog Multiplexer connecting PS Shunts to the Digital Voltmeter. A Status Register is used for controls of polarity, water cooling, etc.

A second group of equipment is the low current PSs of Correction Magnets distributed along Beam Lines. They are controlled from two Beam Control Rooms in the same way.

A third group are Beam Collimators. The multiplexed Collimator Controllers (CAMAC modules in two Beam Control Rooms) perform position control with help of shaft encoders and length modulation controls of DC Motors.

3 THE SOFTWARE DESCRIPTION

The software is developed under MS Windows 98/NT using Visual C++/Visual Basic and client-server approach based on DCOM (Figure 2). The configuration of the PSs is placed in the Microsoft Access database.

The complex data exchange between different PCs in the different buildings and user-friendly interfaces are available. The powerful database tools provide different possibilities in the post-mortem analysis, easy configuration, and extensibility of the data tables.

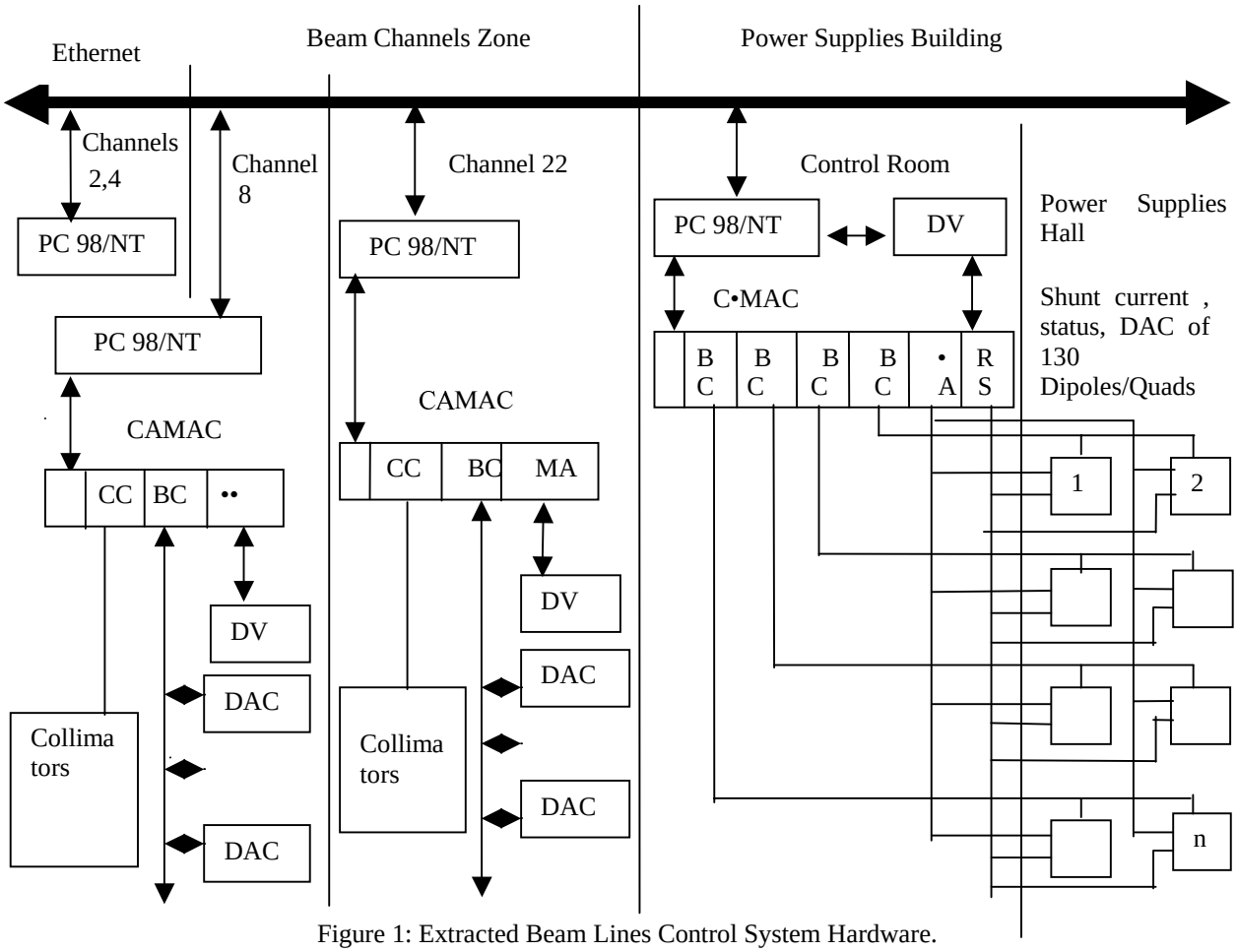


Figure 1: Extracted Beam Lines Control System Hardware.
CC – Collimator Controller, BC – MIL 1553 BUS Controller, MA – Analog Multiplexer,
DAC – Digital to Analog Converter, DV – Digital Voltmeter, RS – Status Register.

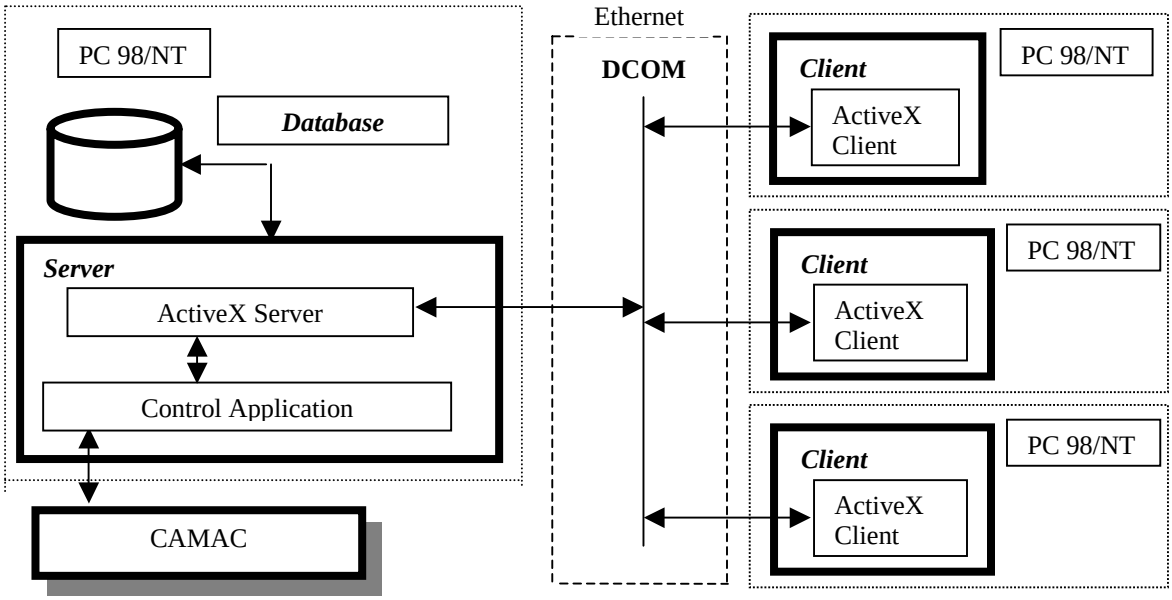


Figure 2: The structure of the Power Supplies control system software.

Control of the PSs from the different control rooms in more convenient ways is also available, and the conflicts with the simultaneous access to the equipment are down to a minimum.

New features to access the data through the web browsers have been added.

The implementation of the equipment controllers on the second stage of the upgrade will provide parallel access to the Power Supplies and Collimators.

REFERENCES

[1] Preprint IHEP Protvino 88-73, Protvino, Russia, 1988.

[2] A.Matioushine at al., A Configurable RT OS Powered Fieldbus Controller for Distributed Accelerator Controls, Proc. Of ICALEPCS'97, Beijing, China, November 3-7, 1997, p.321.

WEAP028
physics/0111147

LINUX-BASED TOOLKIT ON THE VEPP-4 CONTROL SYSTEM

V. Blinov, A.Bogomyagkov, M. Fedotov, S. Karnaev, O. Meshkov, N. Muchnoi, S. Nikitin,
I. Nikolaev, A. Selivanov, S. Smirnov, V. Tsukanov,
Budker Institute of Nuclear Physics, Novosibirsk

Abstract

The control system of the VEPP-4 facility was designed more than fifteen years ago and based on the home-developed CAMAC-embedded minicomputers Odrenok [1]. Five years ago, all computers were connected via Ethernet network. This step allowed us to force an integration of PCs into the VEPP-4 control system. This paper reviews new tools running on the PCs under Linux.

The beam diagnostics tool described in this paper provides the data about position and dimensions of e^-/e^+ beams from CCD matrix controller via 100-Mbit Ethernet.

The next tool provides measuring of beam energy at the VEPP-4M collider using a well-known method of resonance depolarization by the observation of the polarization degree on the effect of internal scattering of particles.

Data visualization tools are based on CERN ROOT framework.

Hardware and software aspects of the systems are presented in this paper.

1 INTRODUCTION

The first steps to control accelerators in BINP with computers began at the end of the 1960's when ODRA-1304 machine (ICL-1900 series) was used to control the acceleration process on the VEPP-3 storage ring. Later in the 1970's, several ODRA-1305 and ODRA-1325 machines were implemented at the VEPP-4 complex, which includes injection complex, VEPP-3 (2 GeV electron/positron storage ring), and VEPP-4M (1 – 6 GeV electron-positron collider). During this time a great number of electronic modules, applications, and a custom real-time operating system were developed in BINP. In the 1980's the VEPP-4 control system was upgraded. The main goal was to replace obsolete equipment and electronics by CAMAC. In the beginning of the 1980's an Odernok intelligent CAMAC controller with ODRA instruction set was developed in BINP. The use of Odrenok allowed the continuation of customary programming for more than 20 years. All applications were programmed on the ODRA and Odrenok in the custom version of Fortran language – TRAN 1900. In 1998 a custom Ethernet network was implemented for Odrenok [2]. This effort led to the use of PCs as operator consoles connected to Odrenok network.

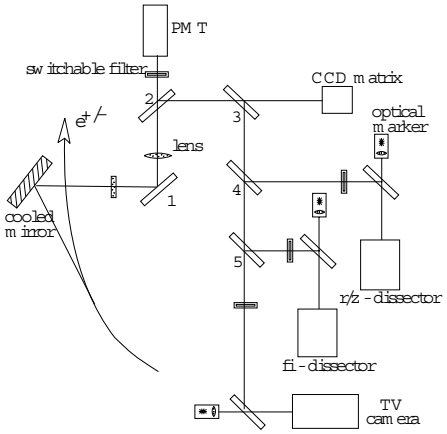
The extensive custom development became an impediment to progress. The alphabetic user interface is primitive by today's standards. The introduction of the PCs as control room machines into the VEPP-4 control system makes programming possible and attractive based on open standards and X display systems.

2 BEAM IMAGE OBSERVATION

2.1 Layout

Optical diagnostics at the VEPP-3 and the VEPP-4M rings [3] are provided by the dissectors and linear CCD arrays. The optical component of the synchrotron radiation (SR) of the beams is used for this. The dissectors are applied for the measurements of the length and the transverse size of the bunches, and for the study of the collective effects. The linear CCD array provides the measurements of the vertical size of the beams. Dimensions of the beam are: σ_z = 0.1 mm, σ_r = 1.5 mm, σ_φ = 5 cm (E = 5 GeV, U_r = 5 MeV). Now the linear CCD arrays are replaced by CCD matrix SONY ICX084AL (659×489 pixels).

The scheme of the optical diagnostic devices is shown in Fig.1.



Two channels of SR output with the optical diagnostics devices are located at the VEPP-4M (each channel is for electron or positron direction only) and one channel is placed at the VEPP-3.

The optical component of SR is reflected by cooled metallic mirror and sent out from the vacuum chamber through a glass window. Mirror 1 matches the SR ray with an optical axis of the system. The beam image is set up on a lens. Mirrors 2-5 split the optical beam to different measurement devices.

2.2 Electronics of CCD

The CCD Matrix Controller (MC) (Fig. 2) includes 5 MHz 16-bit ADC and fast Ethernet transmitter. The buffer registers and the interaction logic are based on an Altera PLD.

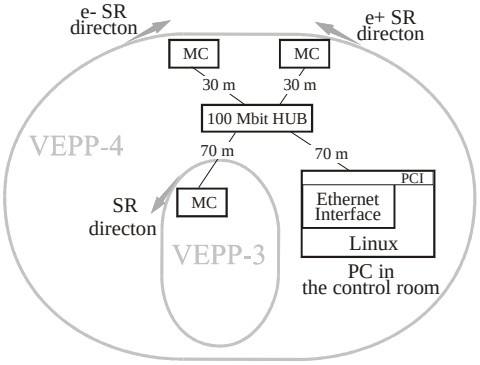


Figure 2: Layout of the beam size observation system at the VEPP4 facility.

The use of Linux allows the easy connection of MCs to the PC via fast Ethernet interface card. The total circumference of the VEPP-4M facility is 366 m therefore intermediate HUB for MCs is used for the connection to the PC.

2.3 Software

The use of the CCD matrix instead of the linear CCD array allows us to observe a beam spot and to obtain both transverse dimensions σ_z, σ_r and transverse beam particle density. The high speed data transmission provides up to 10 images per second on the computer display.

A custom protocol for the transfer of control to MC and data acquisition was developed. This protocol provides an on-line exchange between MC and the PC. Each exchange operation includes sending of the control packet from the PC with function words for controller and status words for ADC, and receiving of the data packets sequence from MC. Each data packet from MC has length of 1.3 kByte including one string of the matrix, therefore the total number of the packets of the beam image is about 500. It takes about 100 ms to receive and process the beam image in the PC. Image processing in the PC is performed with ROOT framework.

Now this system is in preliminary usage. First test pictures are available at this site.[1]

3 BEAM ENERGY MEASUREMENT

3.1 Layout

For carrying out the experiment of the precision measurement of the τ-lepton mass near its production threshold at 1.8 GeV, it is necessary to know exactly the energy of colliding particles. For this purpose, special systems including counters of scattered electrons and plates of the TEM wave depolarizer are installed on the booster storage ring VEPP-3 and on the collider VEPP-4M [4].

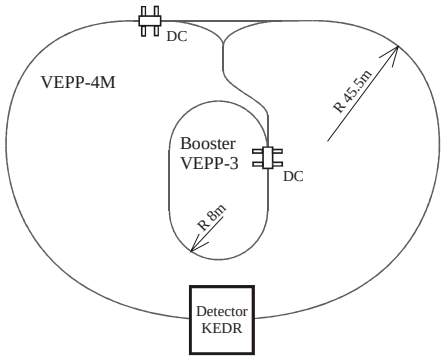


Figure 3: Layout of the Depolarizer-Counters.

The layout of the Depolarizer-Counter (DC) devices is shown in Fig.3. The current hardware configuration of the measuring system is depicted in Fig. 4.

The system consists of one PC, three CAMAC crates connected to the PC via special serial link, and 8 stepper-motor controllers connected to the PC via RS-485 serial link.

The PC is physically located in the main control room. It provides the operator interface to the control and measuring system.

All CAMAC crates are located in the radio control room. Two CAMAC crates include counter electronics, where all signals from counters are concentrated. One CAMAC crate includes the frequency synthesizer with the minimal band width of $\Delta f_d \sim 1 \div 10$ Hz and the rearrangement step of 1 Hz.

The Stepper-Motor Controllers (SMC) are remotely located at the facility's two DCs (see Fig. 3).

[1] http://www.vepp4-pult1.inp.nsk.su/~vepp4/size4/index.html

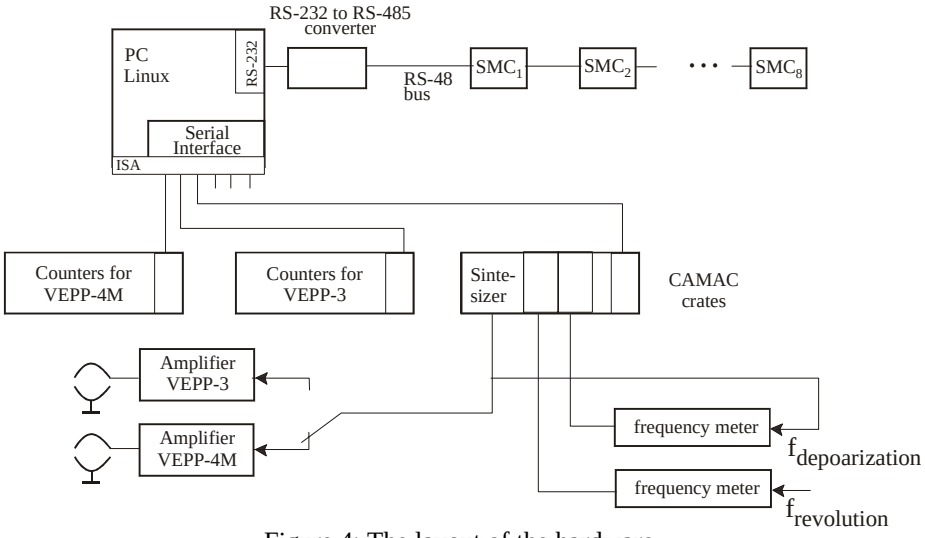


Figure 4: The layout of the hardware.

3.2 Software

The software for the measurement system is based on the ROOT package [5]. The ROOT system was developed in CERN, and provides a set of frameworks sufficient to handle and analyze data.

This software controls the frequency synthesizer and the stepper motors with X-window based tools, which prompts the operator to fill in the necessary values. Fields in the windows can be modified and used values are displayed.

The operator can observe a visual representation of the data acquisition from the counters in different graphical windows. He can interactively process the data with curve fitting, minimization, graphics and visualization tools.

The software is linked with the VEPP-4 control system [2] for getting the status of the VEPP-4 facility, the beam energy and the beam current values.

Some pictures of the depolarization process observation are available at this site.[2]

4 CONCLUSION

There is no possibility to abandon CAMAC and Odrenok computers in a short time. One of the solutions is to expand the use of GUI based applications for new control tasks. This approach will enhance the flexibility of the accelerator operations and provide new options for the accelerator control. The main goal of the future development of the VEPP-4 control system is to entirely retire the Odrenok computers.

REFERENCES

[1] A.Aleshaev et al, “VEPP-4 Control System”, ICALEPCS'95, Chicago, USA, p.799, F-PO-2.

[2] A.Aleshaev et al, “Data Base and Data Flow on VEPP-4 Control System”, ICALEPCS’99, Trieste, Italy, p. 439

[3] E. I. Zinin et al, “Status and further development of the optical diagnostics of the VEPP-4M storage ring”, WPAH318, PAC2001, Chicago, USA.

[4] V.E. Blinov et al, “A Development Of Resonance Depolarization Method At Vepp-4for High Precision Measurement Of Tau Lepton Mass”, RPPH041, PAC2001, Chicago, USA.

[5] http://root.cern.ch/root/Welcome.html

[2] http://www.vepp4-pult1.inp.nsk.su/~vepp4/polar/index.html

WEAP029
physics/0111148

DATABASE AND CHANNEL ACCESS ON THE VEPP-4 CONTROL SYSTEM

D. Filimonov, S. Karnaev, B. Levichev, S. Smirnov, A.Veklov,
Budker Institute of Nuclear Physics, Novosibirsk

Abstract

The VEPP-4 control system was designed almost 20 years ago as CAMAC-based distributed control system [1]. Today the main peculiarity of the upgraded VEPP-4 [2] control system is the employment of obsolete CAMAC-embedded 24-bit Odrenok computers in the middle (process) level.

The communications software is developed to provide an effective interaction between user (operator) applications on the high level and executive programs in the middle level. The communications software is Linux-based and includes the Application Server and Device Servers. The Application Server provides channel and database access for user applications. The Device Servers provide interaction with the hardware. The backend database server is based on PostgreSQL.

This paper describes the structure of the VEPP-4 control system database and data/channel access procedures.

1 INTRODUCTION

The VEPP-4 database for the control programs in the Odrenok computers is distributed in separate files according to the VEPP-4 complex subsystems. These files include information about electronics, control and measuring channels. Data format is 24-bit ICL word used in Odrenok. A few types of custom text editors are used for database viewing and editing.

The grave imperfection of Odrenok-based database was a lack of the possibility to log large volumes of data from equipment, like the vacuum gauge, temperature, magnet current, etc in a uniform way. The data acquisition programs running on Odrenoks support limited collection of the data.

The first attempts to develop a unified database in the VEPP-4 control system were made in the beginning of the 90's, but were not successful because of the low bandwidth connection among Odernok computers and the PCs. To adopt the PostegreSQL database system for the VEPP-4 we need to start with implementing Ethernet and PCs with the Linux operating system [2].

The first goal of the new database is to provide an access to the data about all control and measuring channels of present control system for the new applications running in the PCs. There are data converters to produce special data files for initial program loading and running of the applications in each Odrenok.

The next goal is to provide channel access between new applications running in the PCs and the executive programs in Odrenoks.

The scalability of the database system should be flexible for modification in the case of addition of new accelerator control subsystems.

2 DATABASE ARCHITECTURE

2.1 General description

The VEPP-4 database includes two parts. The first part is a static dataset provided via PostgreSQL server. This part consists of relational tables with all necessary entities for control.

The second part includes the database servers providing data access and data archiving. The main part of the data access system is Application Server (AS). The AS is a dispatcher for all the requests from clients to database, control and measuring channels. If the request is to put/get data in/from control/measuring channel, the AS transmits the request to the corresponding Device Server running in the computer which is connected to the corresponding electronics. If the request is directed to executive programs working on Odrenok computer it handled via Odrenok Server.

A simplified diagram of the VEPP-4 database is shown in Fig.1.

The unique features of the database system are:

- Methods of handling a query are encapsulated into the query. This makes query handling independent of requests and allows enlarging the set of queries easily.
- The use of the benefits of QT v.3 [3], the most significant of them are:
  - Signal-slot architecture [4], [5]
  - Asynchronous sockets
  - Unified access to RDBMS

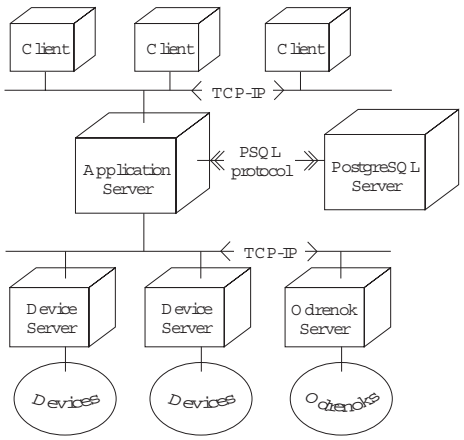


Figure 1: The VEPP-4 database diagram.

2.2 Tables

The static database based on PostgreSQL includes a set of tables with description of the electronics, control/measurement channels, objects of control, stored operation modes, etc. The main database tables are:

- **Objects.** This table includes a description of all physical and logical elements and objects in the VEPP-4 facility such as magnets, power supplies, control elements, etc. All objects are included into the hierarchical structure.
- **ObjectParameters.** This table includes different attributes and characteristics of the objects from the "Objects" table, for example: coil resistance and field/current ratio for the bending magnet.
- **Modules.** It is a description of all the control and measuring electronic modules and units.
- **Crates.** This table includes a description of the physical or logical groups of electronic modules: CAMAC crates, serial buses, etc. Tables "Modules" and "Crates" provide a convenient way for addressing to modules in programs.
- **Channels.** Each record of this table (channel) includes all necessary data for manipulation or monitoring particular physical parameters: code/units ratio, address, limits, etc. There are about 2000 control and measuring channels on the VEPP-4 not including temperature, vacuum, and beam pick-up diagnostics. There are separate tables of channels for these systems.
- **OperationModes.** This table includes a description of the stored operation modes. Each operation mode stored in a separate file according to a specified template. Each template for creating an operation mode includes a list of channels.

The number of the different table types in the VEPP-4 database is about twenty.

2.3 Object oriented scheme

An application sends a specified request (query) to get information from the VEPP-4 database. The Application Server can handle a certain set of the object related query types: get/put crate, get/put channel, etc. The VEPP-4 database is an assembly of different type objects for clients.

The VEPP-4 database object concept is shown in Fig.2.

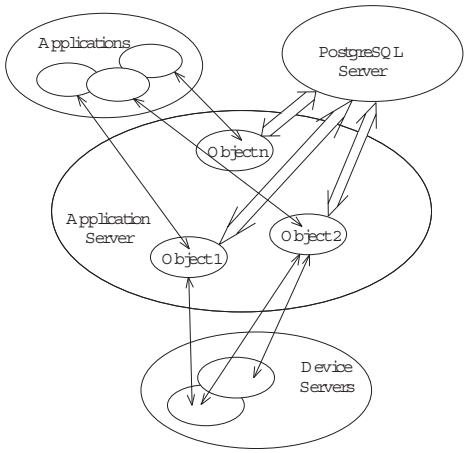


Figure 2: The VEPP-4 database object representation.

The Application Server (AS) creates corresponding objects for handling each incoming query. The objects encapsulate data from the postgres server and methods for the data processing. The objects interact with applications, device servers and postgres server via established separate connections. After the query is performed, corresponding object remains in the AS during a certain time for the possible demand in the next period.

3 APPLICATION SERVER

The Application Server is the main part of the VEPP-4 database system. The AS block diagram is represented in Figure 3.

The AS consists of several functional components. The following components are the same for AS, Device Servers and client applications:

- **Query Hash**. All requests being handled are stored in the hash and each part of the program references the query through its hash key only. This allows handling cases when referenced query was removed to be safe with a little penalty performance. But stability is much more important in this case.
- **Query Processor**. It contains common logic of request processing independent of queries and handles queues of queries being processed.

- **Queries**. Queries are a set of requests containing methods for processing in addition to data.
- **Connections.** Connections are a set of TCP connections between AS and Clients, and between AS and Device Servers.

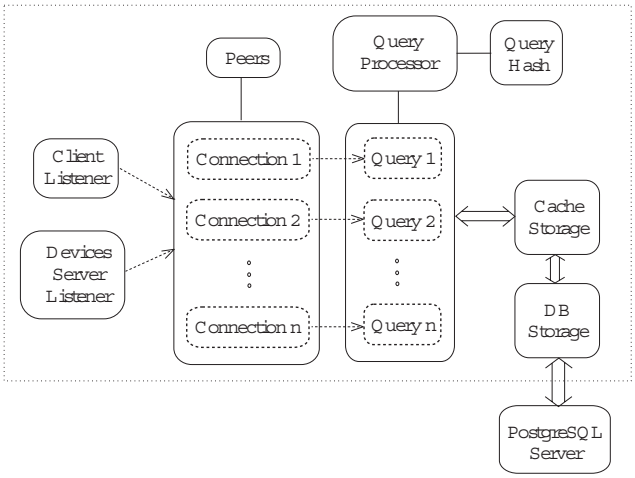


Figure 3: Application Server diagram.

Components specific to the Application Server are:

- **Storage.** It is an abstract class defining set of the operations for saving/loading persistent objects.
- **DB Storage.** It is an instantiation of the **Storage** class for implementing the database storage.
- **Cache Storage.** It is an instantiation of the **Storage** class for caching requests to the **DB Storage** for better performance.
- **Peers.** It is a set of **Connections**; it is needed for request handling.

4 DEVICE SERVERS

The Device Server (DS) is the part of database system corresponding to the data transfer between the AS and devices. The DS handles front-end processing.

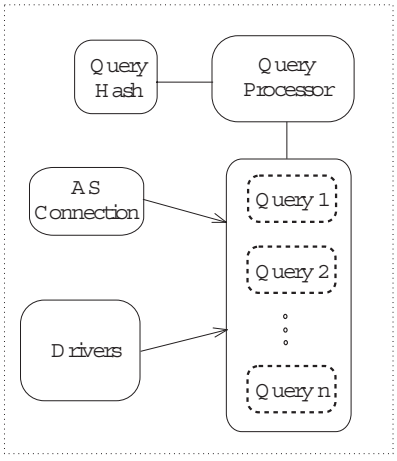


Figure 4: Device Server diagram.

The specific components of the DSs are **Device Drivers** and **AS connection**. Driver includes the logic of the device management. The AS connection is a TCP connection between DS and AS.

In the case of Odrenok Server the driver provides the query transfer from the PC to the Odrenok executive programs.

5 CLIENTS

The client applications provide visualization and monitoring, operator interface for control, data storing, etc.

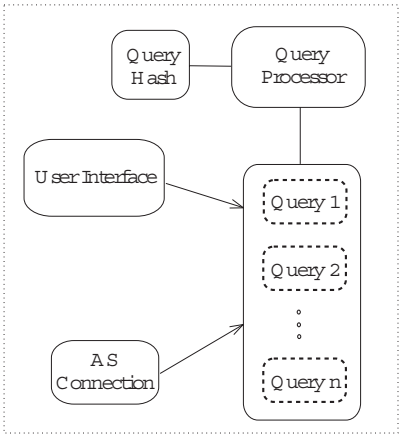


Figure 5: Application diagram.

The specific components of the Clients are:

- **User Interface**. It includes a logic implementing client specific behavior. It may communicate to an operator, or it may be an offline data-manipulating program.
- **AS connection** is a TCP connection between an application and AS.

6 CONCLUSION

The VEPP-4 database system referred above has been implemented. The first stage provides the conversion of all VEPP-4 control system data to the postgres database. This stage provides loading and running Odrenok executive programs.

The second goal to provide an accelerator operator view and control via PC applications is expected in the near future.

REFERENCES

[1] A. Aleshaev et al, “VEPP-4 Control System”, ICALEPCS'95, Chicago, USA, p.799, F-PO-2.
[2] A. Aleshaev et al, “Data Base and Data Flow on VEPP-4 Control System”, ICALEPCS’99, Trieste, Italy, p. 439
[3] QT documentation: http://doc.trolltech.com/3.0/
[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison Wesley, 1995
[5] Grady Booch “Object-Oriented Analysis and Design with Applications”, Addison Wesley, 1994

WEAP030
hep-ex/0111065

EXPERIENCE OF USING MULTIMONITOR WORKSTATIONS UNDER XFREE86 4.X IN VEPP-5 CONTROL ROOM

D.Yu.Bolkhovityanov, R.G.Gromov, I.L.Pivovarov, A.A.Starostenko
The Budker Institute of Nuclear Physics, Novosibirsk, Russia

Abstract

Modern PC workstations often provide more CPU power than required for most control applications. On the other hand, the screen space is always in short supply. One possible solution is to use more PCs, but in fact we need only more screens, not more keyboards, mice, etc. PC architecture allows using more than one videocard, and the X Window protocol is aware that there can be more than one screen. Until the release of XFree86 version 4 there was no freely available server capable of driving multiple "heads." We have been using multiheaded workstations under XFree86 in the VEPP-5 control room since early 2000 (currently 4 4-headed PCs plus several dual-headed). The "Xinerama" mode (one-large-screen) is better suited for accelerator control system than "several separate screens." When moving to this configuration we've encountered a number of, mostly human-related problems. Some of these required modifications to the X server. Additionally, the "style" of performing control has slightly changed.

1 NEED IN MORE SCREEN SPACE

Historically automation at BINP is based on CAMAC. Home-made Odrenok [1] machines were used as both crate controllers and as the main computational power. The information was displayed via CAMAC-based display controllers, which gave a 256×256 color pixel picture. That allowed sufficient display space for most tasks.

On the new VEPP-5 facility the computation and high-level control was moved from crate controllers to Intel-based workstations. So, the aging CAMAC display hardware wasn't an option.

Modern video cards and monitors have resolutions large enough to simply put the contents of all 256×256 displays on them.

This approach was taken by the VEPP-4 team, which exploits a large number of legacy programs using CAMAC display controllers. They made an emulation library, which redirects graphic output of such programs to X11 windows, but there was no reason for VEPP-5 to go this way.

2 POSSIBILITY

The PCI bus allows multiple videocards in one computer. One card is treated as the primary (the one on which the boot screen appears), and others are inactive until a multihead-aware system is loaded. The AGP slot looks like just one more PCI slot.

From the very beginning, X theoretically allowed the use of several screens on one host. These screens are referred to as *hostname:N.0, hostname:N.1*, etc., where *N* after the colon is a display number (typically *0*) and *0*, *1*, etc., after the dot is a screen number.

However, in practice, XFree86 up to version 3.x inclusive didn't support multihead. That capability appeared in a long-awaited version 4.0, released in early 2000.

3 TRADITIONAL MULTIHEAD VS XINERAMA

The traditional X multihead presents each screen separately. Consequently, when a window is created, it is placed on one of these screens, and cannot span screens, or be moved from one screen to another (see Fig.1a).

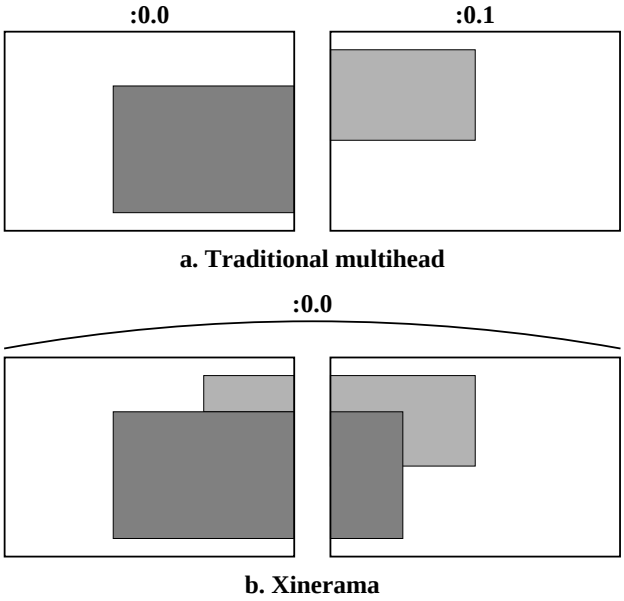


Figure 1: Traditional and Xinerama multihead

On the other hand, Xinerama makes multiple physical screens behave as a single screen, transparently to the clients (see Fig.1b) [2]. So, the windows can be freely moved between screens.

Since the situation on the screen of a control computer isn't static (there's often a need to group windows in different ways, to move more important windows to a "more visible" screen), Xinerama is much better suited for use in a control room than a traditional multihead.

4 XINERAMA PROBLEMS

4.1 Technical problems

When joining screens, Xinerama leaves only depths, which are common to all screens. So, it is impossible to join a 16-bit screen with a 24-bit one. Additionally, the 24+8 "overlay" feature of Matrox cards is lost, since the 2nd head doesn't support it.

However, the main inconvenience is that since all screens look like a single one to all clients, the window managers happily place windows between screens, maximize them on all screens, etc.

We use FVWM [4] in the VEPP-5 control room, so we invested some time in its initial xineramification, which was completed by the FVWM team (now Xinerama support in FVWM is probably the most complete and configurable among all WMs). Currently, most WMs are Xinerama-aware, but some toolkits still aren't.[1]

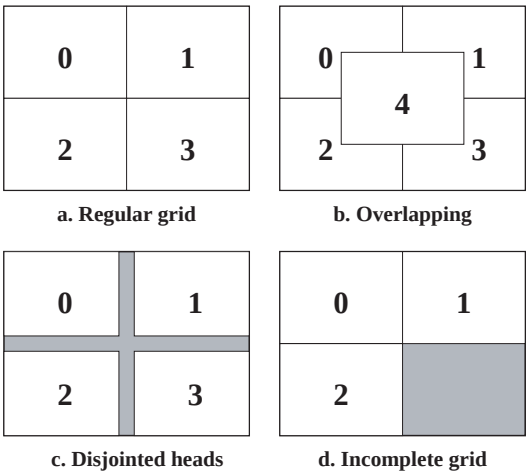


Figure 2: Possible Xinerama layouts

There is one more exotic problem. Technically, Xinerama makes a single large desktop with a size of a bounding rectangle of all screens, and screens themselves function as viewports to the desktop. So, Xinerama allows the placement of heads in many different ways (see Fig.2).

a Heads can form a regular grid – that's the most common case.

b They can overlap (a so-called "clone/zoom" mode). This is used very rarely, since the position of a "zoom" screen is fixed, and can't be moved (e.g. following the pointer).

c Can be disjointed.[2]

d Or the grid can be incomplete.

In the two latter cases, there are "black holes" on the desktop, which aren't visible on any monitor, which can't be reached with the mouse. The consequences are worst in the last case, because complete windows can disappear in the black hole.

Additionally, there are still some problems with the software, which either requires a direct access to a framebuffer, or uses a fullscreen mode (various video capturing and movie playing programs).[3] However, thanks to XVideo extension, these problems became very rare.

4.2 Human Problems

Some of our software developers are greedy: when they see so much display space, they say: "Hey, let's move this and this to another screen, and my program will just fill this screen." The common rule is "some programs tend to grow to occupy all screen space." So, the appetites of some people need reduction.

Another problem is that mouse pointers often gets lost on a large screen space. Finally, a patch for X server was developed [3], which allows 1) doublesizing the pointer and/or 2) changing the default colors from black&white to something more visible. We use red doublesized pointers, which provides good visibility.

5 HOW MANY HEADS TO USE

The most common multihead layout in the world is two heads: side by side horizontally, or one above another (if 2nd head is used rarely).

Three heads are hard to use: the layout will either be as Fig2d, which is inconvenient, or lined up. In the latter case it takes too much time to move the pointer between the first and the last screens. When we *had* to use three heads, we put monitors in the shape of an "r," but the X layout was "three heads vertically." That setup was extremely confusing for operators.

Four heads give the best balance between the "as much screen space as possible" principle and convenience of use. When used in a 2×2 grid, as shown in Fig.2a, there are no black holes, and the distance between heads is small.

6 HARDWARE

6.1 Criteria for selecting video cards

First, hardware should be multihead-capable (e.g., 3Dfx cards are known not to work in multihead mode under XFree86 at all). Second, it must have good support in XFree86 and be very stable. Third, it should produce an

[1] A frequent case: a window with Yes and No buttons is centered, so that [Yes] goes to one monitor and [No] to the other one.

[2] That's a pathological case; more often screens of different sizes are used (e.g. 1024×768 and 800×600), which has the same effect.

[3] Up to XFree86 4.1 Xawtv behaved very funny: the window frame could be on one screen, and undecorated video picture – on another.

excellent picture, and have good 2D performance (3D isn't important). Fourth, video hardware should occupy as few PCI slots as possible.

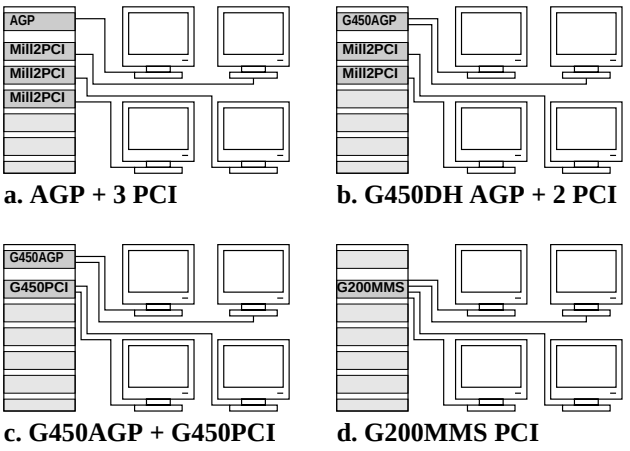
6.2 Solution we use

There were 4 main manufacturers: ATi, Matrox, nVidia and S3 (the latter is almost dead now). We chose Matrox because it satisfied all our criteria, and we already had very positive experience with their products.

The product nVidia doesn't provide specifications for their cards, so the XFree86 driver is very lacking. Instead nVidia provides a binary-only driver. ATi cards are not-so-good, and there are myriads of subversions, which affects stability of the driver.

We use MilleniumIIs as PCI cards, but any PCI card with 4M or more memory[4] would do (Millenium, G100, G200).

6.3 Multiheaded videocards



a. AGP + 3 PCI **b. G450DH AGP + 2 PCI**

c. G450AGP + G450PCI **d. G200MMS PCI**

Figure 3: Hardware options for 4 heads

One more advantage of Matrox cards is that since 1999 they have two heads on one card (G400DH, G450, G550). So, to have 4 heads, an AGP G450 plus two PCI MilleniumIIs were enough (see Fig. 3b), and only two PCI slots are used (which are always in deficite in control machines).

Matrox produces a PCI version of G450, which allows the use of only one PCI slot in addition to the AGP slot (see Fig. 3c). Unfortunately, when running as a non-primary card, the G450 (either AGP or PCI) requires an additional driver module, which is available as binary-only from Matrox (so-called HAL module [5]). However, we hope the native XFree86 support will soon become better, thus making HAL redundant.

Currently two of our 4-headed PCs are equipped with G450AGP+2×MilleniumII, and two are G450AGP+G450PCI.

Theoretically, there exists even a better choice – G200MMS, which supports 4 heads on one card (see Fig. 3d). However, it exists in a PCI version only, so if we need 4 heads total, it occupies the same one PCI slot as the G400AGP+G450PCI. Additionally, G200MMS is almost impossible to find in Russia.

6.4 Motherboards

We chose the ASUS P3B-F (Intel 440BX chipset), which has 1 AGP slot, 6 PCI slots, and one ISA slot (one position is shared). So, you get either 6 PCI slots, and 0 ISA slots, or 5 PCI slots, and 1 ISA slot. The main requirement was a presence of an ISA slot, since we still use old ISA hardware.

7 FUTURE ENHANCEMENTS

One feature our operators wish to have is the ability to control programs on adjacent computers with their mouse. A program *x2x* [6] exists which does exactly this, but it doesn't work with Xinerama. So, we plan to "xineramify" *x2x*.

Currently, the world is moving toward the use of TFT monitors, as they are safer for people. However, most TFTs have a limited viewing angle, which is inappropriate in multiheaded system, and those TFTs which are okay (like SGI 1600SW) are too expensive. So, currently we use 17" CRT displays, but plan to replace them when affordable TFTs appear.

Finally, consistent and ergonomic placement of windows on a 4-monitor desktop is a time-consuming task. So, we are planning to implement some sort of automation for this. Currently, we are experimenting with the X resource database (the *WINDOWNAME.geometry* resource).

8 REFERENCES

[1] G.Piskunov, "CAMAC-embedded 24-bit computer", Autometriya, N4 (1986) pp. 32-38 (in Russian).

[2] Dennis Baker, "Using the Xinerama Extensions to MultiHead XFree86 V. 4.0+", November 2, 2000. http://www.linuxdoc.org/HOWTO/Xinerama-HOWTO.html

[3] D.Yu. Bolkhovityanov, "Visible Cursor Patch for XFree86 4.0.3", http://www.inp.nsk.su/~bolkhov/files/bigcursor/

[4] Official FVWM Homepage, http://www.fvwm.org/

[5] Matrox Graphics, Inc., "Matrox beta drivers for XFree86", ftp://ftp.matrox.com/ pub/mga/archive/linux/2001/beta_133_143/

[6] David Chaiken, "X to X connection", http://gatekeeper.dec.com/pub/DEC/SRC/x2x/

[4] 1152×864 @ 32bpp ≈ 4M RAM for framebuffer.

WEAP031
physics/0111193

AN AUTOMATIC VALIDATION SYSTEM FOR INTERFEROMETRY DENSITY MEASUREMENTS IN THE ENEA-FTU TOKAMAK BASED ON SOFT-COMPUTING[1]

G. Buceti, ENEA, Frascati, Italy
L. Fortuna, A. Rizzo, Università degli Studi di Catania, Catania, Italy
M.G. Xibilia, Università degli Studi di Messina, Messina, Italy

Abstract

In this paper, an automatic sensor validation strategy for the measurements of plasma line density in the ENEA-FTU tokamak is presented. Density measurements are performed by a 5-channel DCN interferometer. The approach proposed is based on the design of a neural model of the observed system., i.e. a model able to emulate the behavior of a fault-free sensor and of a two-stage fuzzy system able to detect the occurrence of a fault by using a set of suitable indicators. The fault diagnosis and classification is also accomplished. The validation strategy has been implemented and embedded in an interactive software tool installed at FTU. Statistics concerning the rate of fault detection agree with the rate of uncertainty usually achieved in the post-pulse manual validation.

1 INTRODUCTION

With the continuous growth of the number of sensors installed in tokamaks and other big experimental physics plants, reliability of measurements has become a fundamental issue, both for feeding the control systems with reliable measurements and for physicists to analyse the real physics of the experiment. At present, most of the work is carried out manually by the experts responsible for the single diagnostic, which is a very time-consuming activity. Moreover, knowledge about the validation process is spread among experts, each of them caring about peculiar aspects of the phenomena. Due to the nature of the knowledge, the huge amount of data stored and the heuristic involved, a suitable approach for the automatic validation of measurements has been individuated in the soft-computing-based techniques [1, 2]. Some results have recently been achieved by the authors at JET, concerning the validation of the measurements of the vertical stresses on the bottom legs of the vacuum vessel during Vertical Displacement Events (VDEs) occurring at disruptions [3, 4]. This papers deals with the design and implementation of an automatic validation tool for a 5-channel DCN interferometer installed at FTU-ENEA, Frascati.

The system is highly modular and organized in a hierarchical structure, providing advantages in terms of feasibility, flexibility of the validation actions, scalability and ease of upgrading.

2 THE VALIDATION SYSTEM

The first step in the design of the validation system consists of collecting information about the manual validation performed by experts. After thorough investigation, the following set of qualitative rules has been individuated:

1. line density is mainly related with the plasma current and with the quantity of gas injected into the tokamak;
2. line density is higher in the central area of the plasma column;
3. line density has a similar temporal trend on all channels (and similar to the plasma current trend, too);
4. line density trend is generally smooth: relevant discontinuities (step-like or spikes) are not admitted.

Based on this set of rules and on the wide set of experimental data available at FTU, the validation system reported in Fig. 1 has been designed. This is a modular system, consisting of two hierarchical stages.

The first stage consists of three main streams, each of them taking into account different aspects of validation coming from the rules illustrated above.

The first stream takes into account rules 1. and 2. With this goal, neural-network-based NARMAX models of each healthy DCN channel have been designed, based on a large set of manually validated data.

[1] Paper supported by MURST project 'Fault Detection and Diagnosis, Supervision and Control Reconfiguration in Industrial Process'

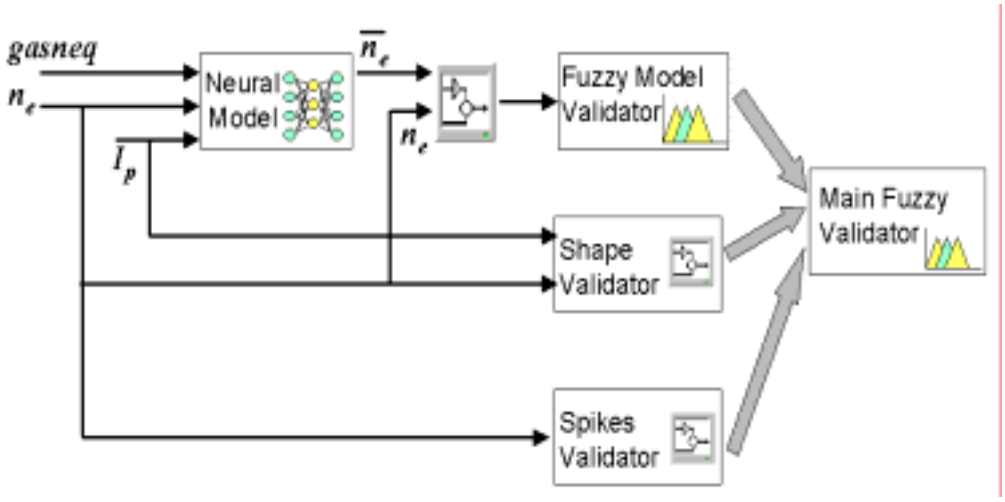


Figure 1: Block scheme of the sensor validation system.

Taking into account, expert knowledge, through a wide series of network training phases, the following model has been considered for each channel:

$$d(k+1)=f(I_p(k+1), I_p(k), gas(k+1), gas(k), d(k), d(k-1)) \quad (1)$$

where *k* is the discrete time (sampling time 10 msec), *d(k)* is the plasma density, *Ip(k)* is the plasma current, *gas(k)* is the total amount of gas introduced up to time *k*. The function *f* has been approximated by a Multi-Layer-Perceptron [1], trained with a wide set of healty experimental data. Once the network has been trained, it can be used to detect the occurrence of faulty measurements by evaluating the residual between the output of the network and the actual measurement. As illustrated in Fig. 1, to increase the robustness of the residual evaluation, two further blocks are added. These are a simple signal processing block and the *Fuzzy Model Validator*. The signal processing block evaluates the average value of the residual (*avgerr*) and the maximum time interval in which the residual exceeds a certain threshold (*outbound*). The *Fuzzy Model Validator* is designed to provide a robust evaluation of the residual, by means of the following rule, in which the linguistic values are represented by suitable *fuzzy sets* [1].

R1: if *avgerr* is *zero* and *outbound* is *small* then *model* is *ok*

R2: if (*avgerr* is *positive* or *avgerr* is *negative*) and *outbound* is *small* then model is *warning*

R3: if (*avgerr* is *poshigh* or *avgerr* is *neghigh*) then *model* is *faulty*

R4: if (*avgerr* is *negative* or *avgerr* is *zero* or *avgerr* is positive) and *outbound* is *high* then *model* is *faulty*

R5: if (avgerr is *negative* or *avgerr* is zero or *avgerr* is *positive*) and *outbound* is *medium* then *model* is *warning*

The linguistic variable, *model*, indicates therefore whether the model output is in agreement with the sensor output, thus indicating that the sensor is working properly or not.

The second stream, *shape validator*, takes into account rule 3. The block evaluates the correlation between the trend of the plasma current and that of the line density coming from each channel of the interferometer and generates an indicator called *xc*.

The last stream, *spikes validator*, is designed to detect the occurrence of spikes or great discontinuities in the trend of measured line density. The block functionality is based on digital filters. The result of this validation is provided through an indicator called *spikes*.

Indicators coming from each of the three streams are fed as inputs to the second stage, *Main Fuzzy Validator*, in which a set of fuzzy rules combines all of the partial validations performed by the previous stages. As a result, each channel is globally evaluated.

The set of fuzzy rules adopted for the global validation is the following:

R1: if *xc* is *high* and *model* is *ok* and *spikes* is *low* then *sensor* is *ok*

R2: if *xc* is *not-high* and *model* is *faulty* and *spikes* is *not-low* then *sensor* is *faulty*

R3: if *xc* is *low* and *model* is *ok* then *sensor* is *warning*

R4: if *xc* is *high* and *model* is *warning* and *spikes* is *low* then *sensor* is *ok*

R5: if *xc* is *medium* and *model* is *warning* then *sensor* is *warning*

R6: if *model* is *warning* and *spikes* is *not-low* then *sensor* is *faulty*

R7: if *xc* is *high* and *model* is *faulty* and *spikes* is *low* then *sensor* is *faulty*

R8: if *model* is *ok* and *spikes* is *not-low* then *sensor* is *warning*

R9: if *xc* is *low* and *model* is *ok* then *sensor* is *faulty*

R10: if *model* is *ok* and *spikes* is *very-high* then *sensor* is *faulty*

R11: if *xc* is *medium* and *model* is *ok* then *sensor* is *warning*

R12: if *model* is *ok* and *spikes* is *high* then *sensor* is *warning*

R13: if *xc* is small and *model* is *warning* then *sensor* is *faulty*

R14: if *xc* is *medium* and *model* is *faulty* and *spikes* is *not-low* then *sensor* is *faulty*

R15: if *xc* is *high* and *model* is *warning* and *spikes* is *medium* then *sensor* is *warning*

It can be observed that a status of *warning* has been introduced, used to gain attention of the experts for a thorough analysis of the measurements of the shot.

Table 1 reports statistics on the performance of the validation tool, both in terms of *successful validation* (healthy measurements classified as healthy), and *successful fault detection* (faulty measurements classified as faulty), for each DCN channel, for a wide set of shots taken at different operational conditions.

Table 1: Margin specifications

Channel #	Successful Validation	Successful Fault Detection
1	92.4%	95.5%
2	90%	95.2%
3	94.3%	72.2%
4	90.9%	98.1%
5	85.3%	85.9%

CONCLUSIONS

In this paper, a sensor validation tool for the DCN interferometer in the FTU tokamak in Frascati, Italy, has been presented. The tool is based on a cascade of a neural model and two fuzzy blocks. The neural model realises a NARMARX model of the plasma line density, taking as inputs the plasma current and the quantity of gas introduced into the tokamak. The output of the model is used to compute a set of suitable indicators which are used in a fuzzy block which produces a first validation of the sensor output. Subsequently, the information coming from the first fuzzy block is integrated in a second fuzzy validator to give a final validation of the sensor output. The designed validation tool has been tested on a large set of FTU data. The statistics on the overall performance have been reported in the paper for a large set of experiments, confirming the suitability of the approach.

REFERENCES

[1] L. Fortuna, G. Nunnari, G.G. Rizzotto, M. Lavorgna, M.G. Xibilia, R. Caponetto, *"Soft Computing"*, Springer, 2001.

[2] R. Isermann and P. Ballé, Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes. Control Eng. Practice, vol. 5, no. 5, pages 709—719, 1997.

[3] L. Fortuna, A. Gallo, A. Rizzo, MG Xibilia, 'An Innovative Intelligent System for Fault Detection in Tokamak Machines', *Proceedings of ICALEPCS 99, International Conference on Accelerators and Large Experimental Physics Control Systems, Trieste, Italy, Oct. 99*

[4] L. Fortuna, V. Marchese, A. Rizzo, M.G. Xibilia, 'A Neural Networks Based System for Post Pulse Fault Detection and Data Validation in Tokamak Machines', *Proceedings ISCAS 99 conference, IEEE International Symposium on Circuits and Systems, Orlando, FL, June 1999*

WEAP032
physics/0111033

UPGRADE OF THE CONTROL SYSTEM OF THE IFUNAM´S PELLETRON ACCELERATOR

R. Macías, E. Chávez, M. E. Ortiz, K. López, A. Huerta, IFUNAM, A. P. 20-364, México 01000
M. C. Verde, Instituto de Ingeniería UNAM, México

Abstract

In 1995 a 9SDH-2 Pelletron from NEC was installed at IFUNAM (Instituto de Física, Universidad Nacional Autónoma de México). Two beam lines have been operational since then and two new lines have been built. In order to implement planned projects in this mature facility, an upgrading of the original manual control system is required. The proposed new control system takes advantage of the existing devices and incorporates the electronics needed for the newer beam lines. The control software from NEC, has been modified to accommodate the larger requirements. It runs on the same dedicated computer but receives commands from a new installed host. Both computers communicate through a local network sharing the accelerator database. The new host computer also handles all parameters related to the new lines. In the future, the old computer will be replaced in order to expand the possibilities of the system and use a friendlier graphical interface. In this work we present the changes made to the control software, the new acquisition and control devices installed and the integration of the old and new systems. Finally, the beam control procedure based on the analysis of the beam profile that is under development is also shown.

INTRODUCTION

After arrival and installation of the IFUNAM's pelletron, the need for additional beam lines was soon recognized. The first two, those that originally came with the machine, are designed to be dedicated to Rutherford Back Scattering (RBS) Analysis and Ion Implantation respectively. Two new lines were proposed, one to be dedicated to Particle Induced X-Ray Analysis (PIXE) and a second for Nuclear Reaction Studies and Mass Spectrometry.

The original computer control system must be expanded to be able to handle the new hardware added to the system. Two options were envisaged: build a whole new control system or modify the existing one. The former requires more resources both in time, money, skilled personal and deep knowledge of the manufacturer's design. The last one, less ambitious, takes advantage of the existing software and hardware and is easier to implement. Additionally, from the standpoint of the operator, most of the operations remain unchanged. We decided to take this line of work, at least in a first stage, keeping in mind that a new system could be built in the future.

SYSTEM OVERVIEW

The control system [1] acts primarily as a switchyard which takes incoming signals from the accelerator and displays the parameters on the screen and or meters. The operator initiates control of the various parameters. The control system software, developed using MS Visual Basic for DOS, includes: a) The control system program to interact with the operator and database to provide control and monitoring of the accelerator and b) a number of utilities programs.

Each component of the accelerator has a unique label and reference number that may be associated with a process control station address, a special control/read, a bypass or an interlock record. Each record has two separate files or database. One file is for data that does not change (cs1.db) and the other is for items that change (cs.db). These records hold the information on address, slot, size, type, etc. for the control system program to control and/or monitor all data points in the system.

THE MODIFICATIONS

A second computer was added (PIII, 128MB RAM, hereupon referred as S2), partly because the original program only runs properly in its host computer (Digital 486, 4MB RAM, we will call this S1 from now on). Both computers were connected through an Ethernet link in such a way that S2 could access the working directory of S1.

The original program from NEC on S1 was modified in order to allow the program running on S2 to remotely interact with the database in S1.

The program on S1 remains the only one with direct access to the database. The interaction between S1 and S2 programs is made through two new files (updrec.db and readrec.db), so S2 can request the modification or readout of the value of any system parameter. Two new procedures were added to the program in S1, to force it to act according to the requests from S2.

In this way the system can be controlled from an external computer, which allows an easy way to expand the original control system, adding additional software in the external computer without any other modification of the original one.

As an example the Fig. 1 shows the front panel of a procedure (written in LabVIEW 5.1 from National Instruments Corp.) for updating any parameter of acclerator. The control rec number identifies the accelerator parameter (i.e. 78 means Injector Magnet Current) and pdatareal is the real value we want to update.

Figure 1: Front panel of the updating procedure.

ONE ADDITIONAL BEAM LINE

Here we describe one of the new beam lines added to the accelerator: the isotope separator.

The 45° port of the switching magnet was selected for it. The separator consists mainly of electromagnetic devices for the transport of the beam, together with beam control and diagnostics as shown schematically in Fig. 2.

These additional elements are controlled by S2 through FieldPoint modules from National Instruments Corp., connected to an RS232 port by optical fiber.

Besides all the components of the new line, three teslameters model DTM-141 from Group3 Technology Ltd., were added to the accelerator system to have a precise measurement of the magnetic induction in the injector, switching and 90° magnets.

Further expansion of the laboratory: new beam lines or additions to the existing ones, can be computer controlled following the procedures here described.

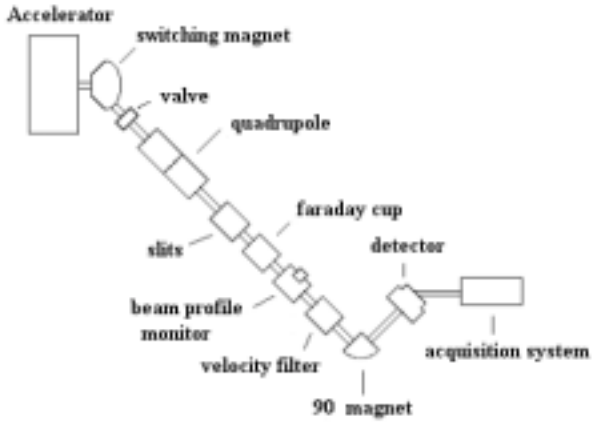


Figure 2: Isotope Separator on the 45º port.

BEAM CONTROL AND DIAGNOSIS

The accelerator is equipped with beam profile monitors model BPM-80 from NEC at the low energy side and in all of the beam lines. This kind of device allows continuous monitoring of the intensity distribution and the position of the beam [2].

Its working principle is the collection of secondary electrons released from a rotating wire that sweeps across the beam twice in each cycle, at a constant speed. The resulting signal (X and Y profiles) is amplified and sent to the control console, together with fiducial marks. These marks are time signals generated internally by the BPM. All these signals are displayed on an oscilloscope (see Fig. 3).

Figure 3: Profile with marks as observed on the control console oscilloscope.

The machine operator uses the scope image to obtain the desired characteristics of the requested beam (focusing and position) by looking at the width, height and location of the peaks relative to the fiducials.

Commonly the intensity of the beam is measured through the use of a Faraday Cup (FC). This is required to stop the beam. A new procedure is developed here to use the BPM signals to obtain information about the beam intensity, with the advantage of not having to interrupt it.

An electronic shaping circuit has been designed. This circuit will allow the acquisition of signals from two

BPMs by the S2 computer, equipped with a PCI-6024E card from National Instruments Corp.

Using a 3 MeV proton beam, 135 profiles similar to Fig. 3 were recorded for different combinations of focus, cathode, extraction and bias voltages. These are all the relevant parameters of the SNICS ion source. For each record the beam current (Ifc) was measured with a FC.

Fig. 4 shows a plot of the measured current (Ifc) and a value extracted by the product of the integrals of the two peaks in the profile (Int(x) * Int(y) * N). The normalization factor N depends on the beam species, energy, amplifiers gain, etc.

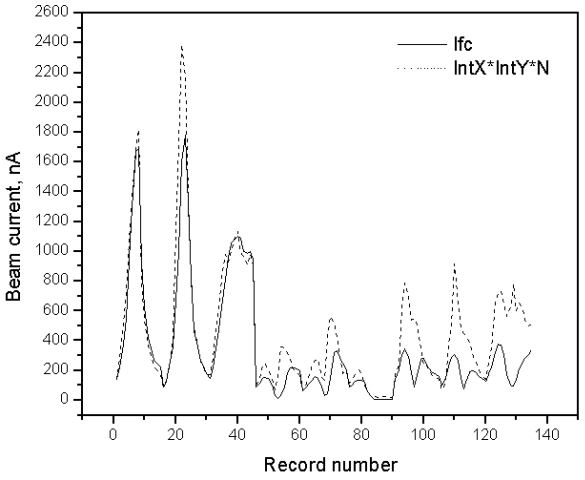


Figure 4: Comparison between the measured and calculated beam currents.

One can notice that both curves follow each other closely up to record 45. From this point an important deviation appears, which can be explained in terms of a displacement of the beam off the center of the geometrical axes, these were not taken into account during calculations.

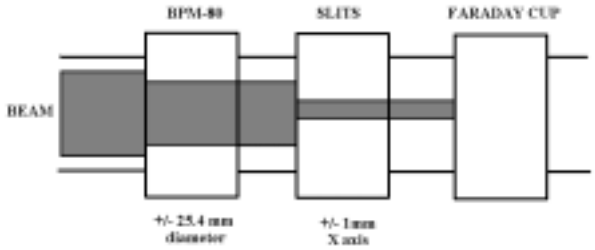


Figure 5: BPM, slits and Faraday Cup as located in the low energy side of accelerator.

The integration limits needed above are determined mainly by the aperture of the slits and BPM as shown in Fig. 5.

Data from the first 45 records is shown in Fig. 6, where the calculated beam current from the BPM is plotted versus the Ifc. The straight line shows the best fit to the data.

This behavior allows us to conclude that the beam current can be measured using the BPM, avoiding interruption of the beam and permitting continuous and controlled operation of the experiment.

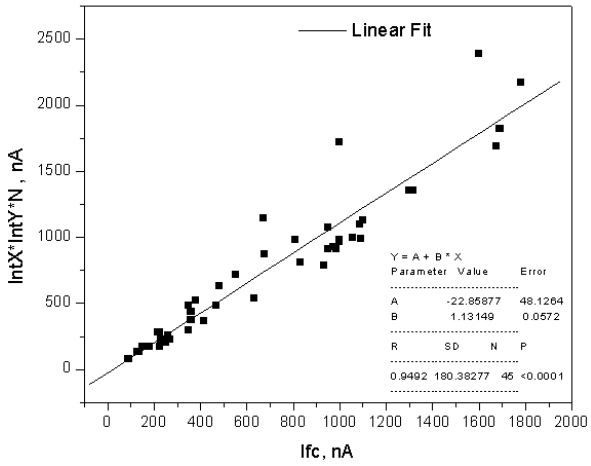


Figure 6: Correlation between the measured and calculated beam currents.

It is worth mentioning that the position of the fiducial marks on each profile is critical and, in order to get better results, they should be accurately measured for every cycle.

ACKNOWLEDGEMENTS

The authors wish to thank the IFUNAM's Pelletron support personnel, specially Francisco X. Jaimes, without whom this work would not have been possible. This work was supported by UNAM through contracts DGAPA IN114896 and CONACyT 32262-E

REFERENCES

[1] T. R. Luck, "Instruction Manual for Operation and Service of the Small Accelerator Control System", National Electrostatics Corp., Jan. 1993.

[2] Instruction Manual for Operation and Service of Beam Profile Monitor Model BPM80", National Electrostatics Corp., Aug. 1995.

WEAP033
physics/0111050

DEVELOPMENT OF AN INTERPRETER PROGRAM FOR AN EQUIPMENT CONTROL

Y. Furukawa, M. Ishii, T. Nakatani, T. Ohata
SPring-8, Hyogo 679-5198, Japan

Abstract

A message oriented software scheme is used for the SPring-8 beamline control [1]. In the early stage, we developed customized control programs for each beamline. As the number of beamlines increased, however, it became necessary to introduce a general software scheme that could handle the control sequence for every beamline. We developed a new server program, Command Interpreter (CI), as a software framework. The CI interprets a high-level compound message issued from client programs and decomposes it to a set of primitive control messages. The messages are sent to the VME computers afterwards. For flexibility, the operation sequences specific to the individual beamline component are standardized and defined in the interface definition files. Using the CI, the response time overhead is reduced to 26% and the XAFS measurement time also decreases by about 50%.

1 INTRODUCTION

SPring-8 was successfully dedicated to public use on October 1997. The beamline control software of the SPring-8 is based on the same framework of the SPring-8 storage ring control software architecture [1]. A control message is sent from X11 based graphical user interface (GUI) programs to a message server (MS). The MS forwards the message to equipment managers (EM). These processes communicate by exchanging the messages formatted as "S/V/O/C" [2]. The EM manages only individual I/O channels such as stepper motors, analog I/Os and digital I/Os. Complex sequence operations, such as the correlated stepper motor operation of the standard X-ray monochromator [3], are achieved by GUI built-in sequence routines. Users conducting experiments should access the beamline control system not only via the local GUI terminal but also via remote computers for their experiments. The GUI process receives a message from the user's computer in order to change the X-ray energy by moving the monochromator for example. The GUI accepts a single message or compound message from the user's computers.

Optical components, such as a monochromator and mirrors, have been standardized for the public beamlines. The control sequences and parameters were built-in software code in the GUI programs. However, a more flexible framework was required to maintain not only the control parameters but also the sequences for the optical component especially for the contracted beamlines, because non standard components have been introduced.

The GUI programs communicate with the MS with one message, i.e. the GUI program sends a message then waits for a return message. This ensures the result of the request message, but the GUI cannot respond to an X event or user messages while waiting for a response message from the EM.

As a result of this design, the response of the GUI to the user's computer was fairly slow in the case of spectroscopy beamlines, such as X-ray absorption fine spectrum (XAFS) beamlines.

The command interpreter is introduced to the SPring-8 beamline control system to improve the response time of the control system and to introduce flexibility.

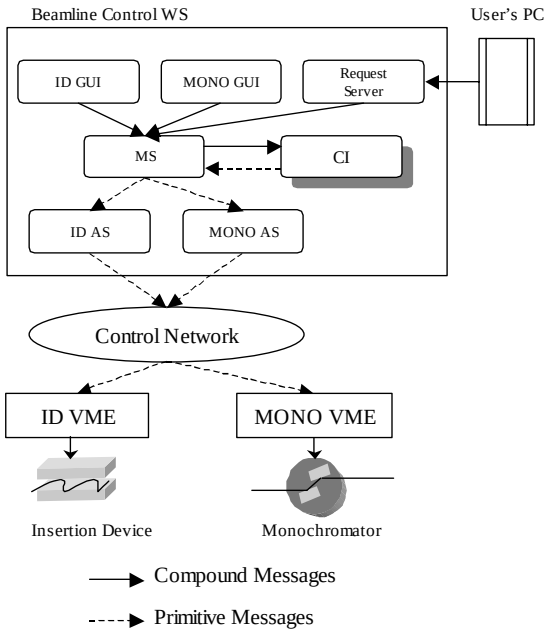


Figure 1: Message flow diagram.

2 DESIGN OF THE COMMAND INTERPRETER

2.1 Conceptual Design

The Command Interpreter (CI) is designed as a server process that communicates with the MS only. The internal structure of the CI is message driven. The CI accepts a compound message from client processes such as a GUI process or remote user's computer and interprets the message into primitive messages which the EM can accept. The primitive messages are sent to the MS at first, then to the EM (Figure 1). The CI does not wait for the returned message from the EM after it sent primitive messages. The EM executes the primitive message to move a stepper motor for example.

Information peculiar to the individual component is written in an interface definition file ("configuration file") described in the following subsection.

For example, one can control a monochromator by sending a message to the CI to change a Bragg angle, X-ray energy or X-ray wavelength. Referencing the configuration file for the monochromator, the CI interprets the message and calculates the pulse position of the stepper motors. Then the CI sends primitive messages to the EM.

2.2 Configuration Files

The configuration file contains component specific information. A simplified configuration file for the standard monochromator is shown in Figure 2 for example.

The configuration file consists of one or more "sections". Each section contains a definition of a high-level object such as monochromator or mirror. A section consists of the object name definition entry (line 2 in Figure 2) and subsections. Each subsection has entries, each entry consists of the entry name and following parameter(s). In a "subsection object", low-level objects that the EM can accept are defined (lines 4-8). In a "subsection function" (line 27-33), how the CI interprets the compound message and decomposes it into the primitive messages are defined.

2.3 Example of Message Flow

As an example, one can send a compound message "S/put/bl_01b1_mono/5.6degree" to the CI in order to set the Bragg angle of the monochromator. The CI interprets the message and decomposes it into primitive messages, "S/put/bl_01b1_mono_theta/10800pulse" and "S/put/bl_01b1_mono_y1/-153715pulse".

To obtain the present Bragg angle, one can send a compound message "S/get/bl_01b1_mono/angle" to the CI. The CI sends a primitive message "S/get/bl_01b1_mono_theta/position" then obtains the reply "S/bl_01b1_mono_theta/get/S/10800pulse". Evaluating the line 38, the CI returns a reply "S/bl_01b1_mono/get/S/5.6degree".

```
 1  SECTION OBJECT
 2    objectname bl_01b1_mono
 3
 4    SUBSECTION OBJECT
 5      objectname theta
 6      sendcomplement position
 7    ENDSUBSECTION
      ……..
15    SUBSECTION CONT
16      constname y1_coeff
17      type double
18      value 1000.0
19    ENDSUBSECTION
      ……..
27    SUBSECTION FUNCTION
28      apply put/%fdegree
29      range 3.0 27.0
30
31      function theta [target()*theta_coeff] pulse
32      function y1 [-15.0/sin(target())*y1_coeff] pulse
33    ENDSUBSECTION
34
35    SUBSECTION FUNCTION
36      apply get/angle
37
38       function [theta/theta_coeff] degree
39    ENDSUBSECTION
40  ENDSECTION
```

Figure 2: Example of configuration for the monochromator.

Figure 3 shows a message flow diagram as a comparison of conditions before introducing and after. The CI is designed to send primitive messages to move the low-level objects without waiting for returned messages from the EM. The CI is also designed to obtain the position of the low-level objects periodically when they are moving. Thus the CI can respond immediately to the client message whenever the client asks the the monochromator for the current angle. As seen from Figure 1, the response time to the client is reduced, and the start time difference between the theta1 and y1 axes is shortened.

3 APPLICATIONS

The SPring-8 standard monochromator, based on a double crystal monochromator, is controlled under two axis correlation mode. One is for the Bragg angle control and the other for the first crystal position control [3]. These two axes are driven by stepper motors. The configuration file for the standard monochromator is longer than that shown in Figure 2, because it needs additional subsections for the messages to obtain the current X-ray energy and wavelength. However the length of the configuration file is much shorter and simpler than if it were written in C-language, i.e. the number of lines is reduced to about 1/5.

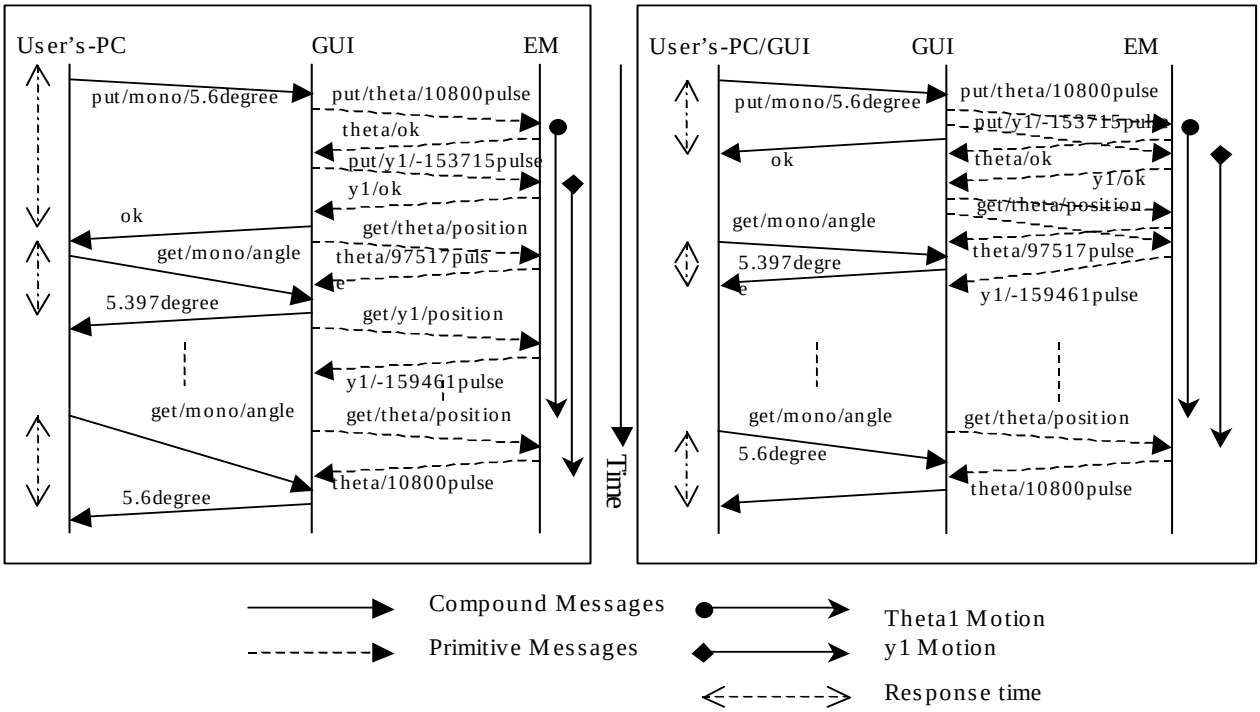


Figure 3: Message flow diagrams. Left: before introducing the CI, right: after introducing the CI.

There are many kinds of mirrors in the SPring-8. The CI laps these differences and the clients' software can control every mirrors with a same manner. We can adapt the CI to the new type mirror by slight modification of the configuration file.

4 PERFORMANCE IMPROVEMENT

The CI was initially applied in the monochromator of BL15XU, a contract beamline of the Advanced Materials Laboratory. The BL15XU monochromator is a non-standard monochromator in the SPring-8 because it covers a wider X-ray energy region. The CI laps the differences of the monochromator from clients' software. We made minor adjustments to the GUI programs to enable the standard monochromator to be used for the BL15XU. Before introducing the CI, it took about 30 minutes to measure a Cu-K XAFS spectrum (8970eV to 9010eV by 0.2eV step). After introducing the CI, the time was reduced to 15 minutes.

Control of BL01B1, a XAFS beamline, has also been upgraded using the CI. The CI controls a monochromator, X-ray mirrors and slits. The time needed to move the monochromator Bragg angle from 8.0 degrees to 8.1 degrees in 0.001 steps was measured on an HP/B2000 workstation with an HP-UX 10.20. The CI improved the moving time from 80sec to 46sec. The total measurement time, the sum of X-ray intensity measurement time and the moving angle time, was improved from 180sec to 146sec.

We also measured response time to the remote user's computer. The mean response time to get the Bragg angle of the monochromator was reduced from about 150msec to 40msec.

5 CONCLUSION

The CI was successfully introduced into the SPring-8 beamline control system. It made installation of the new component to the system easier and improved measurement time. Seven beamlines are already being operated using the CI. We plan to upgrade the control software using the CI for the existing beamlines and apply it to the newly constructed beamlines.

The CI is flexible and can be applied to other equipment controls, such as diffractometers, goniometers and detectors.

REFERENCES

[1] T. Ohata *et al.*, "SPring-8 Beamline Control System", ICALEPCS'99, Trieste Italy, (1999) p.666.

[2] R. Tanaka *et al.*, "Control System of the SPring-8 Storage Ring", ICALEPCS'95, Chicago, USA, (1995) p.201 .

[3] M. Yabashi *et al.*, "SPring-8 standard x-ray monochromator", Proc. SPIE 3773, 19 July 1999, Denver, Colorado, U.S. (1999) 2.

WEAP034
physics/0111056

CONVERTING EQUIPMENT CONTROL SOFTWARE FROM PASCAL TO C/C++

L. Hechler, GSI (Gessellshaft fuer Schwerionenforschung), Darmstadt, Germany

Abstract

The equipment control (EC) software of the GSI accelerators has been written entirely in Pascal. Modern software development is based on C++ or Java. To be prepared for the future, we decided to convert the EC software from Pascal to C in a first step. Considering the large amount of software, this is done automatically as far as possible. The paper describes our experiences gained using a Pascal to C translator, Perl scripts, and, of course, some manual intervention.

1 MOTIVATION

The EC software comprises the device representation layer, the real-time layer, and the device drivers [1]. Except for some assembler code, it has been written entirely in Pascal.

For embedded applications there are no integrated cross development systems that currently support Pascal. The system we use runs under VMS and its support expires completely by the end of 2001.

However until now we invested about 40 person-years in developing and maintaining the EC software. A lot of special know-how has gone especially into the real-time layer. The functionality gained in this work must be preserved.

Future control system developments have to be implemented with modern object-oriented methods. Appropriate up-to-date tools are based on C++ or Java nearly without exception.

Existing hardware (400 VME boards) must be used in the future as well since it cannot be replaced completely due to cost reasons. And, last but not least, the conversion must not affect the day by day accelerator operation.

2 CONVERSION

We decided to convert the EC software from Pascal to C in a first step. This allows us to "re-use" the software on one hand and to establish a basis for re-engineering the control system with modern methods and tools [2] on the other hand.

Considering that EC software consists of about 170 000 lines of code (LOC), comments not counted, it is clear that conversion has to be done automatically as far as possible.

The basis for a conversion is EC software for one device class. There are 61 different device classes, each one controlled by dedicated software. To ease the conversion, we issued a cookbook [3] that describes the process step by step.

To convert the Pascal code into C automatically we use the Pascal to C translator p2c.[1] Perl scripts are then used to adapt the notation of identifiers to our style guide.

In spite of the automation there is a lot of manual intervention required. Besides the preparations for p2c and Perl there are four essential reasons that make manual interaction necessary.

2.1 Compatibility of Data Structures

The p2c manual pages state that "most reasonable Pascal programs are converted into fully functional C which will compile and run with no further modifications". This may be true for stand-alone programs. Given the EC software, it has to be taken into account that in case of communications with other modules, e. g. programs of the operating layer, the structure of interchanged data has to be kept fully compatible because those modules have not been changed.

1. Pascal supports `PACKED` records and arrays to facilitate having minimal alignment space between elements. C does not support this feature.

2. At GSI we use the Organon Pascal compiler from CAD-UL which supports the dialect of the Oregon Pascal/2 compiler. Their syntax only differs in one key word, but they generate completely different codes. However p2c makes some assumptions about the generated code, e. g. the order of bits in a bitset, which is crucial, for instance, when hardware registers are accessed.

3. In Pascal the allocation size of an enumeration type depends on the number of its elements. It may be one or two bytes. In C the allocation size is always an `int`.

4. The Pascal string `ARRAY [1..len] OF CHAR` contains `len` characters. Its allocation size in memory is `len` bytes. A C string with equal size is `char s[len]`. It can hold only `len` - 1 (printable) characters because of the terminating `\0`.

2.2 Linking of Pascal and C modules

A CPU of the device representation layer hosts EC software of up to 12 different device classes. On this layer it must be possible to combine modules written both in Pascal and C because EC software for a number of device classes can not be converted at the same time.

[1] p2c is part of many Linux distributions. It runs under VMS as well.

Combining Pascal and C modules means that they have to be linked together. In this case identical procedure calling mechanisms have to be ensured.

1. P2c translates routine parameters into a structure that contains a pure C function pointer and a "static link", a pointer to the parent procedure's local variables. This structure is passed to the called function. Both of our compilers, the Pascal as well as the C compiler, need plain C function pointers. The option to force p2c to use this concept is available but does not work.

2. Pascal can handle conformant array routine parameters defined as

```
f(a: ARRAY [lo..hi: INTEGER] OF MyType);
```

by syntactically passing the array as actual parameter only:

```
VAR x: ARRAY [7..13] OF MyType;
f(x);
```

On calling the routine, the array, or its address in case of a `VAR` parameter, as well as the lower and upper limit of the array are pushed onto the stack. Thus the array bounds may be checked by the called routine.

P2c generates C code where the routine is declared and called with three parameters explicitly. The order the parameters are pushed onto the stack differs from that of the Pascal compiler.

2.3 Maintainability

The converted software is not a final product. It has to be maintained for changed or extended future requirements. Therefore readable and comprehensible code is indispensable. To achieve this, sufficient work has to be invested into simplifying and refurbishing the plain C code produced by p2c and the Perl scripts.[2]

1. Pascal supports nesting of routines. The parent routine's local variables lie in the scope of the nested routine. C does not provide this concept. So p2c combines the parent routine's local variables to a single structure and adds an additional link parameter to the subroutine's parameter list that points to this structure, thus allowing the subroutine to access its parents' variables. C code designed like this looks somewhat odd.

2. Pascal provides the `WITH` statement to abbreviate the notation for references to fields of structured variables.

```
WITH struc.field DO subfield := 1;
```

P2c creates a pointer for every `WITH` statement with generated names `WITH`, `WITH1`, `WITH2`, etc. to access the field of a structure.

```
T_field *WITH = &struc.field;
WITH->subfield = 1;
```

[2]Due to restricted space only some items are mentioned here.

Often there is no explicit type for the field the `WITH` statement references. In those cases p2c needs to declare an additional pointer type first (`typedef struct T_field ...`) before it can define the pointer itself.

These constructs are hardly found in common C programs.

3. Pascal allows the definition of an array of structures within one statement. A variable definition looks like this:

```
VAR x: ARRAY [1..7] OF
         RECORD i: INTEGER; c: CHAR END;
```

Although C supports a corresponding construct, p2c declares a structured type before it defines the array.

```
typedef struct _REC_x {int i; char c} _REC_x;
_REC_x x[7];
```

To do so p2c must generate a name for the structured type, which is `_REC_x` where `x` is the name of the array.

2.4 P2C Errors

We encountered only two substantial p2c errors not mentioned in the p2c manual. Both of them are very difficult to detect since the compiler does not report an error. Overlooking them during the manual intervention means they occur during the runtime of the software where they are moreover hard to debug.

1. In some cases p2c translates a Pascal 32 bit wide unsigned integer type

```
TYPE uns_long = 0..16#FFFFFFFF;
     myType = uns_long;
```

into a single C character type.

```
typedef char myType;
```

The error occurs only infrequently. Unfortunately we were not able to reproduce the circumstances of its occurrence.

2. The Pascal pointer `ptr` should point to a 16 bit wide type, e. g. a hardware register, that has an offset of 4 bytes to a base address `addr`.

```
TYPE uw_p = ^uns_word;
VAR  ptr: uw_p;
     addr: uns_long;
ptr := loophole(uw_p, addr + 4);
```

In rare cases p2c translates the pointer assignment to

```
ptr = (uns_word*)((uns_long*)addr + 4);
```

which results in a miscalculated pointer value. The expression `(uns_long*)addr` type-casts `addr` to a pointer to a 32 bit type and thus adding $4 \times 4 = 16$ bytes to the base address instead of 4.

3 APPLYING THE STYLE GUIDE

Unlike C, Pascal identifiers are case insensitive. P2c takes the first occurrence of an identifier to determine the notation of all subsequent occurrences. Mostly these notations do not conform to our style guide. To force the notation of identifiers according to the style guide, we developed some Perl scripts that do most of the job.

A Perl script recognizes expressions for instance like

```
#define The_Answer 42
typedef struct my_type {...} my_type;
```

and recasts the identifiers accordingly (getting `THE_ANSWER` and `MyType`).

To handle more complex constructs, a parser-like script would be required. This is not implemented yet. Thus manual modifications are necessary whereby each identifier has to be adjusted only once.

All changes of identifiers in the software of one device class are then stored as key value pairs in a device class specific local data base (DB). The pairs describe the translation from the old into the new style guide conform notation. The creation and completion of the local DB is done by another Perl script.

A third script is used to apply the translations stored in the local DB to all identifiers in all files of a device class. Additionally a global DB is used which applies the translation of the identifiers of the system interface.

4 STATUS

Currently EC software for 15 different device classes has been converted. Devices have been operated with the converted software for more than 6 months. Some of them are even in therapy operation [4]. Apart from teething problems in the beginning of the conversion process the software has showed good quality and bug fixing is an amazingly rare necessity.

4.1 Time

To estimate the manual interaction effort to convert the software for one device class the process can be split into 4 phases. The outcome is the following distribution:

1. p2c including some preparations 10%
2. manual intervention, part I 20%
3. Perl including building of local DB 20%
4. manual intervention, tests, bug fixing 50%

Phase 2 is necessary since some manual interventions are better done before using the Perl scripts. Although 1 and 3 are the “automatic” phases they also need manual actions, particularly phase 3. With more experience the percentage of phase 4 increases but the overall conversion time decreases.

On an average, EC software for one device class consists of 2200 LOC. Its conversion requires us about 2 person-weeks. To convert the whole EC software consisting of 170 000 LOC we will need approximately 39 person-months or 3.25 person-years.

Without the help of p2c and Perl scripts we roughly estimate twice to four times the effort. There was only one attempt to convert a device class completely manually.

Balzert [5] states that software *development* results in 350 LOC per person-month.

Given this, our method is 2 to 4 times faster than a pure manual conversion and more than 10 times faster than a redevelopment.

5 CONCLUSION

Using p2c and Perl scripts converting EC software from Pascal to C is feasible without major problems. In spite of the automation tools there is a lot of manual intervention required before C software for a device class is ready to be released.

Our method allows us to convert EC software in reasonable time. Entirely re-engineering the EC software would have exceeded our manpower capacity excessively.

With EC software converted to C we are well-prepared to take the next step to C++ (or Java). It should be possible, at least on the device representation layer, to re-use the C functions, which are usually straightforward, as methods of classes in C++. The use of C++ on the real-time layer has to be investigated, particularly with regard to the highly demanding 50 Hz linear accelerator operation.

6 ACKNOWLEDGEMENTS

Thanks to Peter Kainberger for all the Perl scripts, and to him, Gudrun Schwarz, and Regine Pfeil for contributions to the cookbook.

7 REFERENCES

[1] U. Krause, V. Schaa, R. Steiner, “The GSI Control System”, Proceedings of ICALEPCS ’91, Tsukuba, Japan, 1991.

[2] U. Krause, “Re-Engineering of the GSI Control System”, these proceedings.

[3] L. Hechler, P. Kainberger, G. Schwarz, “P2C - Umstellung der Gerätesoftware von Pascal nach C/C++”, Accelerator Controls Documentation U-GSW-08, GSI, Darmstadt, November 2000, http://bel.gsi.de/mk/sty/p2c.html.

[4] U. Krause, R. Steiner, “Adaption of a Synchrotron Control System for Heavy Ion Tumor Therapy”, Proceedings of ICALEPCS ’95, Chicago IL, USA, 1995.

[5] Helmut Balzert, “Lehrbuch der Software-Technik: Software-Entwicklung”, Spektrum Akad. Verlag GmbH, Heidelberg, Berlin, Oxford, 1996.

[6] Dave Gillespie, “p2c - Pascal to C Translator Manual Pages”, Caltech.

[7] Udo Krause, “C/C++ Style Guide”, Accelerator Controls Documentation O-SIS-10, GSI, Darmstadt, December 2000, http://bel.gsi.de/mk/sty/cstyle.html.

WEAP035
physics/0111055

DATA ACQUISITION AND USER INTERFACE OF BEAM INSTRUMENTATION SYSTEM AT SRRC

Jenny Chen, C. J. Wang, C. H. Kuo, K. H. Hu, C. S. Chen, K. T. Hsu, SRRC, Hsinchu, Taiwan

Abstract

Data acquisition systems for the accelerator complex at SRRC are composed of various hardware and software components. Beam signals are processed by related processing electronics, and connect to control system by various type of interfaces in data acquisition front-ends. These front-ends include VME crates, personal computers and instrument bus adapters. Fast Ethernet connected all elements together with control consoles. The user interface is running on the control console. Real-time data capture, display and analysis are supported on the control console. Analysis tools based on Matlab scripts are adopted. Hardware and software implementation of the system is presented. User interface supports are also described.

1 INTRODUCTION

The control system of SRRC is essential to operate the light source efficiently. The console level is composed of control console and the control server. Field level computer composed of more than 30 VME crates and several PCs for special devices. Both level computer systems are connected by a dedicated control fast Ethernet. Beam instrumentation system is composed of various beam monitors and associated processing electronics. The most important beam parameters include beam current monitor and closed orbit, beam profile, etc. Various software tools are integrated with the system and provide an efficient means of operation.

2 OUTLINE OF THE CONTROL AND BEAM INSTRUMENTATION SYSTEM

2.1 SRRC Control System

The control system is a two-level hierarchy computer system [1]. Upper layer computers include two process computers and many workstations and PCs. Database management, archive and various application programs are executed on the process computers. The main purpose of the workstations is for use as operation consoles. Bottom layer computers are distributed VME crate controllers that are in charge of the control and data acquisition for accelerator equipment. Both computer layers are connected by a local area network. The software environment can be divided into four logical layers. They are device access layer, network access layer, database layer and applications from bottom to top. The database plays a role as data exchange center for applications and subsystems of accelerator system. Most of the data is updated into database at ten times per second.

2.2 Beam Instrumentation System

The beam instrumentation system is composed of various monitors and supporting electronics. The system supports precision intensity measurements and lifetime calculations. Orbit measurement, synchrotron radiation monitor and destructive monitor. All devices are controlled by VME based local controllers. The control console can access these devices through the user interface. The synchrotron radiation monitor is controlled by a PC and connected to control system via Ethernet. Beam diagnostic instrumentation systems provide the necessary electron beam parameters for commissioning, routine operations and beam physics studies.

2.3 User Interface

The Common Desktop Environment (CDE) is used for user interface development. It is an integrated graphical user interface for the SRRC control system, combining X Window System, OSF/Motif, and CDE technologies. Motif GUI Builder and Code Generator, UIM/X GUI builder is a used to generated various user interfaces. It enables software developers to interactively create, modify, test and generate code for the user interface portion of their applications. To satisfy various requirements, some applications are development in a LabVIEW based environment. For fast prototyping, users can customize user applications in the Malta environment. The control system supported various database access MEX files. User's can access the database directly in Matlab. For the IEEE-488 based interface, a fast Ethernet based GPIB adapter was support. A GPIB/2 interface is also included to allow user access these instruments within Matlab.

3 SPECIFIC APPLICATIONS

3.1 BPM System

The BPM system consists of 57 BPMs equipped with switched electrode processing electronics. The data acquisition is done by a dedicated VME crate equipped with 128 16 bit ADC channels. The VME host sends raw data to the orbit feedback system via reflective memory. Averaged data are updated to the control database 10 times per second with a resolution of around one micron.

3.2 Orbit Display Utilities

To provide better orbit observation, a Motif based orbit display utility was developed. This utility provides basic orbit display and difference orbit display. Several display attributes can be selected by users, such as full scale range, persistent display mode enable/disable, average number, save, print, … etc. The update rate of the display is up to 10 Hz. This is very useful for routine operation and various machine studies.

3.3 Turn-by-Turn BPM System

To support various beam physics studies, several BPMs are equipped with log-ratio processors for turn-by-turn beam position measurement. A multi-channel VME form factor digitizer with 12-bit ADC acquires turn-by-turn beam position. A server program running on a VME crate manages the data acquisition of beam position. A client program running on control console and Motif based GUI are used for the user interface.

3.4 SmartLink System to Acquire Beamline Data

To acquire data from a remote site, a SmartLink based system was setup. The link used a private Ethernet as a field bus. The Ethernet is connected to a PMC Ethernet module installed on the VME host. The SmartLink data acquisition module is used to acquire data from the beamline monitor with high resolution, including photon flux (Io) monitor and photon BPM blades current. The update rate is about 2 times per second with 20-bit resolution. This slow update rate is the major disadvantage.

3.5 Synchrotron Radiation Monitor Interface

The synchrotron radiation monitor is used to measure beam profile. It consists of an optics and a high resolution CCD. To acquire profile information, a PC acquires the image from the CCD camera, analyses the profile and extracts profile parameters. The local display is also broadcast via facility-wide machine status CATV system. A server program running on PC serves the data request form control console. A client program running on the control console can access information of this PC by the help of LabVIEW program or Matlab MEX files running on the control console.

3.6 Gap Voltage Modulation Study Support

RF gap voltage modulation was adopted to relieve the effect of longitudinal coupled-bunch instability in routine operation of the storage ring at SRRC. Systematic measurements were done recently to investigate the mechanism why RF gap voltage modulation can do this. The experimental setup is shown in Figure 1. The gap voltage modulation frequency is about 50 kHz. The rolled off frequency of the LLRF gap voltage regulation loop is about 7 kHz. A function generator in VME form factor generates the modulation sinusoidal wave. This generator integrates with the control system to satisfy the requirement of routine operation. Frequency and amplitude can be adjusted on the control console. The modulation signal is injected after loop filter and added with a correction signal in the RF gap voltage regulation loop. HP4396A spectrum/network analyzer observes the beam spectrum form BPM sum signal.

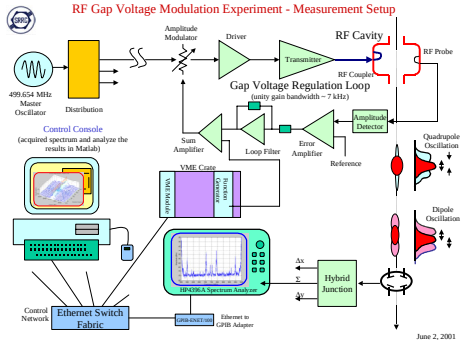


Figure 1. Experimental setup for RF gap voltage modulation study.

National Instruments GPIB/ENET-100 controller connects the spectrum analyzer to the control network. The control console supports database accesses MEX-file that allows Matlab to read and set the modulation parameters. GPIB MEX-file allows Matlab to read and write the GPIB devices via a GPIB/ENET-100 controller. The experimental data can be acquired directly into Matlab. The gap voltage modulation is effective because the modulation frequency is far beyond the unity gain cutoff frequency of the gap voltage regulation loop. Modulation amplitude is about 10 percent of total gap voltage in routine operation. The experiment's sequence is programmed by a simple Matlab script running on the control console that selects frequency scan range as

well as modulation amplitude. The spectrum analyzer can select to measure upper or lower sideband either 1fs or 2 fs synchrotron oscillation frequency. Figure 2 shows the measured result and clear show that the instability is suppressed near 50 kHz.

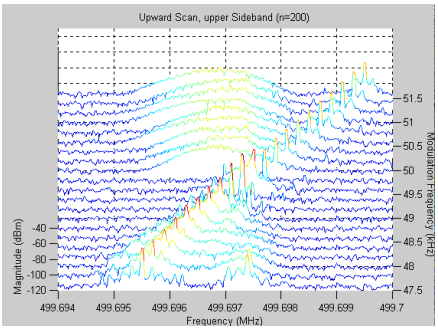


Figure 2. Typical scan spectrum of RF gap voltage modulation study.

3.7 Booster Tune Measurement and Correction System

The electron beam is accelerated form 50 MeV to 1.5 GeV at the booster synchrotron with 50 msec ramping time. Tune is an important index indicated the tracking performance of the White circuit based power supply system. Tune information was obtained by the tune measurement system; it provides the νx and νy during energy ramping. Tune variation in the energy ramping process are correlated to the tracking performance of three families White circuit. The optimized lattice can be obtained by the help of measured tune for the booster synchrotron to get a better working point for efficient operation. An application program was developed to automatically measure and correct the tune variation. The stored beam is excited by extraction kicker to perform damped betatron oscillation. Trigger timing and field strength of the kicker is set properly as the function of beam energy to ensure sufficient beam excitation and without killing the stored beam. Beam motion signals are picked up by stripline and processed by log-ratio BPM electronics [2]. A transient digitizer in the VME crate records the betatron oscillation of the stored beam. Server programs running on the VME crate coordinate the process of data acquisition. A client program running on control consoles is invoked by a Matlab script to perform data acquisition and analysis. Data analysis includes Fourier analysis, peak identification and visualization. The tune correction signal is generated and downloads to waveform a generator located in the VME crate. Figure 3 shows the system structure of the tune measurement and correction system [3]. The Matlab script is used to generate a correction waveform by the measured tune and measured sensitivity matrix between tune and quadrupole setting. Figure 4 shows the tune during ramping with and without correction. Tune variation during ramping can be reduced drastically by this feed-forward correction.

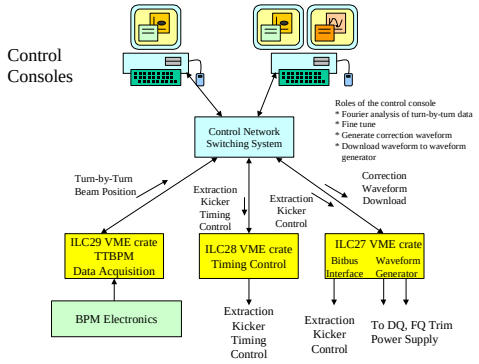
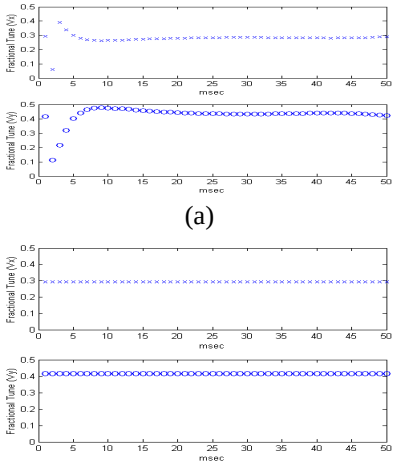


Figure 3. Tune acquisition and correction system.



(a)

(b)

Figure 4. Tune variation during energy ramping; (a) before correction, (b) after correction.

4 SUMMARY

Data acquisition and user interface of SRRC are summarized in this report. These systems are essential for the operation of accelerator system for machine tuning, various feedback, machine study, etc. The system is evolving continually with technology advanced.

REFERENCES

[1] Jenny Chen, et al., "Virtual Instrumentation Interface for SRRC Control System", PAC95, 2256 (1995) and reference therein.

[2] K. H. Hu, et al., "Turn-by-Turn BPM Electronics Based on 500 MHz Log-Ratio Amplifier", Proceeding of PAC99, 2069 (1999).

[3] C. S. Chen, et al., "Study of the White Circuit Tracking Performance in the Booster Synchrotron of SRRC", Proceeding of PAC2001, paper FPAH301.

WEAP037
physics/0111113

PLC BASED UPGRADES FOR THE CAMD LINAC AND STORAGE RING CONTROL SYSTEM

P. Jines and B. Craft, Center for Advanced Microstructures and Devices, Louisiana State University, 6980 Jefferson Highway, Baton Rouge, LA 70806, USA

Abstract

Louisiana State University Center for Advanced Microstructures and Devices (CAMD) began a control system upgrade project in early 1997. At the time, the storage ring was controlled by a VAX/VMS system, primarily using CAMAC for I/O. The injector Linac was controlled by a separate VME/OS/9 system using custom I/O components. The storage ring control system has been replaced with a PC/Linux based system, utilizing the existing CAMAC subsystem [1]. In an effort both to migrate channels away from CAMAC and integrate the Linac controls with the storage ring controls, a design for upgrades utilizing AutomationDirect.com PLCs was presented [2]. This paper will discuss the detailed design for the Linac PLC based control system, and discuss the status of PLC upgrades for the Vacuum system, Kicker controls, 2nd RF system, and utilities monitoring.

1 INTRODUCTION

When the CAMD control system upgrade project began in early 1997, CAMD was faced with the task of integrating two separate control systems, with no reliable communications method between the two. Although CAMD's LINAC and storage ring were both procured from Maxwell Laboratories via a single, fixed price contract, early in the design phase the Linac was subcontracted to CGR-MeV.

At the end of commissioning, the ring's control system was a VAX/VMS based system running Vista Controls VSystem software, utilizing CAMAC, GPIB, RS232, and Allen-Bradley Remote I/O communications. The Linac control system, however, was a VME/OS/9 system using custom I/O cards. While Maxwell and CAMD had specified TCP/IP access to the control system computer, subsystem acceptance tests had shown that this type of communications was not possible. If the system was "pinged" via TCP/IP, the watchdog timers expired, and the software interlocks shut down the Linac.

The first control system to be upgraded was the storage ring control system. Initially, the VAX was replaced with a PC/Linux based system, preserving all existing hardware interfaces. Later phases of development were focused in two areas. The first was enhancing the controls capability, including adding additional control channels for a superconducting wiggler and additional corrector magnets, and increasing logging and operator automation. The second area was reducing the dependence on CAMAC and proprietary Allen-Bradley protocols, and allowing remote data acquisition. In this effort, GPIB and RS232/422/485 channels were given remote capability using National Instruments GPIB-ENET and Comtrol Rocketport Serial Hub Si remote ethernet based hubs. For all other upgrades, CAMD has standardized on AutomationDirect.com DL405 series PLCs.

2 PLC/CONTROL SYSTEM OVERVIEW

2.1 PLC Types

The AutomationDirect.com DL405 series PLCs have two different types of base controllers: the DL405 CPUs and the Ethernet Base Controller (EBC). The DL405 CPUs provide traditional CPU/Ladder Logic type control of the PLC, whereas the EBC eliminates the CPU, and provides a "pass thru" to allow the PC to control the PLC's I/O directly.

In combination with the DL405 CPU, the ethernet communications module allows the PC to read and write the PLC's memory without special communications software in the PLC. This led to the idea of using a "dual-port" memory model, where machine status and other parameters are communicated between the PLC and the control PC thru the PLC's memory. This provides a system with PLC type control, but PC based remote configurability.

2.2 Control System Integration

Channel information for CAMD's control system is stored in a PostgreSQL database, and is organized by device type, with device specific parameters as needed. As the CAMAC channel information is organized by crate, slot, channel address, and channel type, this seemed a logical approach to organize the PLC channel information. For the EBC type controller, this provided an exact match for the access method that the communications library provides. For the CPU based

controllers, additional abstraction is necessary. The DL405 CPU architecture provides a flexible memory map: Inputs, outputs, control registers, and timers can be accessed directly, or as a "V memory" address. The ethernet communications protocol differentiates between different I/O and memory types by simply providing an address and a hex value for I/O or memory type. By combining this flexibility with the fact that the DL405 CPU's can provide a fixed starting I/O address for each slot (thereby eliminating I/O space size differences between a slot containing a 8 channel module and one containing a 64 channel module), a software mapping can be made between the "crate, slot, channel" view and the "V memory" address space. By adding fields to control the "base V memory" address, the PC code can be written without concern as to whether it is writing directly to I/O, or to a memory location, indicating that the PLC CPU will handle the I/O itself.

3 LINAC CONTROL SYSTEM UPGRADES

3.1 Current Linac Configuration and Migration Plan

The current Linac control system is based on a VME computer with custom I/O components. A custom scanner card in the VME crate using a dual-port memory map protocol communicates with the I/O cards via a 50-pin ribbon cable using a CGR-MeV in-house protocol. Each I/O card is homogenous in that all digital I/O is a TTL signal, and all analog I/O utilizes a 0-10V signal. Each card is given its address via external TTL voltages. The cards have a ribbon cable connector on one side, and connect to a 3U Eurocard backplane that handles all incoming and outgoing control signals. This backplane then connects to any signal specific hardware necessary, such as TTL to 24V contact relays, voltage to frequency converters, or directly to the devices controlled.

As all I/O is homogenous at the Eurocard backplane, this is the point at which the conversion from the old control system to the new PLC system will take place. Special cables have been fabricated to connnect D-shell DIN-rail breakout boxes to the 96-pin backplane connectors. Using this setup, the new control system can be tested, and the Linac can be reverted to the old control system simply by removing the adapter cables, and reinserting the I/O cards as before.

3.2 PLC Linac Control System Architecture

The Linac control system design utilizes simple, dedicated purpose PLCs. Analog interlocks and associated digital permits, control, and interlocks are all performed by separate PLCs or relays. This allows portions of the system to be distributed and enhanced without affecting the entire system.

Most digital Linac interlocks will be performed by dry relays. These will receive control inputs from EBC PLCs, and permits from smarter CPU based PLCs, which are responsible for all analog limit checking.

Temperatures, power supply readbacks, and other analog signals are monitored by dedicated "analog limit checking" PLCs. Each PLC is responsible for checking a group of analog inputs against upper and lower limits contained in its memory. The readbacks and limit values are all readable and/or writeable by the operator using the dual-port memory interface. In the event of an out of range indication, a digital output, or "permit", is turned off, which in turn is monitored by interlock PLCs or relays. The faults are latched, and must be reset by the operator before continuing.

Digital interlock PLCs and relays are used to monitor the analog "out of range" faults, and well as other incoming digital signals. These are responsible for removing permits in cases of low water flow, valve closures, power supply out of range conditions, or other indicators.

4 RING CONTROL SYSTEM UPGRADES

In addition to the Linac control system currently in progress, CAMD has also upgraded several ring systems to PLC based systems. These include Vacuum interlocks and monitoring, TSPs, and other digital control functions.

4.1 Vacuum Interlocks and Monitoring

CAMD has recently upgraded its vacuum system. There are significantly more ion pump controllers and vacuum gauge controllers than in the older system. The decision was made to replace the old control system with a combination EBC/CPU PLC system. The system now consists of one identical rack per quadrant, each containing an EBC based PLC. These PLCs monitor the readbacks for each Ion pump controller and vacuum gauge controller in the quadrant. Contact closures from the vacuum gauge controllers are sent to a CPU based PLC, which contains the ladder logic and digital outputs necessary to close various ring valves in cases of high pressure conditions.

4.2 Titanium Sublimation Controllers

The initial TSP system for the CAMD storage ring contained only one sublimation controller, had no indication of the success or failure of the firing operation, and provided no indication as to how much current was applied to each filament. An upgrade was

planned to provide one sublimation controller per quadrant, routing its output via a PLC controlled multiplexer, and to provide a graphical user interface with logging capability for the firing operation. For this activity, unused digital output channels from the vacuum system upgrade's EBC PLCs were used to control the multiplexer. The sublimation controllers' RS232 interfaces were controlled using the Comtrol Rocketport Serial Hub Si ethernet hubs. At the completion of the upgrade, the TSP system now can be run in one fourth of the time previously required, has a graphical indication of its activity, and has logging capability to monitor the vacuum system's performance.

4.3 Other PLC Upgrades

At present, CAMD is attempting to utilize CAMAC only for machine ramping functions, and to migrate all other controls to PLC based systems. In this effort, controls for both a second RF system being commissioned and the existing RF system are being converted from CAMAC to PLC control. Kicker controls have been converted from an Allen-Bradley PLC5 platform to the AutomationDirect.com platform. Monitoring for water plants, critical Linac temperatures and SF6 pressures are being provided by EBC based PLCs. Digital controls for new superconducting power supplies are being developed using the EBC platform. Finally, Ion Clearing Electrode controls have been converted to PLC based control.

5 CONCLUSION

CAMD has converted many existing controls to the AutomationDirect.com PLC systems. A new Linac control system is being developed using a combination of smart CPU based PLCs for interlock control, and less expensive EBC based control where only pure I/O is needed. At the conclusion of the upgrades, CAMD expects to have a consolidated, inexpensive, open and supportable system utilizing CAMAC where ramping is required, and non-proprietary PLCs for most other functions.

REFERENCES

[1] B. Craft and P. Jines, 'The GNU Control System at CAMD,' Proceedings of the 1997 ICALEPCS, pp. 194-196, 1997

[2] P. Jines and B. Craft, 'The GNU Control System at CAMD: Part Two,' Proceedings of the 1999 ICALEPCS, pp. 118-120, 1999

WEAP038
physics/0111155

CONTROL SYSTEM DESIGN FOR THE LIGO PRE-STABILIZED LASER

R. Abbott and P. King
LIGO Laboratory
California Institute of Technology, Pasadena, CA 91125, USA

Abstract

To meet the strain sensitivity requirements [1], [2] of the Laser Interferometer Gravitational Wave Observatory (LIGO), the laser frequency and amplitude noise must initially be reduced by a factor of 1000 in the pre-stabilized portion of the interferometer [3]. A control system was implemented to provide laser noise suppression, data acquisition interfaces, diagnostics, and operator control inputs. This paper describes the VME-based analog and digital controls used in the LIGO Pre-stabilized Laser (PSL).

1 INTRODUCTION

Gravitational waves, the ripples in the fabric of space-time, were predicted by Einstein's General Theory of Relativity. Although astronomical observations have inferred the existence of gravitational waves, they have yet to be detected directly. The Laser Interferometer Gravitational-wave Observatory (LIGO) is one of the large-scale gravitational-wave detectors currently being built worldwide.

The Pre-stabilized Laser (PSL) subsystem is the light source for the LIGO detector as shown in Figure 1. The output of the PSL is modematched into the suspended modecleaner before being coupled into the LIGO interferometer. The term *pre-stabilized* is used because the laser undergoes two stages of stabilization prior to being injected into the interferometer.

The 10-W laser used is configured as a master-oscillator-power-amplifier (MOPA), with a 700 mW single-frequency, single-mode, non-planar ring oscillator used as the master oscillator. The control strategy uses the actuators of the master oscillator in order to stabilize the frequency. Power stabilization is achieved by control of the power amplifier output.

2 HIGH-FREQUENCY INTENSITY NOISE SUPPRESSION

The PSL topology is shown in Figure 2. Light from the laser is modematched into a high-throughput, ring Fabry-Perot cavity called the pre-modecleaner (PMC). The PSL has a design requirement that the output be close to the shot-noise limit for 600 mW of detected light at the interferometer modulation frequency of 25 MHz. As this is beyond the bandwidth of any electronics servo, it is done by passive filtering by the PMC. By appropriate choice of mirror

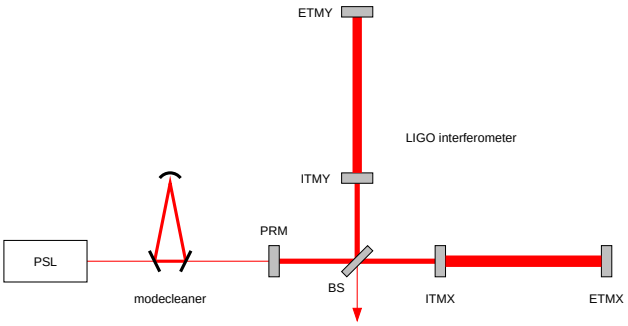


Figure 1: The LIGO interferometer. PRM: power recycling mirror. BS: beamsplitter. ITM: input test mass. ETM: end test mass.

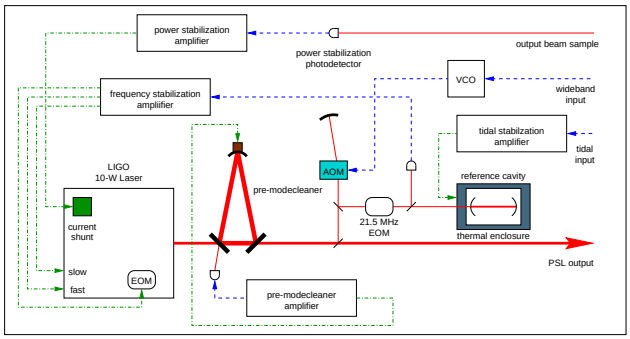


Figure 2: The components of the PSL.

reflectivity, the PMC acts as a tracking bandpass filter with a pole at the cavity half-bandwidth. One of the PMC mirrors is epoxied to a piezoelectric transducer (PZT) to vary the length of the cavity. The servo electronics constantly adjusts the PZT voltage in order to keep the incident light resonant with the cavity.

3 FREQUENCY STABILIZATION

Astrophysical models suggest that in order to plausibly detect candidate gravitational-wave sources, the LIGO detector must achieve a displacement sensitivity of better than $10^{-19}\,\mathrm{m}/\sqrt{\mathrm{Hz}}$ at 100 Hz. This corresponds to a frequency noise of $10^{-7}\,\mathrm{Hz}/\sqrt{\mathrm{Hz}}$ at 100 Hz.

3.1 Actuators

The frequency stabilization servo utilizes three frequency actuators inside the 10-W laser. A thermo-electric cooler (TEC) bonded to the laser gain medium actuates on the laser frequency by thermally changing the optical path length. DC–1 Hz adjustments to the laser frequency are made with the TEC. This actuator, modeled as three poles at 0.1 Hz, has a coefficient of 4 GHz / V and is used for large scale adjustments to the laser frequency. Also bonded to the laser gain medium is a PZT, which covers DC–10 kHz. A voltage applied to the PZT stresses the laser medium and induces refractive index changes to change the laser frequency. The PZT has a flat response to $\sim$ 100 kHz and is known to have a number of mechanical resonances beyond 100 kHz. Fast frequency fluctuations beyond 10 kHz are handled by the third frequency actuator, a Pockels cell located between the master oscillator and power amplifier.

3.2 Implementation

A small fraction of the output of the PMC is sampled and the frequency is shifted through an 80 MHz acousto-optic modulator (AOM). The output of the AOM is focussed into a phase modulator that imparts sidebands at 21.5 MHz. The output of the phase modulator is then modematched into a high-finesse, linear Fabry-Perot cavity which is used as a frequency reference against which the laser frequency is stabilized. The frequency stabilization scheme employs the well-known Pound-Drever-Hall technique in which the light incident on the reference cavity is phase-modulated [4]. Both the carrier and sideband light reflected from the reference cavity is focused onto a tuned photodetector. The output of the tuned photodetector is bandpass filtered and synchronously demodulated to derive the error signal.

In order to ensure closed-loop stability, the open-loop gain of the PZT actuator must be well below that of the Pockels cell at the PZT mechanical resonance frequency. To ensure this, the PZT actuator path is aggressively rolled off after the designed 10 kHz crossover. In the absence of the Pockels cell, the PZT path is naturally unstable at $\sim$ 15 kHz. With a dynamic range some 30 times greater than that of the Pockels cell, a self-sustaining oscillation may arise if saturation occurs in the Pockels cell path. Limiting the dynamic range of the PZT actuator prevents this instability.

4 INTENSITY STABILIZATION

Photons in the laser light induce a source of noise in the interferometer known as radiation pressure noise. This noise arises from the momentum imparted to the mirrors as statistically different numbers of photons reflect off the mirrors in the interferometer. To minimize the movement of the interferometer mirrors due to radiation pressure, the intensity fluctuations of the laser must be stabilized to the level of $\sim 10^{-8}\, 1/\sqrt{\text{Hz}}$.

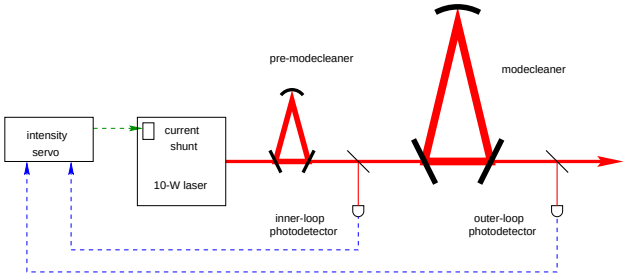


Figure 3: The intensity stabilization system layout.

4.1 Actuator

Currently in the prototype design phase, the intensity servo utilizes a current shunt for fast regulation of the power amplifier pump diode current. Placed in parallel with the power amplifier pump diodes, the current shunt was designed to carry $\sim \pm 250$ mA.

4.2 Implementation

The intensity stabilization servo adopts a dual-loop topology as illustrated in Figure 3. Inputs from photodetectors located after the PMC and modecleaner are used in either a single or dual sensor configuration. In the single sensor configuration, the outer-loop photodetector provides the signal to the servo electronics. In the case where the modecleaner is not locked, the single-sensor signal comes from the inner loop photodetector. In the dual sensor case, both the inner and outer feedback paths provide signals to the servo electronics.

In the dual loop configuration, noise suppression is established in two phases. Closing the inner loop yields a high-bandwidth, well-behaved inner loop with partial noise suppression. The outer loop is then closed around the inner loop to provide the balance of the noise suppression.

5 DATA ACQUISITION AND USER CONTROL

The user control and interface is via the Experimental Physics and Industrial Control System (EPICS). Through EPICS the operator can remotely monitor the performance of the PSL, and adjust the various servo loop gains and settings. The operator interface is a series of graphical screens, that indicate the current status of the PSL. Processing the data and events is the input / output controller (IOC), a Baja4700E MIPS-based processor running the vxWorks kernel. The IOC performs the real-world input/output tasks and local control tasks, and provides status information through the Channel Access network protocol.

The control software for the PSL is event-driven and is written in state notation language. Although not fully debugged, automated operation from cold start through to full operation has been demonstrated. One software routine constantly adjusts the TEC on the 10-W laser to keep the laser frequency well within the dynamic range of the

PZT. One consequence of this is that lock re-acquisition is instantaneous once the cause of the loss of lock is removed.

At present, a dozen signals are acquired and logged through the LIGO data acquisition system. Fast signals are acquired at the rate of 16 kHz, whilst slower signals are acquired at 256 Hz. All signals are recorded and logged.

6 ACKNOWLEDGEMENTS

We thank the entire LIGO team for assistance and support. This work is supported by the National Science Foundation under cooperative agreement PHY–9210038.

7 REFERENCES

[1] Alex Abramovici *et. al.*, "LIGO: The Laser Interferometer Gravitational-Wave Observatory", Science, **256**, 325, April (1992).

[2] A. Lazzarini and R. Weiss, "LIGO Science Requirements Document (SRD)", internal LIGO document E950018-02-E.

[3] P. King, R. Savage and S. Seel, "(Infrared) Pre-stabilized (PSL) Design Requirements", internal LIGO document T970080-09-D.

[4] R. W. P. Drever *et. al.*, "Laser phase and frequency stabilization using an optical resonator", Appl. Phys. **31**, 97, (1983).

WEAP039
physics/0111163

RHIC Power Supply Ramp Diagnostics*

J. T. Morris, T.S. Clifford, B. Frak, J. Laster, A. Marusic, J. van Zeijts
BNL, Upton, NY 11973

Abstract

Reliable and reproducible performance of the more than 800 Relativistic Heavy Ion Collider (RHIC) magnet power supplies is essential to successful RHIC operation. In order to support power supply commissioning, a system was developed to capture detailed power supply measurements from all the RHIC ring power supplies during acceleration ramps. Diagnostic tools were developed to allow experts to assess ramp reproducibility and rapidly identify problems. The system has now become a routine part of RHIC operations, with data captured for every acceleration ramp. This paper describes the RHIC power supply ramp diagnostic system and considers its impact on RHIC operations.

1 INTRODUCTION

More than 800 power supplies are required to operate the superconducting magnets in the Relativistic Heavy Ion Collider (RHIC). Successful operation of RHIC depends on reliable and reproducible performance of these power supplies during acceleration ramps. A system was developed to capture detailed power supply measurements from RHIC ring power supplies during ramps. Diagnostic tools were developed to support expert analysis of this ramp data.

The ramp data sets have proven to be a useful source of information for operators and beam commissioners as well. They provide a quantitative measure of the reproducibility of power supply performance. They are used to analyze power supply response when new modes of ramping are introduced. The ramp data sets are used to identify power supply problems that may go undetected by ordinary power supply alarm mechanisms. Problems in the power supply control system may also be detected by analysis of these data sets.

The ramp diagnostic system has now become a routine part of RHIC operations. Data is captured during every ramp and routinely analyzed by control room personnel. Diagnostic tools have been expanded to simplify data analysis and identify problems quickly. A watching mechanism has also been introduced to bring problems to the immediate attention of operators.

2 DATA ACQUISITION

RHIC power supply references are controlled with Wave Form Generators (WFGs), which deliver new values to all of the RHIC power supplies at a 720Hz rate. Multiplexed Analog to Digital Converters (MADCs) [1] provide measurements of a number of power supply parameters including currents and references. These MADCs are configured to capture data at the same 720Hz rate used by the WFGs.

Once a second, the 720Hz WFG and MADC data is read by RHIC Controls Front End Computers (FECs) [2]. The reading of the data is triggered by the 1Hz event on the RHIC Event Link (REL) [3] so that it is precisely synchronized for the WFGs and MADCs for all power supplies. The data is made available for live monitoring programs and stored in 10 second history buffers that are used for post mortem analysis after power supply quenches. The data is also averaged and added to extended history buffers that are used for ramp diagnostics. In order to accommodate a full acceleration ramp, the extended history buffers are currently configured to hold 260 seconds of 30Hz averaged data.

At the end of a RHIC acceleration ramp, the Sequencer program [4][5] triggers a RHIC Event Link event to start the process of saving ramp data in files. When power supply FECs see this event, they simultaneously halt the accumulation of data in their extended history buffers. A console level server process gathers the contents of all of these frozen 30Hz history buffers from the FECs and stores the data in a set of SDDS files [6]. When acquisition is complete, the server process triggers another event which restarts the accumulation of history data.

The term *snapshot* has been used to describe the capture of similar data for individual power supplies. The complete set of power supply data for a RHIC acceleration ramp is called a *snapramp*. The *snapramp* data set includes all

{*} Work supported by US Department of Energy

of the available signals for each power supply. In order to facilitate rapid analysis of *snapramp* data, a compact version of the *snapramp* data set is also created. Three compact ramp data set files are created by 1Hz sampling of the WFG setpoints, the MADC measured current, and the MADC measured reference from the 30Hz *snapramp* data sets.

3 POSTRAMP ANALYSIS

3.1 Viewing data by supply

Snapramp data can be viewed with the *PMViewer* tool, which is also used to view RHIC power supply post mortem data [7]. *Barshow* is another program that has been specifically designed to do graphic comparisons of power supply waveforms for ramp analysis. The user of the *barshow* program selects individual power supplies for analysis. The *barshow* display includes all available signals from the power supply. In addition, calculated difference signals are added to the display.

3.2 Comparing ramp data

Users of *barshow* or *PMViewer* must decide which power supplies should be examined. The *pscompare* program is used to do a comparison of ramp data sets for all power supplies, rapidly identifying the power supplies that require further examination. The *pscompare* user selects two ramp data sets. WFG setpoints are typically compared with MADC measured currents or references from the same ramp. Corresponding data sets from different ramps may also be compared. For example, measured currents from a ramp may be compared with measured currents from the preceding ramp.

The results of the comparison are presented in a sorted table as shown in Figure 1. Differences are typically listed as a percentage of the full range of operation for that power supply. The power supply with the largest difference is listed at the top of the table. Filtering options allow a user to limit the presentation to a

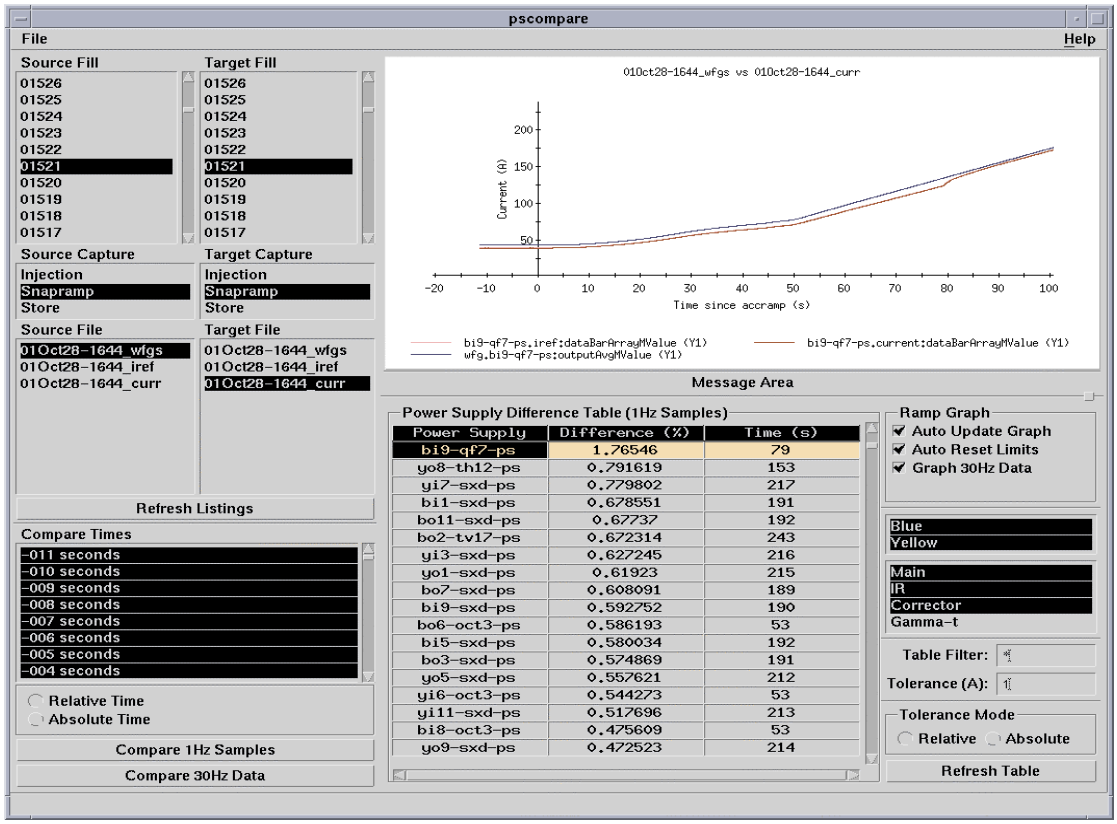


Figure 1: The *pscompare* application with problem power supply highlighted and graphed.

selected set of power supplies (e.g. all sextupoles or all correctors in alcove 9). For performance reasons, *pscompare* comparisons are typically performed on the compact 1Hz *snapramp* data sets. This has proven to be adequate to identify almost all power supply or control problems. An optional comparison of the full 30Hz *snapramp* data is available but it is significantly slower.

As a user clicks on power supply names in the sorted table, data for that power supply is presented in the graphics area above the table as shown in Figure 3. The default graphic view is a quick plot of 1Hz data for the ramp data sets being compared. A user may optionally display the full 30Hz *snapramp* data for the selected supply. This display may optionally include all available signals from the power supply.

4 ALARMS

As described above in the Data Acquisition section, RHIC power supply FECs read and process WFG and MADC data once each second. The data available to the FEC for each power supply includes the current state of the power supply, the WFG setting, an MADC measure of the power supply reference, and an MADC measure of the actual current in the power supply. Software in the FEC compares these values for each power supply that is in an ON state. If these values differ by more than a specified tolerance, an alarm is reported to the RHIC alarm screen in the Main Control Room. During RHIC acceleration ramps, these alarms may be fleeting as some power supplies may only temporarily fall outside tolerance limits. At the end of each ramp, therefore, the RHIC Sequencer displays a record of all alarm conditions that occurred during the ramp.

5 FUTURE

Power supply ramp diagnostics have been an essential tool during RHIC commissioning and they are expected to continue to be important for future RHIC operations. The emphasis to date has been on identification of problems by comparing measured currents with expected values. Under more stable operating conditions, comparison of power supply currents with currents measured during a designated reference ramp will be expected.

The data acquisition mechanism is now mature and has been performing very reliably. The system requires significant storage space with 120MB of data in each *snapramp* data set. The demands of this system and the RHIC post mortem system are forcing a significant expansion of data storage capabilities. Only modest changes are expected in data presentation tools.

The alarm capabilities require the most attention. Alarming has been useful but has not realized its full potential. Tolerances have been loosely set to accommodate calibration offsets in MADC measured values. False alarms still are generated due to intermittent measurement errors and a failure to properly detect state when power supplies trip off. A concerted effort needs to be made to correct calibration errors, eliminate all other sources of false alarms, and then set tolerances to the appropriate levels for each power supply.

References

[1] R. Michnoff, “The RHIC General Purpose Multiplexed Analog To Digital Converter System,” Proc. 1995 IEEE Particle Accelerator Conf., Dallas, p.2229

[2] T.S. Clifford, “The Relativistic Heavy Ion Collider Control System”, ICALEPCS ’97, Beijing, p. 12

[3] B.R. Oerter, ”Accelerator Timing at the Relativistic Heavy Ion Collider,” Proc. ICALEPCS’99, Trieste, p.191

[4] T. D’Ottavio “Description of the RHIC Sequencer”, this conference

[5] J. van Zeijts, “The RHIC Sequencer”, PAC 2001, p. 703

[6] M. Borland, “A Self Describing File Protocol for Simulation Integration and Shared Postprocessors,” Proc. 1995 IEEE Particle Accelerator Conf., Dallas, p.2184

[7] J.S.Laster, “Post Mortem System – Playback of the RHIC Collider”, this conference

WEAP041
physics/0111144

REPLACEMENT OF MAGNET POWER SUPPLIES, CONTROL AND FIELD-BUS FOR THE PSI CYCLOTRON ACCELERATORS

D. Anicic, T. Blumer, G. Dzieglewski, G. Janser, I. Jirousek, H. Lutz, A.C. Mezger
PSI, CH-5232 Villigen PSI

Abstract

Magnet power supplies in the PSI accelerator complex with their control and field-bus are old. Some components are more than 30 years old. To facilitate further maintenance and to meet the more demanding specifications for operation with the 2mA beam, they have to be replaced. The switched power supplies, developed for SLS, will be used. This implies a major redesign of part of the accelerator control system, which is currently based on CAMAC, ROAD-C and other in house developed hardware including the machine protection system. The modified control for the new set-up will be based on VME and alternatively CAMAC, with dedicated processors for the functionality of the machine protection system.

1 INTRODUCTION

The everlasting need for renewal of the control system has hit us again. Power-supplies (PS) are old, the field-bus is outdated, the Front-End processors are no longer available and serial CAMAC is not fast enough. Technology of some of these components goes back to the early 1970's. Stability, ripple and mains rejection of the PS is no longer up to the requirements imposed by the 2 mA beam. The new PS's have no longer an analog interface.

All this implies a new control concept for the PS, including the KOMBI, a local controller that also acts as a supervisor to generate Interlock signals, Interlock being our hardware implemented fast run permit system. ROAD-C, the field bus used to control the PS via the KOMBI is old, in house developed and lacks error detection. All I/O in our system is based on CAMAC. We intend to add the possibility of using VME I/O. For some applications the I/O bandwidth of the 5 MHz bit serial loop is a limiting factor.

On the other hand the overall system is very good and flexible to changes. We plan to use the same control system for the PROSCAN project (the new biomedical facility at PSI) under construction. This is motivation enough for a major upgrade of the system.

2 MODIFICATION PATH OF SYSTEM

We will extend the control system Ethernet to strategic points in the equipment buildings. 100Mbps optical links with the option to go to 1Gbps and beyond will be used in conjunction with Switches that support diagnostic facilities. The addition of front-ends at these locations adds the possibility of VME I/O in the field. Intelligent interface modules to the PS with a dedicated CPU in VME, will replace the old "KOMBIs" as well as part of the Interlock logic.

This enhancement implies the extension and the port of the front-end software to a new processor.

3 THE NEW FRONT END

3.1 FEC software upgrade/replacement

In our distributed client-server based Control System, the Front-End Computers (FEC) act as servers, providing the I/O for data acquisition and control. The FECs continuously wait for client requests (Ethernet 802.3, UDP or Unix Message Queues), perform the requested operations and return status and results. FECs are configured at boot time with the data related to Device, Module-handler, Address etc. provided by our configuration database. Module-handlers correspond to different electronic components (modules). Fig.1 shows the FEC software architecture.

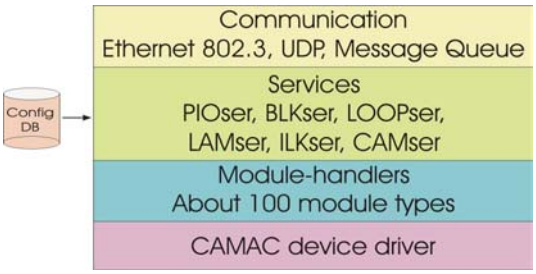


Fig.1: CAMAC based FEC software structure

The present system is entirely CAMAC based with the ROAD-C bus transparently mapped to CAMAC. Currently we support about one hundred different modules. At present, the addition of a new module type requires implementation of a new module handler. In

the future we intend to make better use of inheritance with object-oriented methods.

The motivation for the FEC upgrade and replacement program is twofold. Firstly, the present HP rt743 single board VME real-time computers, under the HP-RT operating system (LynxOS), are almost ten years old and will not be supported for a long time anymore. Secondly, we have new projects based on VME. The upgrade path has already been chosen. We will use new VME computers with the LynxOS operating system. We have ordered Motorola MVME 5100 PowerPC boards and the LynxOS 3.1.0 operating system.

The FEC software migration path is set up in a few steps. Initially, the existing software will be ported to the new platform and operating system, opening the door for present FEC replacement as needed. For that purpose we keep the existing VME to CAMAC Serial Highway Driver. This step has already been done and took three weeks for two men. There were no significant problems besides the compiler influenced warning messages and the effort of implementing the CAMAC device driver. We previously redesigned the original HP-RT driver, while porting our SW to Linux. We are porting it back to LynxOS. This software has now to prove itself in long-term operational tests. In a second phase we will add the VME device driver and gradually implement VME Module handlers, as they become needed. This will then provide a mixed CAMAC and VME system.

3.2 The new FEC database extension

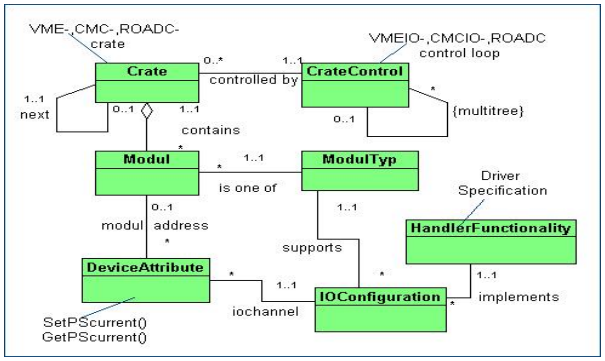


Fig.2: Simplified scheme of I/O-Module configuration

All configuration data used by the Front-End Software is stored in an Oracle Database. The present version only supports CAMAC or ROAD-C based I/O. I/O-Modules are organized by their address: identifier for Front-End, CAMAC Branch, Crate Address in the loop, slot number, and a channel or substation number. Channel numbers are used to identify ADC channels. Substation numbers are used to identify ROAD-C module address lines.

The association or mapping between the I/O and the real device attribute description (e.g. read PS current, set PS current) is normally configured by the Module-handler functionality.

To add the support of VME I/O-Modules both the I/O-Module and the Module-handler mapping part will be extended in the database. New tables, views and their relationships together with the appropriate procedures have to be created, in order to support the new requirements.

For data entry, new application programs are developed using the Oracle Forms Developer tool. The data extraction will use programs written in PRO-C or Java Stored procedures. These programs then supply the Front-End computers with the necessary data to access all connected I/O-Modules. Configuration data contains module identification lists and Device Attribute lists. At Front-End boot time the software first checks for the existence of each I/O-Module and then creates the corresponding device attribute model objects.

4 THE NEW PS INTERFACE

The new PS interface is under development in cooperation with industry. HYTEC Electronics will provide a VME IP carrier board (model 8003) with 4 IP-Sites, where they incorporate a Digital Signal Processor (DSP) to allow for local processing.

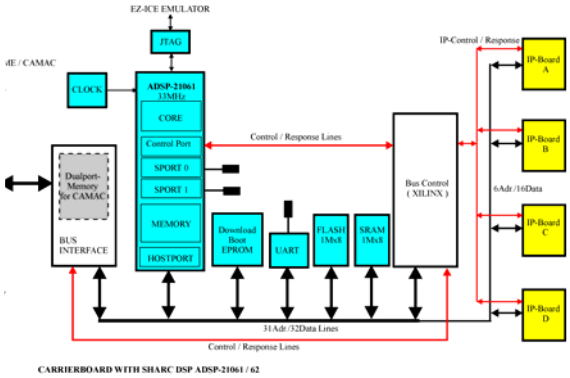


Fig.3: VME-IP Carrier Board

A CAMAC version will also be produced with a dual-port memory acting as interface to CAMAC. The DSP will boot from an internal flash EPROM that can be loaded through VME/CAMAC or by UART (Fig.3). The same Carrier board will also be used for the Interlock logic. Fig.4 shows a structural diagram of PS boards in connection with the Interlock. The DSP provides the data presentation and communication to the Front-end process via memory mapped Registers in

its SRAM memory. The on-board processor controls up to six PS, connected to the carrier board, three times two links per IP module are supported. One IP module is used to implement the Interlock functions. The physical connection to the PS is via transition boards and uses fiber-optic cables. The DSP provides the PS supervision to generate Interlocks for current limits, impedance out of tolerance, etc. Up to six PS may be combined to form virtual devices.

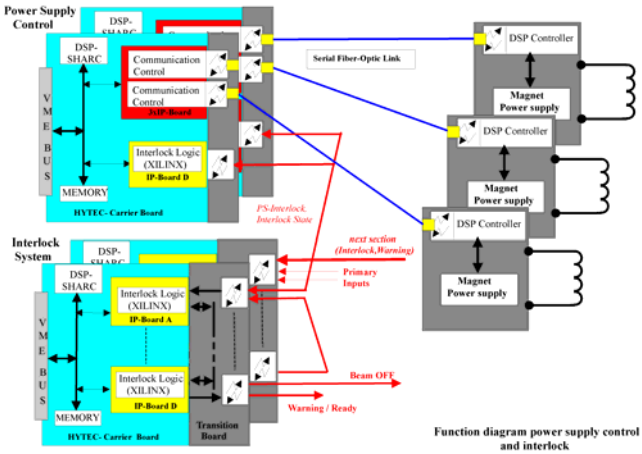


Fig.4: Function Diagram for PS Control and Interlock

Fig.5 shows an overview of the fiber link controller for the point-to-point communication with the PS. On the IP module a XILINX FPGA is used to implement the required logic functions. The opto-electronics part is realized on the VME-Transition board. The FPGA firmware is divided into three logical parts, for two fiber optic links.

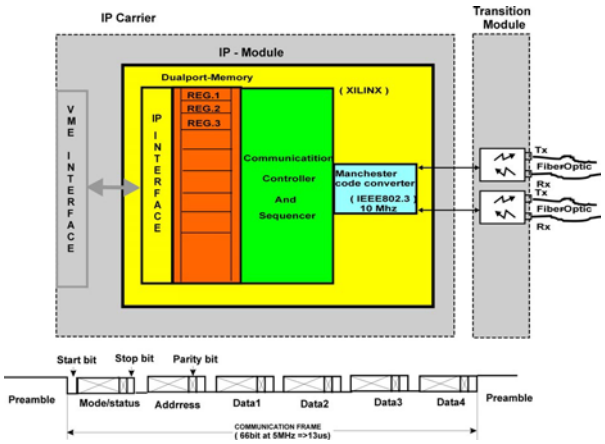


Fig.5: Fiber link hardware and protocol

The protocol of the communication link is always organized in identical blocks (Fig.5). Each block starts with a preamble, followed by a six-byte frame (a byte consist of 1 start bit, 8 data bits, 1 parity- and 1 stop bit). The preamble is a repeated 1 signal for the duration of 12 bits or longer.

The dual port memory used for communication with the FEC is realized in the FPGA. It consists of 64 bytes write-only and of 64 bytes read-only memory for each link. Both the DSP and the VME-CPU may access the memory-mapped registers of the IP-board. A write access will causes the module to send the appropriate data as a command over the optical fiber link to the power supply controller. The answer comes back via the fiber link is stored in the corresponding read register. In addition an interrupt or flag informs the CPU that new data has arrived.

5 INTERLOCK

The new Interlock module is composed of the VME carrier board, four IP modules and the transition board shown in Fig.6. One IP module can serve up to ten Interlock channels, four IN, two OUT and six configurable as IN or OUT. All channels are isolated from the XILINX by opto-couplers. The transition board supplies the Interlock channels via DC/DC converters, insuring over-current and over-voltage protection. The new Interlock will also allow resolving pre- and post-trigger of Interlock event conditions, this is required for a refined diagnostic of the causes of an Interlock. In order to allow for the pre- and post-trigger analysis, input channels are equipped with time counters. The same IP Interlock module is also used in the fourth location of the carrier board for PS control.

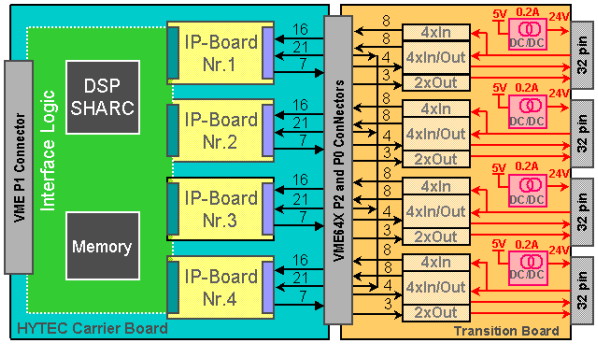


Fig.6: Interlock IP module and transition board

CONCLUSIONS

We presented an upgrade path to bring our control system up to state of the art again. Both control systems for PROSCAN and for our accelerator will profit from this development.

SUBSYSTEM FOR FAST DIGITAL FEEDBACK AT NSLS

S.Ramamoorthy, B.Kushner, B.Podobedov, Y.N.Tang, E.Zitvogel, BNL, N.Y. 11973, USA

Abstract

A fast digital feedback system has been installed in the UV ring at the National Synchrotron Light Source facility and is in operation. A similar system is being developed for the X-ray ring. The micro subsystem is VME-based and uses fast ADC boards to sample the orbit. Corrections are computed and the resulting kick values are sent to the correctors (trim magnets) at 5 kHz through DACs. The subsystem uses the standard real-time Control Monitor [1] developed at NSLS and can accept commands from the high-level workstations. However, during the feedback operations, there is minimal intervention from the rest of the control system. The system provides a bandwidth of ~200Hz (@DC gain 100) and reduces the vertical drift to a small fraction of the beam size. This paper focuses on the architecture of the system and software strategies employed to achieve the 5 kHz rate. For diagnostic purposes, the software stores the orbit and the kick values in the CPU memory in a ring buffer. This data can be retrieved, analyzed and displayed in time and frequency domains[1].

1 INTRODUCTION

The NSLS facility has two storage rings, one for UV and one for X-ray. To improve the orbit stability, analog local feedback systems in the X-ray ring insertion device beam lines and analog global feedback systems for both rings were installed in the early 90's and have been in operation since. These systems provide an order of magnitude reduction in orbit motion. Though successful, the analog hardware cannot easily be modified to experiment with various correction algorithms or different selection of BPMs (Beam Position Monitors) and trim magnets in the rings. Hence a prototype digital system was set up for the X-ray ring and its performance was studied.

The system used two of the existing micros (orbit and trim micros) in the control system. The orbit micro samples the BPM signals at 550 Hz and stores the data in an off-board memory. The trim micro is used by the operators to control and monitor the trim magnet power supplies. A separate micro was installed for computer-intensive feedback calculations. This micro shared the memory with the orbit and trim micros via the bit-3 memory adapter boards [2]. The results of the prototype system were encouraging. However, the bandwidth was small compared to the analog system due to the slow sampling rate.

To match the bandwidth of the analog system, the sampling rate should be at least 30 to 50 times the desired bandwidth. Use of fast digitizers at the BPM end and a second processor in the trim micro could improve the system. This would entail major changes in the electrical cabling and extensive software upgrades to the existing micros. Since the rings are operational most of the time with minimum downtime for maintenance, this is not a viable option. The decision was made to install dedicated micros with minimum modifications in the electrical configuration.

2 HARDWARE CONFIGURATION

Three systems have been implemented, one for the UV ring and two for the X-ray ring (horizontal and vertical planes). These are VME-based microprocessor systems consisting of three standard boards (a CPU, a General Purpose Light Source board and a non-volatile memory) to run the NSLS Control Monitor software and additional I/O boards specific for the digital feedback. The CPU board is a Motorola 2300 series single board computer with 32 Mbyte of DRAM and a 604 PowerPC running at 330 MHz.

For digitization of the orbit signals, VMICVME-3123 ADC boards from VMIC Corporation are used. Each board has 16 differential input channels, each with its own sample-and-hold amplifier and 16-bit A/D converter. It is capable of digitizing all 16 channels simultaneously up to 100 kHz. The board has a dual port memory (DRAM) capable of holding 1 sample to 1 Mega samples of digitized data. The DSP on the board applies gain and offset corrections to the digitized data using the values stored in DSP memory before depositing the data in the DRAM. The feedback corrections are output to VMICVME-4116 DAC boards. Each board has 8 channel D/A converters with 16-bit resolution. The settling time is 10 μsec.

The UV ring has 24 BPMs and 16 trims for each plane and the feedback micro has 3 ADC and 4 DAC boards. The X-ray ring has 48 BPMs and one more

[1] Work performed under the U.S. DOE Contract no: DE-AC02-98CH10886

signal from a photon beam position detector. The number of trims for vertical is 40 and for horizontal 56. Hence the micro has 4 ADC and 7 DAC boards for the horizontal plane and 4 ADC and 5 DAC boards for the vertical plane.

Prior to the implementation of the digital feedback system, buffer amplifiers were used to split the BPM signals among the orbit micro, analog global feedback and local feedback systems. Each amplifier can handle 4 BPM signals. To route the orbit signals to the digital feedback micro without extensive cabling work and without impacting the operational micros, a second buffer amplifier is interposed between the BPM receivers and the existing buffer amplifier. The correction signals from the analog and digital systems and the DAC outputs from the trim micro are output to a summing junction. (see Fig. 1). During the development period, one can switch easily from the analog to the digital system or vice versa by disconnecting the cables.

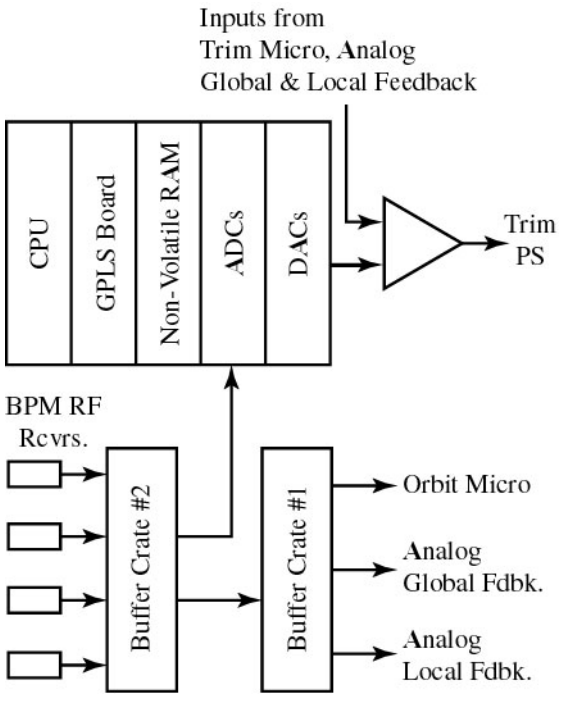


Figure 1: Hardware configuration.

3 SOFTWARE

3.1 Overview

The micro is driven by the same real-time software executed by other micros in the facility. The software uses the real-time VxWorks operating system and provides a standard interface to the high-level workstation programs. The application modules control the hardware and software functions that are specific to a micro. These modules are plugged into the Control monitor software and loaded into the micro. The digital feedback software has logical devices for feedback control and status monitoring. Array devices are also set up for storing data blocks used for the digital feedback correction. Prior to turning the feedback on, high-level software loads a configuration block (specifies the list of disabled BPMS, number of eigenvector and correctors, feedback filter coefficients, feedback gain etc), eigen values, eigenvectors, and reference orbit into the micro. The feedback cycle consists of data acquisition from the BPMs, calculation of the corrections for the trims and data output to the DACs. It is implemented at the highest VME bus interrupt level, generated by the ADC board (see next section). When the feedback is on, there is minimal intervention from the rest of the control system. The frequency of the network messages for status and heartbeat monitoring is usually small and does not have any significant effect on the feedback cycle time.

3.2 Data Acquisition Strategy

At start up, the CPU initiates the calibration mode, during which the gain and offset corrections are generated for each channel and stored in the DSP memory. At the end of calibration, the other parameters are configured. Sampling clock source is internal. To synchronize the sampling of all the ADC boards, they are configured to operate in the multi-board mode and the sample clock signals at the front panel TTL level input are bussed across all the boards. The first board is programmed as a master and the others as slaves. The master mode configures the hardware to drive the same clock signal out of the connector for use by slaves. The slave mode configures the hardware on the board to receive the clock from the master via the connector.

The board provides two modes for the digitization of the signals: 1.Transient Capture mode and 2. Continuous Sampling mode. The first mode cannot be used because of the inherent delays (~1.27 milliseconds) at the start and end of the capture. In the second mode, the data is stored sequentially starting from the beginning of the DRAM buffer. The buffer size is pre-programmed by the host. Once the buffer fills up, it rolls over and starts filling up again. The host has to read the data before it is overwritten. A Halt command can be used to stop the sampling but this will place the board in idle mode causing time delays.

The following software scheme is employed to achieve the 5 kHz rate. The board is sampled at 10 kHz which is twice the desired rate and the buffer size is set to 64 bytes to store 2 sets of samples. When the ADC interrupts are enabled, the board generates a VME bus interrupt when the buffer is **half full** or **full.** The

interrupt routine queries the status to determine the cause of the interrupt. When the **half full** interrupt is asserted, the ADC data is moved from the DRAM to CPU memory using 32-bit transfer. It takes 1 μs to transfer 1 sample. The CPU has a window of 100 μs. before the first set is overwritten. After data transfer, the feedback algorithm is executed in the same interrupt routine. The buffer **full** interrupt waits until the CPU relinquishes the earlier one. The CPU discards the data in the second half of the buffer. To ensure that the digitization is complete for all the boards, the last board in the chain is set up for the interrupt.

3.3Feedback Software

The singular value decomposition (SVD) algorithm is used for the orbit correction. The difference between the measured and the reference orbits is filtered and then decomposed in the orbit eigen space to obtain the projections of the difference orbit on a pre-specified number of eigenvectors. In the corrector eigen space, the kick values for the correctors are constructed along the corresponding correctors' eigen vectors using these projections. Too many eigenvectors can lead to an unstable system, while too few eigenvectors would compromise the correction. For the UV ring, 24 BPMS, 8 trims and 8 eigenvectors are used. The time taken for the feedback cycle is 138 μs. For the X-ray ring, we are trying various configurations. While running with all the BPMs and the trims enabled, the system allows us up to 12 eigenvectors before we run into the CPU constraints. The trim corrections are output to the respective DAC channels. The data is latched simultaneously at the output of the DACs by a software strobe.

3.4 Diagnostic Utility

To provide diagnostics, the orbit data and the trim corrections are stored in a double-buffered ring buffer. The buffer is set up to hold 1 seconds worth of data at 5 kHz. The duration can be easily increased up to 3 seconds with the available memory. The buffers can be switched from one to the other by a command from a workstation or by an external hardware interrupt generated as a result of beam dump. When a client from a workstation requests the data, the Server in the micro will pass the data from the buffer that is not currently active, thus preserving the integrity of the data. The LAN activity does not produce any noticeable increase in the feedback cycle time while retrieving the data. Analysis of the data in frequency and time domain can help one to diagnose the problems causing the orbit instability. Fig.2 is a Fourier transform of the orbit data (top) and the trim data (bottom) plotted as a function of frequency. The top plot indicates some orbit disturbance at 22 Hz. In the trim plot, one can easily locate this disturbance. It is evident that trims around #10 are driven the hardest in correcting the orbit. In this case the orbit disturbance was due to the Elliptically Polarized Wiggler, which is located between the trims 10 and 11 and is typically running at 22 Hz. At the time of studies the EPW orbit compensation system went out of regulation.

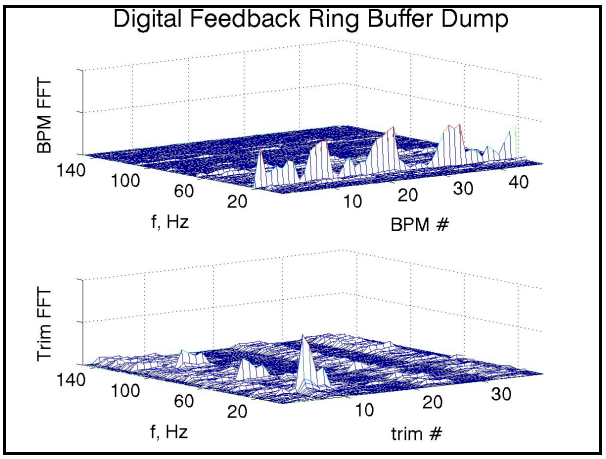


Figure 2: FFT Plot of Orbit and Trim Data

4 RESULTS

The digital feedback micro has been in operation for almost a year in the UV ring. At DC gain of 100, the low frequency noise is practically eliminated. At 60 Hz power line frequency and its harmonic 120 Hz, the reduction in noise factor is 3.5 and 1.5. [3]. The bandwidth is better than the analog system. A significant reduction of the slow orbit drift over a 5 hour-fill is also observed. For the X-ray ring, studies indicate better corrections of the orbit noise as compared to the analog system. The digital feedback for the X-ray ring will be operational soon.

5 ACKNOWLEDGEMENTS

The authors wish to thank R. Biscardi, E. Blum, S. Krinsky, R.Michta , J.D.Smith and O. Singh of APS for productive discussions.

REFERENCES

[1] S.Ramamoorthy et al, "NSLS Control Monitor Upgrade", Proc IEEE, PAC 1993, 1849.
[2] E.Bozoki et al., "First Results of Digital Feedback System at NSLS", Proc IEEE, PAC 1995, 2696.
[3] B.Podobedov et al, "Fast Digital Orbit Feedback system at NSLS", PAC June 2001.

WEAP047
physics/0111061

ORBIT FEEDBACK SYSTEM FOR THE STORAGE RING OF SRRC

C. H. Kuo, Jenny Chen, C. J. Wang, K. H. Hu, C. S. Chen, K. T. Hsu, SRRC, Hsinchu, Taiwan

Abstract

The orbit feedback system plays crucial roles for the operation of the 3rd generation light source. There are various issues in the orbit feedback system that should be addressed to achieve ultimate performance. The orbit feedback system in SRRC was upgraded recently to satisfy the requirement of demanding users. Based upon operational experiences of the last few years, the new system was designed with more robustness and flexibility. Performance analysis tools are also developed to monitor system performance. Algorithms for feedback control, data acquisition and analysis are described and measurement is also presented.

1 INTRODUCTION

The orbit feedback system is used to eliminate orbit excursion due to various perturbation sources. Work to improve orbit stability started to form in 1995 with the orbit feedback system. This orbit feedback system has been incorporated with the insertion devices operation, include undulator (U5 and U9) and elliptical polarized undulator (EPU5.6). Suppressed orbit drift and low frequency oscillation was also achieved. The orbit feedback system for the storage ring of SRRC is being upgraded to improve its performance. The effort includes increased feedback bandwidth, increasing sampling rate, compensating eddy current effect of vacuum chamber with filter, and enhance performance and robustness of the control rules. In this report, we will summarize the status of the orbit feedback development in SRRC.

2 EXISTING ORBIT FEEDBACK SYSTEM

A digital orbit feedback system [1,2] had been developed to suppress orbit disturbances caused by long-term drift, low-frequency oscillation and perturbation from insertion device operation. First, a linear response matrix is measured by taking electron beam position monitor (BPM) readings when the corrector is individually perturbed. Then, this response matrix is used to design a local orbit bump. The feedback controller is based on a PID algorithm. Digital filtering techniques were used to remove noise from the electron beam position reading, to compensate the eddy current effect of the vacuum chamber, and to increase the bandwidth of the orbit feedback loop. The infrastructure of the digital orbit feedback system is composed of the orbit acquisition system, gigabit fiber links, digital signal processing hardware and software, and high precision digital-to-analog converters. From a controls point of view, the orbit feedback is a typical multiple input multiple output problem. The basic concept of the orbit feedback system is shown in the Figure 1. Technically, it is difficult to implement an analog matrix operation consisting of large amounts of BPMs and correctors. Consequently, a digital based feedback system is a natural way to implement orbit feedback system.

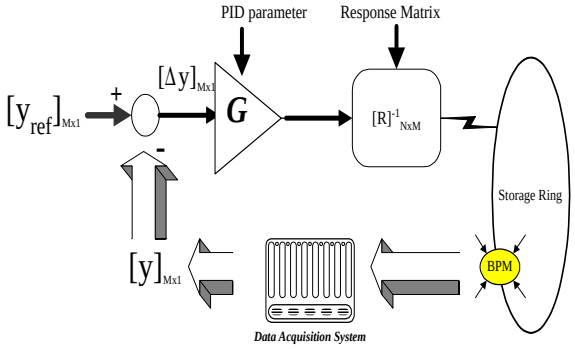


Figure 1. Basic concept of the orbit feedback system

There are two orbit feedback system in this control loop. One is local feedback, and another is global feedback. Integrating these two feedback loops yield better coordination. A system bandwidth of 10 - 100 Hz is necessary to suppress vibration and power supply ripple related beam motion, etc. The feedback system is integrated with the existing control system. BPM data and corrector readbacks are updated into the control system dynamic database in the period of 100 msec. The digital feedback system is bounded on I/O as well as computation. It is important to arrange the real time task and to arbitrate the computer bus properly in order to optimize system performance.

2.1 Hardware structure

The hardware configuration of the corrector control system in SRRC is shown in Figure 2. The low layer is a VME crate system including a PowerPC 604e CPU board and I/O interface cards. The front-end devices are connected to this system via analog and digital I/Os. A PowerPC based server system is used as the TFTP file server for OS downloads and mounted disk for the network file server (NFS). All application programs are put on the server disk. These programs are developed and debugged on client nodes to relief loading of server. The real-time multi-tasking kernel on the VME bus single board computer provided satisfactory performance, reliability, and a rich set of system services. A new device is easy to create by modifying the device table file

as if editing on line. The system can automatically boot and execute different applications in every VME node with the same operation system environments. The upload process handles device (analog input) and sends acquisition data to the database when it receives the broadcast upload message from the Ethernet every 0.1 second. The sampling rate of the feedback loop is 1 kHz by VME interrupt.

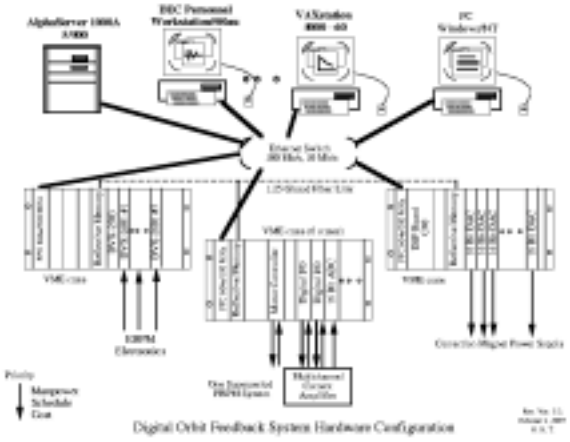


Figure 2. Hardware architecture of the orbit feedback system.

The present system consists of two VME crates, orbit server VME crate, corrector and DSP VME crate. The orbit server acquires BPM data every 1 msec and transfers the BPM data to the corrector node by the aid of reflective memory. The host processor also provides average BPM readings and updates to the control database every 100 msec, the average number is programmable. A JTAG emulator is used for the development of the DSP program. The bus adapter is fit in slot 1 of the VME crate as system controller. Feedback software was developed and debugged on a PC and downloaded to DSP board via a front panel JTAG port. The DSP board, carrying two TMS320C40 TIM modules, handles all signal processing, including a digital low pass filter and PID controller. It takes 1 ms to complete feedback processes including data input, operation of PID, digital low pass filtering, matrix operation, BPMs data reading and corrector settings. The corrector setting has been applied to multi-channel 16 bit DACs with sub-μrad steering resolution. It is allowed remote adjustment when the corrector setting from the graphical users interface feedback loop is turn on.

Intrinsically, the performance of the feedback system is limited by BPM and PBPM resolution. The PBPM data is directly acquired from a multi-channel electrometer by the A/D channel of the VME crate, which are distributed to the beam line. These crates are used as PBPM server nodes. It contains a PowerPC 604, reflective memory and a 16 bit A/D card in each crate. The upper plate and low plate signals of PBPM are sent to the PowerPC with the VME bus. The vertical signals are sent to RM with the PMC bus after transformation processing of two plates. This loop is synchronized with a 1 ms PMC interrupt that is requested by the server crate with RM and fiber link. There is 133 Mbytes per second data communication in the PMC bus, so it is provided for PBPM data transfer quickly and largely. All data is collected at the BPM server node. The orbit server provides fast beam position information to be used for the feedback loop. It also provides slow orbit information for the centralized database. The fast orbit information is sent to the corrector via gigabit fiber linked reflective memory and computation needs in the VME crates.

2.2 Software structure of system

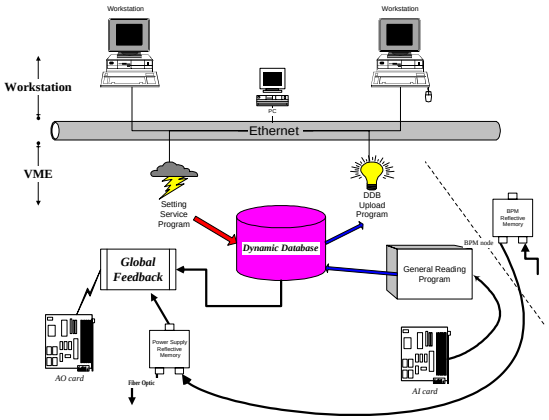


Figure 3: Software structure of the orbit feedback system.

The corrector node is supplied to corrector control with a PowerPC, 16 bit D/A card and DSP card. The structure of the application software is described in Figure 3. There are some service tasks in the PowerPC. The setting task handles corrector settings when the setting command arrives from the database. It spawns child tasks to process all setting requests corresponding to each incoming UPD facility setting packet. The reading process is triggered by the external 10 Hz clock from the network when it receives the broadcast upload message from Ethernet. And sends an event to wake up the data acquisition process. The data acquisition process is directly controller by remote login. The increased I/O card is easily updated by modifying the configure table file. All acquired data is broadcast to Ethernet every 100 ms. The DDB process is the server of shared memory that coordinates the communication between the reading process and setting process. It also provides the data access from another process.

3 AN EXAMPLE APPLICATION OF THE ORBIT FEEDBACK SYSTEM

Changing gap and phase parameters of insertion devices is essential in the operation scenario of a modern light source. Residue fields from the insertion devices are the major perturbation source leading to orbit excursion. It is hard to eliminate with lookup table compensation

schemes. In routine operation of the storage ring, look-up table scheme is used to reduce orbit excursion and orbit feedback system is used to keep the orbit change within micron level. During the gap change of U5 (4-meters undulator with 5 cm period), the orbit changes due to field error. The orbit changed without and with global orbit feedback while adjusting the U5 gap as indicated in Figure 4. The difference orbit is defined to be the orbit changed at 100 μm and 40mm from 219mm of U5 gap. The displacement of the orbit was much smaller when the digital global feedback was turned on in comparison with the case when it was off.

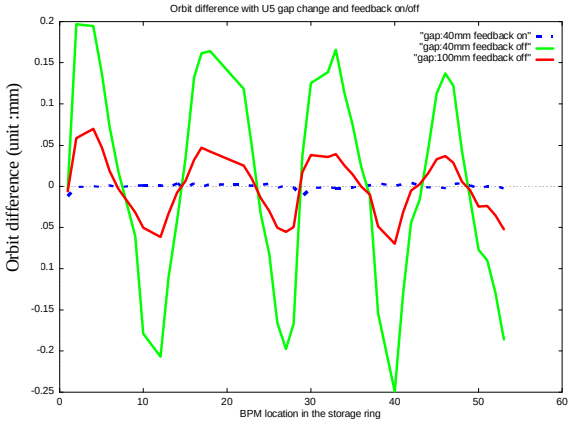


Figure 4: Orbit difference with U5 gap change and feedback on/off.

4 ORBIT FEEDBACK SYSTEM UPGRADE

Upgrading of the orbit feedback system is underway. It is planned that the new system will put into operation in early 2002. There are several reasons for the upgrade: Firstly, to improve system maintainability and stability. The DSP board of the original feedback system is embedded in the corrector control VME crate. It is inconvenient for the development of the feedback system, interference between machine operation and feedback loop R&D is a troublesome problem. Secondly, the system was implemented in 1995 with a slow DSP board; the functionality of the feedback loop is limited. There is not enough computing power to handle the increase demand of the number of BPMs and correctors. The selection of control algorithms is also limited. In the new implementation, the DSP board is located in a separate VME crate. An Ethernet based DSP development system was selected to provide remote access capability. The corrector node is loosely coupled to the DSP VME crate. There are three VME crates in the upgraded system: BPM node, corrector node and DSP node. These three nodes are connected by reflective memory. Several of the fiber link reflective memory cards are tied together by one dedicated reflective hub that simplifies the wiring of the fiber link. The corrector node handles power-supply control. The DSP node handles graphic interface connection of feedback control, calculation of control algorithm and signal conversion from orbit information to correction of corrector. The correction value results are sent to the corrector node and notice the host processor of corrector node by interrupt. The functional block diagram is shown in Figure 5. The planed photon BPM information is sent to the DSP node by a private fast Ethernet network. Local data acquisition for PBPMs are planed to use a compact PCI crate system. In the meantime, this data is sent to the database of console level with the control network.

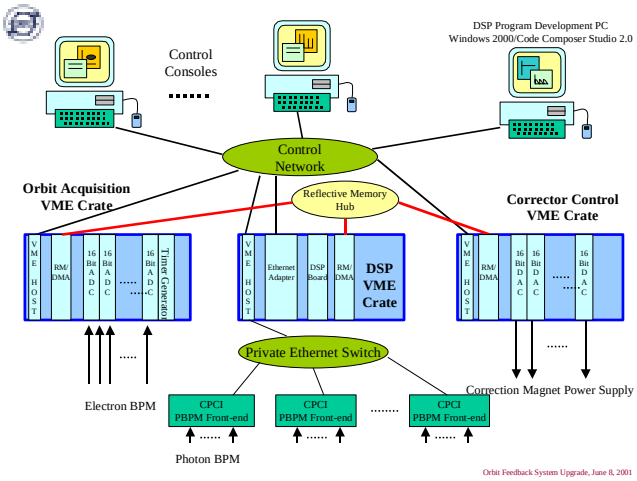


Figure 5: Functional block diagram of the upgrading orbit feedback system.

5 CONCLUSION

An orbit feedback system was developed at SRRC in 1995. To satisfy the requirement of demanding users, the system is being upgraded now. The new system will provide sufficient computing power to execute various control rules and improve the system maintainability at the same time. The performance of this system will be improved as the hardware and software is upgraded and eliminated in any shortage of the existing system.

REFERENCES

[1] C. H. Kuo, et al., "Local Feedback Experiment in the Taiwan Light Source", Proceedings of 1997 IEEE Particle Accelerator Conference, Vancouver, 1997.

[2] C. H. Kuo, et al., "Digital Global Orbit Feedback System Developing in SRRC", Proceedings of 1997 IEEE Particle Accelerator Conference, Vancouver, 1997.

WEAP048
physics/0111209

FAST ELECTRONICS FOR THE DAΦNE TRANSVERSE FEEDBACK SYSTEMS

A. Drago, A. Argan, M. Serio,
Laboratori Nazionali di Frascati - INFN, Frascati, Italy

Abstract

Transverse feedback systems for controlling the vertical coupled-bunch instabilities in the positron and electron main rings are installed at DAΦNE. They started to be operative respectively from June and September 2000. For the horizontal plane, similar systems have been installed in summer 2001 with less kicker power. Design specifications and the basic system concepts are presented. Real time bunch-by-bunch offset correction is implemented using digital signal processors and dual-port RAM's. Fast analog to digital sampling is performed at the maximum bunch frequency (368 MHz). The system manages at full speed a continuous flow of 8-bits data and it has the capability to invert the sign or put to zero the output for any combination of bunches. A conversion from digital to analog produces the output correcting signal.

1 INTRODUCTION

DAΦNE is a ϕ–factory, mainly dedicated to the study of CP violation, currently in operation at Frascati (Italy). It has achieved a peak luminosity of ~5*10^{31} $cm^{-2}s^{-1}$ in the Kloe detector interaction region. Moreover, maximum currents of 1411mA in the electron ring and 1150mA in the positron one have been stored. In Table 1, some DAΦNE parameters are shown.

Table 1: DAΦNE parameters

Ring	e- / e+
Energy	0.510 GeV
Circumference	97 m
RF frequency	368.29 MHz
Harmonic #	120
Rev. frequency	3.069 MHz
Betatron tune x	5.1170 / 5.1615
Betatron tune y	5.1623 / 5.2244
Horizontal freq.	360 / 495 kHz
Vertical freq.	498 / 689 kHz
Max bunch curr	•44.1 mA
Bunch distance	2.7 nsec
Typ. fill pattern	47 stepped by 2

At the beginning of 2000, during commissioning, it was decided to install vertical feedback systems on the two main rings. Similar devices have been successfully used in other colliders or synchrotron light sources [1 - 7], however the design of our system is peculiar under some aspects. The design of the system developed at the Frascati laboratory was stepped in two phases: the first one was made operative in June 2000 and at the begin of 2001 the power of the back end stage was increased [8]. The e+ system is shown in figure 1.

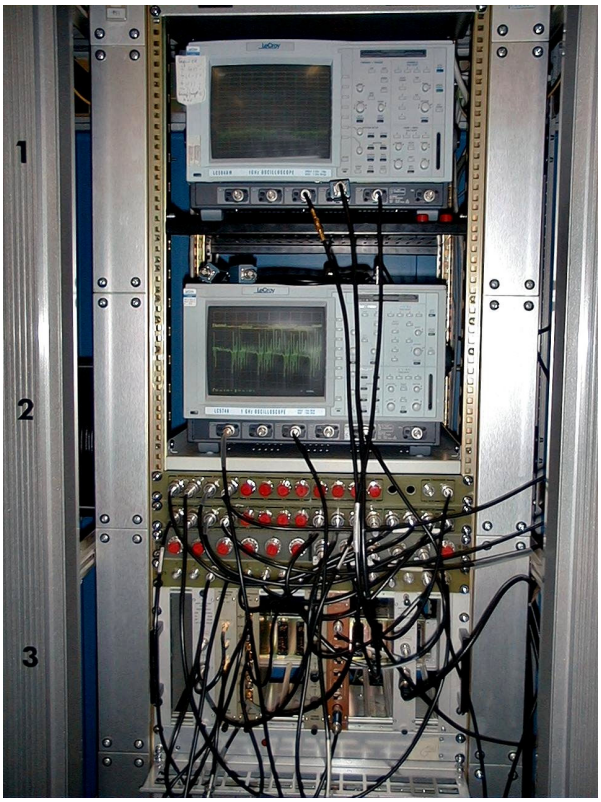


Figure 1: Vertical Feedback running for the positron ring.

In the second phase, the feedback system includes new modules added to the first version setup; it is currently under test. It can allow more flexibility in the system and a better performance of the power stage by computing a more efficient signal by a bunch-by-bunch offset correction scheme.

2 THE INSTALLED TRANSVERSE FEEDBACK SYSTEM

A four buttons beam position monitor produces pulses at the passage of a bunch. It is possible to generate a difference signal connecting the up and down (or left and right) button to a hybrid junction. Fast analog to digital sampling of the signal is performed at base band at the DAΦNE Radio Frequency (368 MHz) by a MAX101A, an 8-bits ADC by MAXIM. The system manages at full speed a continuous stream of data and it has the capability to invert the sign or put to zero the output for single or multiple values in real time. To manage a so fast flow of data, ECL components from the ECLinPS family MC100E (by Motorola) are used. Printed circuit boards with traces at controlled impedance have been designed at the Frascati laboratory. The skew between 16+1 differential signals is a very critical parameter and requires a careful lay out of the PCB traces. A conversion from digital to analog produces the bunch-by-bunch error signal that, after a stage of pre-amplification, is sent as a correcting signal to the power amplifiers. Figure 2 shows the correction signal and the kicker signal. In this implementation of the system local orbit bumps are used to avoid saturation of the power amplifiers due to the static and not useful part of the signal coming from the orbit.

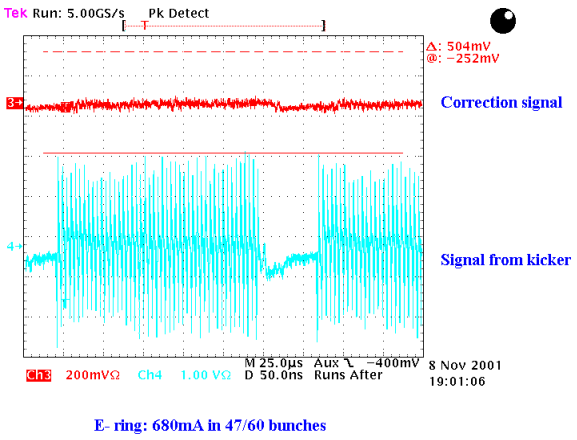


Figure 2: Feedback output signals.

3 THE UPGRADED FEEDBACK SYSTEM

The upgraded version of the system adds two modules to the previous ones to improve flexibility and efficiency of the power stage. The simplified scheme is shown in Figure 3.

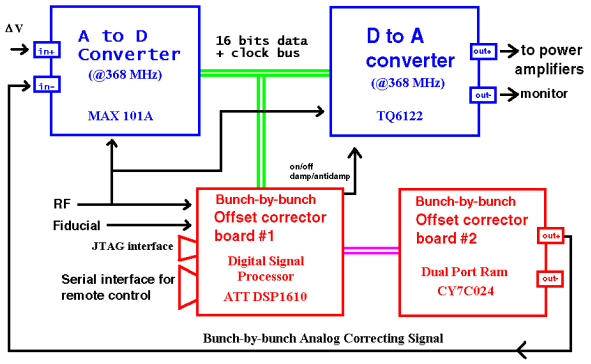


Figure 3: Scheme of the complete feedback system.

A Digital Signal Processor ATT DSP1610 can record a block of data relative to a selected bucket at the revolution frequency (>3MHz). In a typical run, the DSP computes and updates continuously the average value for each bucket (empty or not). Then it puts the results in a dual port RAM (CY7C024 by CYPRESS). An 8 bits digital to analog converter (TQ6122 by Triquint) outputs at 368MHz the value to be subtracted from the input signal. The hold buffer is circular and it is scanned with a proper delay. The four modules are shown in figure 4.

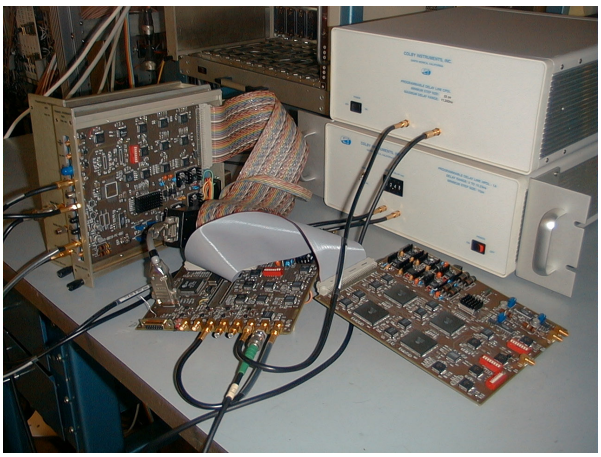


Figure 4: The complete feedback system in the lab.

The digital signal processor can address up to 128 Kbytes of programs, including an FFT routine. It has a JTAG interface for debugging purposes. Moreover, through a 4 MHz synchronous serial interface that can connect up to eight slave devices to a master, it is possible to exchange commands and data following the pseudo language listed in Table 2. The post processing software still is under development.

Table 2: Multipoints serial interface commands list

FO	Feedback on
FF	Feedback off
TE	Test
DA	Damping feedback
AD	Antidamping feedback
OC	Start offset corrector
ST	Stop offset corrector
Rxx	Reduce correction at xx%
Nxx	Number_of_average/100 [0:255], i.e. from zero to 25500
Bxx	Bunch selected [1:120], 0=scan forever on all the bunches
I	Store a block of data for a selected bunch to internal RAM
J	Download a block of data from internal RAM to serial interface
M	Store a block of data for a selected bunch to dual port RAM
L	Download a block of data from dual port RAM to serial interface
Sxx	Select slave device (0:7)
Txx	Compute tune bunch xx (1:120, 0=all)

The system shown in figure 4 is currently under test in the laboratory. A good manner to evaluate its linearity and to prove that the internal skew is adequate can be realized by sending to the input a ramp in the working range of the ADC. Then the digital data cross the modules and at the end of the chain are converted again to an analog signal. The result of this test is shown in figure 5.

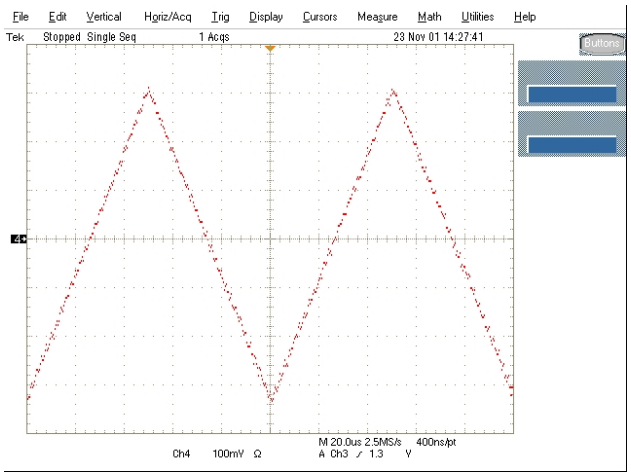


Figure 5: Analog output of the test signal.

4 CONCLUSIONS

The vertical feedback system damps efficiently coherent instabilities in DAΦNE, allowing the storage of 1411 mA in the electron ring and 1150 mA in the positron one. The increase of the stored currents was a crucial item in the improvement of peak and average luminosity. The system has shown a reliable behaviour during more then one year and it has offered in the same time useful diagnostics to manage it correctly. The next upgrade of the system is expected to further improve DAΦNE performance.

5 ACKNOWLEDMENTS

Donato Pellegrini, Olimpio Giacinti, Gianfranco Baccarelli and Giuseppe Fallica under the supervision of Oscar Coiro have taken care of system setups and cabling. Umberto Frasacco has skillfully loaded the boards. F. Ronci and F. Galletti have prepared the printed circuit layouts. Thanks to Pina Possanza for the patient text editing.

Thanks also to Marco Lonza and Daniele Bulfone for ideas and data exchanges on the power amplifiers and to John Fox, Dmitry Teytelman and Shyam Prabhakar for many interesting conversations.

REFERENCES

[1] W. Barry, G.R. Lambertson, C.C. Lo (LBL, Berkeley), 'Electronics Systems for Transverse Coupled Bunch Feedback in the Advanced Light Source (ALS)', SLAC-PEP-II-AP-NOTE-39-93, SLAC-PEP-II-EE-NOTE-05-93, LBL-34405, CBP-NOTE-030, Nov 1993.

[2] J.N. Corlett, W. Barry, J.M. Byrd, G. Lambertson, J. Johnson, M. Fahmie (LBL, Berkeley), J.D. Fox, D. Teytelman (SLAC), 'Transverse Feedback Systems for the PEP-II B Factory', SLAC-PEP-II-AP-NOTE-95-55, SLAC-PEP-II-EE-NOTE-95-10, LBL-37781, CBP-NOTE-156, Oct 1995. Presented at International Workshop on Collective Effects and Impedance for B Factories, Tsukuba, Japan, 12-17 Jun 1995.

[3] W. Barry, J.M. Byrd, J.N. Corlett, J. Hinkson, J. Johnson, G.R. Lambertson (LBL, Berkeley), J.D. Fox (SLAC), 'Design of the ALS Transverse Coupled Bunch Feedback System', LBL-33269, May 1993. 993 Particle Accelerator Conference (PAC 93), Washington, DC, 17-20 May 1993. Published in IEEE PAC 1993:2109-2111 (QCD183:P3:1993).

[4] W. Barry, J. Byrd, J. Corlett, J. Johnson, G. Lambertson (LBL, Berkeley), J. Fox (SLAC). 1995, 'Commissioning of the ALS Transverse Coupled-Bunch Feedback System', In *Dallas 1995, PAC, vol. 4* 2423-2425.

[5] W. Barry, J. Byrd, J. Corlett, M. Fahmie, J. Johnson, G. Lambertson, M. Nyman (LBL, Berkeley), J. Fox, D. Teytelman (SLAC), 'Design of the PEP-II Transverse Coupled Bunch Feedback System',SLAC-PEP-II-AP-NOTE-95-09, LAC-PEP-II-EE-NOTE-95-02, LBL-36469, May 1995. Contributed to 16th IEEE Particle Accelerator Conference (PAC 95) and International Conference on High Energy Accelerators (IUPAP), Dallas, Texas, 1-5 May 1995. Published in IEEE PAC 1995:2681-2683.

[6] M.Tobyama, E. Kikutani, J.W.Flanagan, S.Hiramatsu, “Bunch by Bunch Feedback Systems for the KEKB Rings”, KEK preprints 2001-32, and PAC2001, Chicago, June 2001.

[7] D. Bulfone, R. Bressanutti, M. Lonza, V. Smaluk, L. Tosi, L. Zambon, M. Dehler, R. Ursic, 'Exploitation of the Integrated Digital Processing and Analysis of the ELETTRA/SLS Transverse Multi-Bunch Feedback System', PAC2001, Chicago, June 2001.

[8] A.Drago et al. “High Current Multibunch Operation in DaΦne“, PAC2001, Chicago, June 2001.

WEAP051
nucl-ex/0111002

DEVELOPMENT OF THE SIMULATOR FOR THE GLOBAL ORBIT FEEDBACK SYSTEM USING EPICS

K. H. Kim*, J. Choi, T-Y Lee, J. H. Kim, Yujong Kim, C. Kim, Guinyun Kim[1],
M. H. Cho, W. Namkung, and I. S. Ko†

Pohang Accelerator Laboratory, POSTECH, Pohang 790-784, Korea
[1]Center of High Energy Physics, Kyungpook National University, Daegu 702-701, Korea

Abstract

We have carried out the basic research for the accelerator and tokamak control system based on the Experimental Physics and Industrial Control System (EPICS). We have used the process database and the state notation language (SNL) in the EPICS to develop the simulator which represents as a virtual machine. In this paper, we introduce the simulator of the global orbit feedback system as an example. This simulates the global orbit feedback system under the constraint conditions for Pohang Light Source (PLS) storage ring. We describe the details of the feedback algorithm and the realization of the simulator.

1 INTRODUCTION

When the beam position change takes place during the conventional global orbit correction processes, the photon beam through the beamline is affected, and it results in the alignment of mirrors and monochromators. This is particularly severe for a long beamline such as the undulator beamlines. This problem can be overcome by introducing a local bump at the particular beamline. However, for some light sources, there are not enough corrector magnets to generate local bumps as much as needed. This difficulty can be overcome when we correct the COD under the condition where the beam positions at particular points are not changed. This is our main objective to develop a method of the closed orbit correction under constraint conditions.

2 THEORY

2.1 Ordinary COD correction and regularization

The beam position is normally described as a vector $|\mathbf{x}\rangle$ measured by M beam position monitors (BPM). In order to correct the COD, we need N corrector magnets with their strengths described as a vector $|\mathbf{k}\rangle$. When the corrector magnets kick a beam, the new beam positions $|\mathbf{y}\rangle$ can be described as follows.

$$|\mathbf{y}\rangle = \mathbf{R}\,|\mathbf{k}\rangle + |\mathbf{x}\rangle\,. \tag{1}$$

Here, $\mathbf{R}$ is called the response matrix of $(M \times N)$ dimensions whose components are given by

$$\mathbf{R}_{ij} = \frac{\sqrt{\beta_i \beta_j}}{2\sin\pi\nu}\cos\left(|\Psi_i - \Psi_j| - \pi\nu\right), \tag{2}$$

where ν is the betatron tune of the storage ring, and (β_i, Ψ_i) and (β_j, Ψ_j) are the beta function and the phase function for the i^{th} BPM and j^{th} corrector magnet, respectively. In order to reduce the COD, we have to choose the kick of each corrector magnet satisfying

$$\mathbf{R}^T\mathbf{R}\,|\mathbf{k}\rangle + \mathbf{R}^T\,|\mathbf{x}\rangle = 0\,. \tag{3}$$

It is called the PSINOM algorithm[1].

The COD correction is actually a minimization procedure of S defined as

$$S = \frac{1}{2}\left\{\langle\mathbf{k}|\,\mathbf{R}^T\mathbf{R}\,|\mathbf{k}\rangle + 2\,\langle\mathbf{x}|\,\mathbf{R}\,|\mathbf{k}\rangle + \langle\mathbf{x}\,|\mathbf{x}\rangle\right\}. \tag{4}$$

By using the relations of vector operators which are hyper-dimensional gradient operators, we can get the same result of PSINOM algorithm as shown in eq. (3).

This algorithm includes an inversion procedure of the matrix $\mathbf{R}^T\mathbf{R}$. In some cases, we can get unacceptable corrections due to the ill-posedness of $\mathbf{R}^T\mathbf{R}$. A regularization method is introduced to avoid this problem. In this case, S is written as

$$S = \frac{1}{2}\left\{\langle\mathbf{k}|\,\mathbf{R}^T\mathbf{R}\,|\mathbf{k}\rangle + 2\,\langle\mathbf{x}|\,\mathbf{R}\,|\mathbf{k}\rangle + \langle\mathbf{x}\,|\mathbf{x}\rangle\right\} + \frac{1}{2}\,\langle\mathbf{k}|\,\alpha\,|\mathbf{k}\rangle\,. \tag{5}$$

where α is the regularization parameter. Then the minimum corrector kicks can be determined by

$$\left(\mathbf{R}^T\mathbf{R} + \alpha\mathbf{I}\right)|\mathbf{k}\rangle + \mathbf{R}^T\,|\mathbf{x}\rangle = 0\,. \tag{6}$$

This equation represents the modified PSINOM algorithm, and we can relax the inversion problem of the singular matrix by using the diagonal matrix $\alpha\mathbf{I}$ when $\mathbf{R}^T\mathbf{R}$ is singular[2].

* kimkh@postech.ac.kr, http://fal.postech.ac.kr
† isko@postech.ac.kr

2.2 Method with constraint conditions

After having the new closed orbit with the minimum distortion from eq. (6), the new orbit is generally different from the original orbit. Sometimes, this difference can be taken place at very sensitive locations such as the entrance and the exit of an undulator. If the beamline is well aligned for this undulator, a COD correction should be avoided in this region. A constraint condition can be described in terms of the beam position at $i^{\rm th}$ BPM such as

$$\langle \mathbf{R}_i \, | \mathbf{k} \rangle + x_{0i} = x_i \, . \tag{7}$$

Here, $\langle \mathbf{R}_i |$ is the $i^{\rm th}$ row of the response matrix $\mathbf{R}$. Also, x_{0i} and x_i are the beam positions before and after the correction, respectively. Since we want to keep this position unchanged, $\langle \mathbf{R}_i \, | \mathbf{k} \rangle$ should be zero. If there are L BPMs involved in the constraint condition, we can write the constraint condition as follows.

$$\mathbf{C}^T \, | \mathbf{k} \rangle = 0. \tag{8}$$

Here, $\mathbf{C}^T$ is the $(L \times N)$ sub-matrix of the response matrix. Each component of $\mathbf{C}^T$ corresponds to the BPM involved in the constraint condition. We also assume that $|\mathbf{k}\rangle$ has a non-trivial solution.

We now add this constraint condition to the modified PSINOM algorithm to obtain the new S such as,

$$S = \frac{1}{2} \left[\langle \mathbf{k} | \, \mathbf{R}^T \mathbf{R} \, | \mathbf{k} \rangle + 2 \, \langle \mathbf{x} | \, \mathbf{R} \, | \mathbf{k} \rangle + \langle \mathbf{x} | \mathbf{x} \rangle \right] + \frac{1}{2} \, \langle \mathbf{k} | \, \alpha \, | \mathbf{k} \rangle + \langle \mathbf{\Gamma} | \, \mathbf{C}^T \, | \mathbf{k} \rangle \, . \tag{9}$$

Here, $\langle \mathbf{\Gamma} |$ is the Lagrangian multiplier, and it is an L dimensional vector. By following the derivative to the corrector strength, we can get the vector $|\mathbf{k}\rangle$ which minimizes the closed orbit distortion outside the constraint region such as,

$$(\mathbf{A} + \alpha \mathbf{I}) \, | \mathbf{k} \rangle + \mathbf{R}^T \, | \mathbf{x} \rangle + \mathbf{C} \, | \mathbf{\Gamma} \rangle = 0. \tag{10}$$

Here, we define the square matrices of $N \times N$ dimensional $\mathbf{A}$ and $L \times L$ dimensional $\mathbf{D}$ as follows.

$$\mathbf{A} = \mathbf{R}^T \mathbf{R}, \tag{11}$$

$$\mathbf{D} = \mathbf{C}^T \, (\mathbf{A} + \alpha \mathbf{I})^{-1} \, \mathbf{C} \, . \tag{12}$$

Eq. (10) can be rewritten as

$$| \mathbf{\Gamma} \rangle = - \mathbf{D}^{-1} \mathbf{C}^T \, (\mathbf{A} + \alpha \mathbf{I})^{-1} \, | \mathbf{x} \rangle \ . \tag{13}$$

Now, we can remove the Lagrangian multiplier $|\mathbf{\Gamma}\rangle$ in eq. (10) by using the above equation. Then, we can finally get the kick values of the corrector magnets as follows.

$$| \mathbf{k} \rangle = - \, (\mathbf{A} + \alpha \mathbf{I})^{-1} \left\{ \mathbf{I} - \mathbf{C} \mathbf{D}^{-1} \mathbf{C}^T \, (\mathbf{A} + \alpha \mathbf{I})^{-1} \right\} \mathbf{R}^T \, | \mathbf{x} \rangle \ . \tag{14}$$

3 DEVELOPMENT OF SIMULATOR USING EPICS

The algorithm developed in the previous section has been successfully tested for the Pohang Light Source (PLS) operation. The new correction code is written in C language and it is installed in one of the operator consoles which is a SUN workstation. Although the PLS control system is not using Experimental Physics and Industrial Control System (EPICS) which is used in many accelerator laboratories world-widely, there is a plan to upgrade the control system based on EPICS technology[3, 4]. As a part of such upgrade activity, we have started the development orbit correction algorithm with EPICS. Since we do not have any EPICS-based control system yet, we decide to develop the orbit correction simulator using EPICS.

In order to seek the way to adopt our orbit correction algorithm into EPICS, we have considered two ways: one is based on the subroutine record and the other is using the state notation language (SNL) program. The first method needs new record support that runs the orbit correction algorithm. To do this, the correction code is required to be written upon the protocol required by the record support such as the entry structure and the callback structure. This approach gives relatively fast response because this method uses database access and can access the record by process passive mode[4, 6]. This is a good feature for the realtime system but the large portion of the orbit correction code we have already tested must be rewritten according to the protocol of the record support.

One of important tasks of EPICS input/output controller (IOC) is the sequencer that runs programs written in SNL. The SNL considers the control object as the state machine and treats transitions between states. The sequencer monitors the transitions for the SNL and runs callback functions using the entry table the corresponding program written in SNL. Since the sequencer accesses to the record via channel access (CA) and the record access is only possible through non-process passive mode, there are some restrictions in access time or the treatment of records. However, this method can directly embed the program written in C language. Also, unlike the subroutine record, this method can remove program tasks without rebooting the system, which gives the code debugging very easy[4, 5]. Thus, the second method gives more benefits when the system does not require heavy realtime demands. Upon reviewing the two methods, we have decided to use the latter method.

3.1 SNL program

The orbit correction simulator is developed in IOC level and has the SNL program embedding C codes and several database records, as shown in Fig. 1. There are two parts in the SNL program: one for the feedback including the orbit correction algorithm and the other for the simulator which emulates the PLS storage ring. The latter calculates orbit changes from *ai:KICK$(kick_no)* records which store the corrector strengths obtained from the response matrix

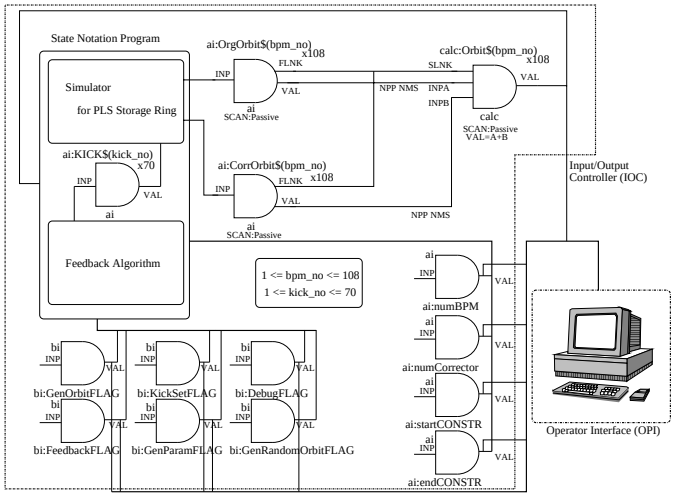


Figure 1: Schemematic diagram of the orbit correction simulator

measured at the PLS. This process is actually a product of matrix and column vectors. The calculated orbit changes are stored at *ai:CorrOrbit$(bpm_no)* records. In the next step, the corrector strengths s calculated by the algorithm described in the previous section and the results are stored at *al:KICK$(kick_no)*.

3.2 Database and record linkage

The SNL program we have developed consists of four state sets. Since they are linked together and interact dynamically via EPICS database, we need to understand the database and its records as well as the linkage between them. The *ai:KICK$(kick_no)* record uses the analog input record to represent the corrector strength. There are 70 records altogether as representing 70 correctors in the PLS storage ring. They are linking the simulator and the feedback in a parallel manner. The index *$(kick_no)* is the integer value between one and 70.

The records of *ai:OrgOrbit$(bpm_no)* and *ai:CorrOrbit$(bpm_no)* are the orbit distortion before the feedback is applied and the orbit change after the feedback, respectively. They are using analog input records. Since these records represents the changes at the PLS BPM, *$(bpm_no)* is the interger value between one and 108. The superposition of these two records gives new orbit. This can be done by the calculation record *calc:Orbit$(bpm_no)*. Since this calculation record is linked forwardly from *ai:OrgOrbit$(bpm_no)* and *ai:CorrOrbit$(bpm_no)*, the superposition is newly calculated whenever the values of two records are changed.

On the other hand, *ai:numBPM*, *ai:numCorrector*, *ai:startCONSTR*, and *ai:endCONSTR* are representing the BPMs and correctors used in the correction algorithm, start and end point of the constraint region, respectively. They are using analog input records and notifying the SNL program if necessary.

The other records are binary input records and they are representing necessary status flags. They are used as the mediator of state transitions between state sets in the SNL program.

4 CONCLUSION

We have developed the COD correction algorithm under the constraint condition where the beam position at particular point is not changed. The new algorithm is based on the modified PSINOM algorithm which includes the regularization process in order to avoid the inversion problem of the ill-posed response matrices.

We have confirmed that this algorithm is working well and is in good agreement with the experimental results [7]. Even though the PLS is planning to upgrade their control system with EPICS, there is no working EPICS based control system at PLS. Due to this, we have developed the orbit correction simulator using C-code embedded SNL program based on EPICS technology. This simulator part can be replaced by the real control system with minor changes after the completion of the upgrade.

5 REFERENCES

[1] W. Herr: "Algorithms and procedures used in the orbit correction package COCU," CERN SL/95-07 (AP), 1995.

[2] Y. N. Tang and S. Krinsky: Proc. AIP Conf. **315** (AIP Press, 1993) 87.

[3] See URL"*http://www.aps.anl.gov/epics*"

[4] Martin R. Kraimer: "EPICS IOC Application Developer's Guide," APS/ANL, 1998

[5] Andy Kozubal: "State Notation Language and Sequencer User Guide," LANL, 1995

[6] Philip Stanley, *et al.*: "EPICS Record Reference Manual," LANL, APS/ANL, 1995

[7] Kukhee Kim, Jinhyuk Choi, Tae-Yeon Lee, Guinyun Kim, Moohyun Cho, Won Namkung, In Soo Ko: Jpn. J. Appl. Phys. **40** (2001) 4233.

WEAP053
physics/0111092

OPTIMIZED FILLING OF THE ALS STORAGE RING

Jan Pusina, Lawrence Berkeley National Laboratory, 80-101, Berkeley, CA, 94720

Abstract

The Storage Ring at the Advanced Light Source is capable of being filled in selectable buckets. Typically, 276 consecutive buckets out of 328 possible are selected. It has been found that the amount of charge injected is not uniform bucket-to-bucket, sometimes varying by 100% or more. This variability causes problems with beam stability, and it may shorten the beam lifetime. We have devised a feedback system that allows for beam filling in an orderly manner, yielding bunch heights uniform to within 17%. At fill time, multiple bunches are injected into the Storage ring first, then, using feedback, selected single bunches are used to top off the fill.

1 INTRODUCTION

The Storage Ring beam is monitored by a BPM in summing mode and wired into the control room to a Tektronix TDS 784A oscilloscope. A GPIB interface connects the scope signal to a Labview program, Ogetwave.vi[1], running on a PC. This program detects the bunch stream, processes the information, and periodically stores the file on a server. Another program written in Delphi, SRInject.exe, runs simultaneously, reading the files, and further analyzing and processing the bunch data. This program contains an algorithm which uses the individual bunch heights as feedback and selectively fills each of them according to how much charge is required to bring them up to an even height while filling the storage ring to a current of 393 mA. Then a "camshaft" bucket is automatically filled to 10mA in the dead space after the bunch train. By this time this last step is finished, it is assumed the multibunch buckets have decayed a few milliamps, bringing the total current to 400mA.

2 FILLING THE STORAGE RING

The ring is filled using a graphic method of bucket selection (see Fig. 3). The start and end buckets are specified, as well as the pattern of spacing and cycles that are made necessary by the 8 nanosecond spacing of gun bunches compared with the 2 nanosecond spacing of storage ring buckets. The 276 bucket filling sequence is begun with target buckets 1, 13, 25 . . . and intervening numbers filled in by the 3 gun bunches. I.e., when bucket 1 is targeted, buckets 5 and 9 are automatically filled, because of the three gun bunches. After the first part of the cycle is complete and target bucket 265 has been reached, the pattern starts over with 2, 14, 26, . ., then 3, 15, 27, . . ., and finally 4, 16, 28, ... Thus all buckets are filled once. In the case of non-optimized fills, this procedure would run freely, repeating as many times as necessary to achieve 393 mA, starting and ending at unpredictable target buckets. Since it would not have filled each bucket the same number of times, and the fact that each bucket would have received unpredictable amounts of beam on each shot, the fill would be uneven. See Figure 2.

3 OPTIMIZATION

The solution to the uneven fill is to regulate the amount of charge in each bucket. One problem immediately encountered when observing the oscilloscope trace and the digitized version of it on SRInject.exe, is that there is a distortion of the heights of isolated bunches, or bunches that are significantly higher than their neighbors, apparently due to a cabling mismatch. At this time, no hardware solution of this problem has been found, so a formula for compensation of this error has been derived in software. This formula makes use of the fact that the increased height of a given bunch is inversely proportional to the height of the bunch immediately preceding it. If the preceding bunch is zero, the height of the bunch is greater by an error, e, and the compensated height, I_c, of the bunch in question is calculated as

$$I_c = I_1 - e\left(1 - I_0 / I_1\right),$$

where I_0 and I_1 are two consecutive bunch heights.

At fill time, the Storage Ring current has decayed by about one halflife, i.e., to about 200 mA. At this point the standard multibunch fill is executed for N cycles, i.e., each of the 276 buckets is targeted N times. Next, a determination is made whether there is room for another cycle. If not, the gun output is reduced so that the next cycle fills to just under the nominal fill point of 393 mA, or a threshold level (V_t) of 1.42 mA per bucket. To derive the proper gun output, its output was graphed as a function of grid bias. The graph is a typical triode curve (see Figure 1.) To obtain the

[1] Jones, Orland "OGETWAVE.EXE", ALS note, 2001.

reduced gun output, y, the gun grid bias is increased until

$$y = \frac{(V_t - V_{\max})}{dI/dT},$$

where V_t - $V_{\max}$ is the difference between the threshold voltage and maximum bunch voltage, and dI/dT is the fill rate.

Once the fill has been extended such that any bucket is >=V_t - $0.5dIdT$, the single bunch optimization, or topoff begins. In this case, the heights of the buckets are recorded, and all buckets less than $V_t - 0.5dIdT$ are targeted for a single bunch injection. Assuming the buckets to be of unequal charge, the last step is repeated until all buckets are within $V_t \pm 0.5dIdT$. The gun output may be optionally lowered in the final iteration, as in multibunch fill, to make the topoff smoother. Fills with and without optimization and their spectra are shown in Figure 2.

The optimization procedure can be executed in a 3-button operation: Multibunch Prefill, Optimization with Single Bunch Topoff, and Single Bunch Camshaft fill. The simplification of the procedure with just a few strokes minimizes operator errors as the fill is repeated throughout the week.

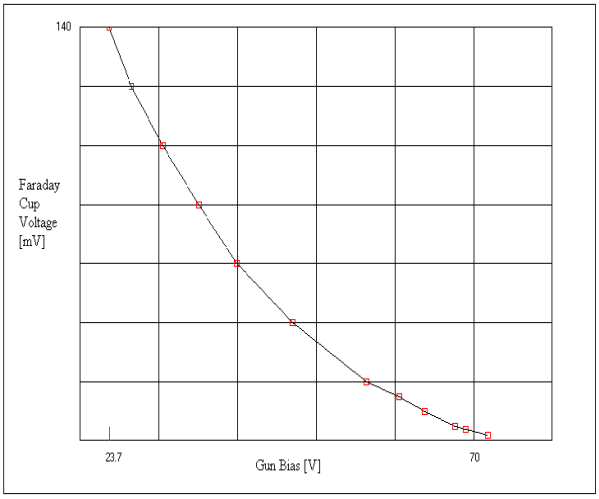


Figure 1: Gun output vs. Grid Bias

4 THE CAMSHAFT BUCKET

In the final phase of filling, the feedback scheme described above is also used to inject a camshaft bunch to a value of 10 mA. This bunch is used for precise timing of certain experiments, e.g., chemical dynamics, but its separation from and unequal charge with the other bunches is transparent to other users.

All phases of filling can be done while observing a dynamic display of the buckets being filled as a bar graph, with the possibility of selecting an individual bucket and reading its height.

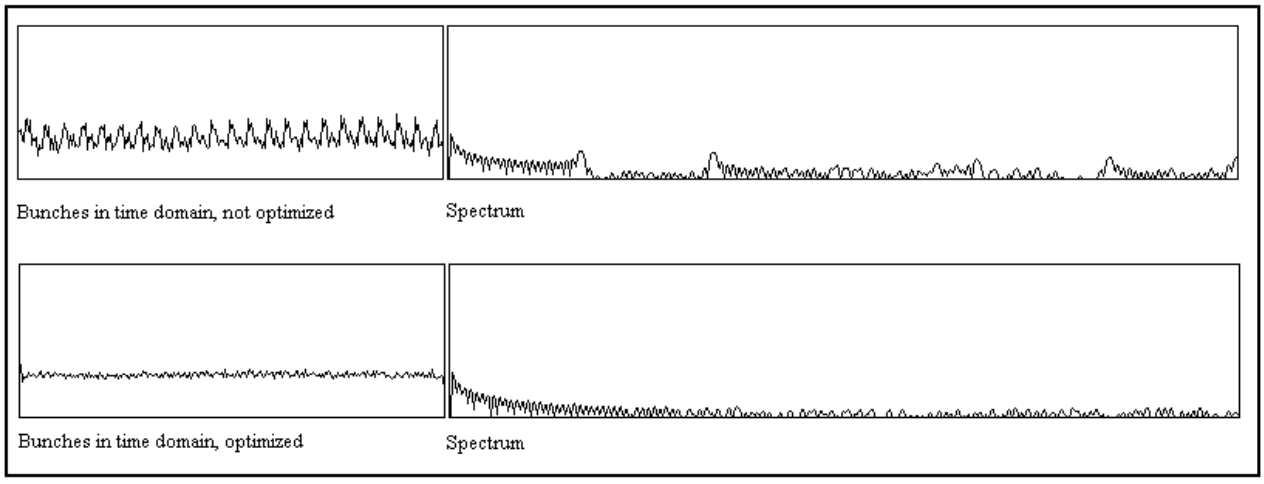


Figure 2: Bunches and Spectra

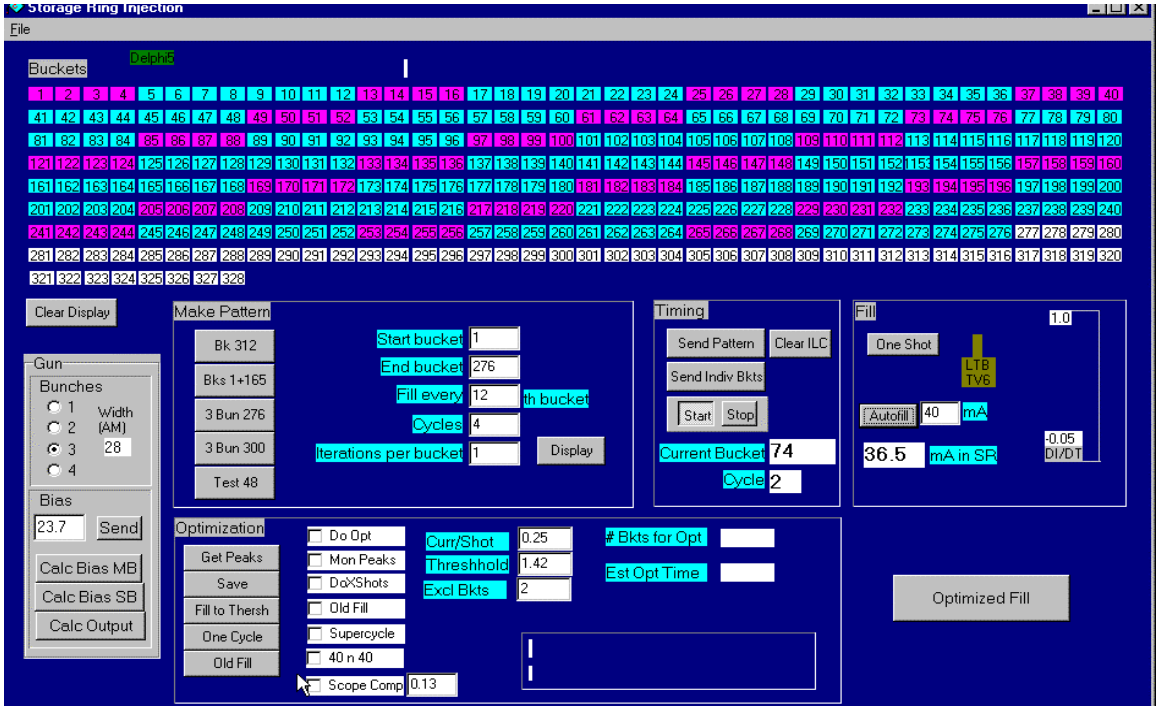


Figure 3: SRInject.exe

WEAP058
physics/0111214

PLD-BASED RECONFIGURABLE CONTROLLERS FOR FEEDBACK SYSTEMS

V.F. Gubarev, L.M. Yakovleva, N.N. Aksenov, Space Research Institute, Kiev, Ukraine
A.V. Palagin, V.N. Opanasenko, V.G. Sakharin, Institute of Cybernetics, Kiev, Ukraine

Abstract

Recently, interest has grown for application of reconfigurable devices in robust and adaptive control systems. The main advantage of such devices is that its structure is not fixed and may be varied depending on the currently used control algorithm. In this paper a new PC-based reconfigurable microcontrollers (RMC) are offered for Experimental Physics control systems. Programmable logic devices (PLD) XILINX Inc. of a type FPGA are used in RMC. Reconfigurable devices are connected to PC as the coprocessor through the system bus (for example, PCI) forming the system with programmable architecture as a result. Each control algorithm is realized as a file of a PLD-configuration. The set of model- based control algorithms make up the functional library. Model selection procedure on the base of control relevant identification is a part of united control system design. Methodology of the joint and iterative identification and control is proposed which well suited for the realization in RMC and convenient for users. Control design is arranged as follows. First the identification experiment is carried out and approximate model is reconstructed using special identification algorithm and model structure in normalized form. Then select from the library the appropriate liner or nonlinear parametrized control law. After that the parametric class of controllers which guarantee stability of the closed loop system is automatically determined. The best robust controller in given class is obtained by parameter tuning procedure, which give opportunity to impact performance of the transient processes in closed system without loosing asymptotic stability.

1 INTRODUCTION

The microcontrollers are applied in modern control system's of technological processes, instrumentation, robots - manipulators, physical and other systems of real time, intended for to execution of the large sizes of calculations. They are realized, as a rule, or as sets of modules, which are built in the computer or as standalone control system's of the equipment, technological processes etc. For record of the control algorithm to the memory can be used ROM (the program is written at manufacture of the microcontrollers or with the help of the programmator) and RAM allowing to rewrite the program practically an infinite number of times.

Typically, the Experimental Physics and Industrial Control System (EPICS) is available on many platforms: VME -, VXI- and PC-based input output controllers. Advantage PC-based controllers is that there is only one software model for all PC motherboard, whereas there are typically many different software models for each of the many different VME-based of single board computers. In this paper new PC-based class of reconfigurable microcontrollers (RMC), which architecture varies with a program way is considered. The difference between reconfigurable and programmable microcontrollers is that their structure is not fixed and varies depending on the control algorithm. Thus the computational efficiency of algorithm is bounded by hardware only that is essential for real time systems.

PLD (programmable logic devices) Xilinx Corporation of a type FPGA (Field Programmable Gate Array) are used as element base for RMC mainly [1]. Essential advantage of PLD is their universality and possibility of fast program customization on the given control algorithm.

2 REALIZATION OF SET-VALUED CONTROL SYSTEM

Offered RMC is a new approach to construction of control system of experimental physical plants. This approach allows for multiple implementation to change both the parameters and structure, laws and strategies of control. It is especially important for physical plants when the conditions of experiment, targets and the tasks are varied. It is desirable for the experimenters and operators working with the system, without substitution of the equipment, to change algorithms and target of control, not breaking thus of stability and other important dynamic parameters of the system.

The system consists of standard hardware and reconfigurable coprocessors on base FPGA type PLD and set of programs realizing algorithms of iterated synthesis of control with identification of an approximating model. The creation of the well-structured library of algorithms and structural implementations, appropriate to them stored as files of

configurations, is supposed. The task of synthesis in this case consists in a choice of optimal pair (algorithm-structure) for concrete source experimental data. Thus, the task of optimal synthesis is reduced to the task of an optimal choice on previously generated (and constantly extended) set of solutions. Distinctive feature of the offered controller is the possibility of creation of two-level nonlinear stabilization system, in which the lower automatic level provides stability and robustness of the close-looped system, and the top level allows to the engineer flexibly to operate quality of regulation.

The information for synthesis is measuring responses of the system on inputs. It is supposed also that the input and output information can be entered in PC using converters (analogue / digit and digit / analogue).

3 RECONFIGURABLE STRUCTURES

Usually structures of reconfigurable devices contain one or several chips PLD, memory for storage a configuration files (PROM or FLASH-memory), port of testing / debugging JTAG IEEE Std. 1149.1, SRAM or DRAM-memory for caching data, plug and controller for connection to buses ISA, PCI (for modules-coprocessors), plugs for connection of peripherals etc. The configuration file can be written down in ROM and automatically (at power up) to boot in a chip PLD. In the other mode a configuration file loads in a chip PLD from RAM of the computer.

The process of designing of PLD-based devices is completely supported by tools CAD, for example, Foundation Series of the Xilinx Corporation. Structure Foundation Series includes the library of parametric modules [2], which supports functional solutions in a range from small blocks (adders, registers, multipliers) up to larger blocks (correlators, DFT- 32, 64 and 128 points, FFT - 1024 points). The possibility of reconfiguration allows to make modifications of a working product or to use this product for implementation of the various control algorithms taking into account the specific features of hardware. The architecture of chips of series Virtex is completely compatible with technology IRL (Internet Reconfigurable Logic). Technology IRL gives the possibility of remote dynamic reconfiguration through the Internet [3], and also share development of standalone system or chips through Internet by various workgroups.

The abstract architecture RMC can be described as follows:

$D = \langle G, B_i, F \rangle$,

where: $G = \{G_i\}$ is set of control objects ($i = 1 \div n$);

$B_i = \{B_{ij}\}$ is set of the control algorithms, $B_i : X_{ij} \Rightarrow Y_{ij}$ where $\{X_{ij}\}$ is the set of input signals and $\{Y_{ij}\}$ is the set of output signals for i - th object ($j = 1 \div m$);

$F = \{F_\mu\}$ is a configuration files set ($\mu = 1 \div k, k = n \times m$), defining structure of algorithms B_{ij} for control objects G_i. After loading a configuration file in FPGA the structure of the device for implementation of appropriate algorithm include the structure of operational and internal control device will be generated.

4 IDENTIFICATION FOR CONTROL

It is offered to realize the model-based control system. The structure of a model can be known or to be determined in identification experiment The uniform methodology is developed which provides identification well suited for set of the control algorithms, i.e. identification and control are completely coordinated with each other. As the target is put to change by a desirable manner initial dynamic characteristics of the controlled object, this process can be carried out by an interactive way with restoring in identification experiment only of that part of a model, which we are going to change with the help of feedback. For an obtained submodel in the class of results guaranteeing stability and other important properties of transients, for example, robustness, the interactive way selects best feedback from the point of view of the experimenter. Thus can vary both parameters, and structure of feedback. If the result does not satisfy the experimenter, then identification of the system is repeated, but already for the close-looped system. Thus a submodel is identified again and parameters will be improved with the help of new feedback. The iterative process can proceed while obtaining desirable result.

For such iterative procedure of control synthesis was converging to desirable result, in the developed methodology the following two ways are used. The first one concerns to a choice of structure of an identifying model, which accepts splitting it on standalone blocks or subsystems. Usually it is canonical or normal forms of representation of the equations describing the system dynamic. The equivalent representation can be written down for discrete analogue of this dynamic system.

In the close-looped system the eigenvalues of the matrix corresponding to only subsystem are varied. All other roots of the characteristic equation are saved. Therefore second way of the developed methodology consists in estimation of considered submodel on base output of a current state. The estimation method advanced in [4] with use of observation results on a finite sliding interval is applied for this purpose.

5 LIBRARY OF CONTROL ALGORITHMS

Set of the admissible feedback laws, which can be realized in the control system, in a general case is written in matrix form as follows:

$$U = C^{(i)}(t, x) \cdot x, \; i = \overline{1, N},$$

where x is the state vector of the dynamic system and set of matrices represents the some classes of the control laws.

Each class of appropriate controllers has the structure, in which the concrete implementation is satisfied by the defined set of parameters. For example, the class $C^{(0)}$ will generate a subset of linear regulators, which satisfy to a matrix $C^{(0)}$ with the constant elements. Other classes correspond to the nonlinear laws of regulation of various structures.

The distinctive feature of the offered methodology of share synthesis of control and identification is the iterative procedure with submodels of the low order used on each step and the nonlinear feedback laws. The nonlinear laws have more wide potentialities then linear ones, but their synthesis by known methods was rather bulky. It is proposed the formalized procedure of construction of set of nonlinear regulators on the base of Lyapunov functions and Sylvestr inequalities allowing to obtain areas of acceptable values of parameters, to select a subset of robust regulators, to optimize parameters of a regulator with the regard for the system of contradictory criterions. The structures of nonlinear regulators with tuning parameters are generated which can be changed over a wide range, effecting on quality of transients, not breaking thus of stability of the close-looped system.

6 CONTROL SYSTEM STRUCTURE

The library of configuration files (LCF) can settle down as in internal memory RMC, and, at the large size, in RAM PC. The swapping of configuration files from memory PC in RMC is made if necessary. The realization of the control system with PLD-based controller assumes library-building configuration files for hardware implementation of set of the control algorithms. Further on the base of this system and generated library LCF the experimenter in an interactive mode carries out the process of identification and synthesis of control. The experimenter on the display PC directly observes a system response to anyone controls and determines most appropriate feedback.

The external digital ports allow to carry out one- and multichannel input/output data processing Besides RMC can be used autonomously, irrespective of the computer. Additional logical resources in this case are released at the expense of absence of the controller of the external bus.

7 MANUSCRIPTS

Customization of structure for realization of chosen algorithm (or its fragment) and it implementation in a chip at a gate level allow to increase speed of the system in comparison with program solutions and to execute set of various algorithms with speed of specialized hardware.

The work of the experimenter with the offered control system does not require a special knowledge of the control synthesis and all operation is reduced to simple following of the instructions and recommendations applied to the system.

REFERENCES

[1] The Programmable Logic Data Book, Xilinx Inc., 2000.
[2] Core Solutions Data Book, Xilinx Inc., 1998.
[3] W. Westfeldt, “Silicon Xpresso and Internet Reconfigurable Logic”, Xcell, Xilinx Inc., 1999, • 32, pp.10-12.
[4] V.F. Gubarev and N.N. Aksenov, “Applying the Regularization Method to Estimation Problems” Journal of Automation and Information Sciences. 1995, Vol.27, N.1, pp.45-54.

WEAP059
physics/0110071

THE LIGO SUSPENDED OPTIC DIGITAL CONTROL SYSTEM

J. Heefner, R. Bork, LIGO Project, California Institute of Technology, Pasadena, CA 91125, USA

Abstract

The original LIGO[1] suspension control system [1] used analog circuitry to implement the closed loop damping required for local control of each of the suspended optics. As installation and commissioning activities have progressed, it has become apparent that these controls do not provide the performance and flexibility that will be required for operation of the interferometers. Recent developments in ADCs, DACs, increased processor speed and performance, and the use of reflective memory have made a digital alternative possible. This paper will describe the real-time digital servo control systems that have been designed, developed and implemented for the LIGO suspended optics. In addition, the paper will describe how the suspension controls have been integrated into the overall LIGO control and data acquisition systems [2].

1 INTRODUCTION

The LIGO interferometers located in Hanford, Washington and Livingston, Louisiana are Michelson laser interferometers enhanced by multiple coupled optical resonators. These coupled optical resonators are 4 kilometer long Fabry-Perot cavities placed in each arm of the interferometer. The mirrors that form the cavities are suspended from a single loop of wire mounted inside suspension cages that are, in turn, mounted on seismically isolated optical platforms within the LIGO vacuum system. Control of the optic is achieved using voice coil actuators that act on magnets attached to the surface of each optic. Shadow sensors are used to measure movement and orientation of the optic with respect to the suspension cage.

There are two types of optic suspensions in LIGO: the Large Optic Suspension and the Small Optic Suspension. Generically these suspension types are identical and only differ in optic size, dynamic range of motion and absolute noise floor of the controls. In the case of the Large Optic Suspension controls the key requirements are:

- *Dynamic range of longitudinal motion: 20um$_{p-p}$*
- *Dynamic range of angular motion: 500urad$_{p-p}$*
- *Noise: 5 x 10^{-20}*(f/40)$^{-2}$ for f>40Hz*

Key requirements of the Small Optic controls are:

- *Dynamic range of longitudinal motion: 27um$_{p-p}$*
- *Dynamic range of angular motion: 1500urad$_{p-p}$*
- *Noise: 3 x 10^{-18}*(f/40)$^{-2}$ for f>40Hz*

The Large and Small Optic controls have two functional requirements:

- Provide for "local" damping of the suspended optic using the shadow sensors and voice coil actuators (nominally 6 Large and 7 Small)
- Provide a means for the LIGO Length (LSC) and Alignment (ASC) controls to control the longitudinal position and orientation of the optic

2 HARDWARE AND SYSTEM DESIGN

In the original LIGO suspension control system, the local damping loop for each optic was implemented using a completely analog solution. In this implementation, the shadow sensor signals were amplified and combined to form signals corresponding to longitudinal position, pitch and yaw readings. These signals were then filtered by a 10-pole Chebychev filter with a zero at DC. The filtered signals were then combined with inputs from the LSC and ASC to form control signals in the position, pitch and yaw degrees of freedom. These signals were passed to an output matrix and sent to each of the actuators. Early on in the commissioning of the LIGO interferometers it was recognized that this analog solution did not provide the performance or flexibility that was required for operation of the interferometers at their design sensitivity. Some of the problems were:

- The velocity damping provided by the Chebychev filter was not necessarily optimal and in some cases a DC coupled servo was desired. In other cases it was desired to have different filter characteristics for different optics or different modes of interferometer operation. These characteristics included gain bubbles at the micro-seismic peak (~0.16 Hz).
- The input and output matrices were simple gain stages. A more optimal solution required frequency shaping in the matrices.
- The interface to the LSC and ASC systems was via cables and connectors with the ASC and LSC signals originating in racks many meters away. Interference and noise was an issue for these sensitive signals.
- The ASC and LSC systems [3] were implemented using digital servos and a more integrated

[1] Laser Interferometer Gravitational-Wave Observatory

approach including the suspension systems was desirable.

After carefully considering the alternatives and the operational issues involved with replacing existing systems it was decided to pursue a digital alternative for the suspension controls. This solution uses VME based CPUs, ADCs, DACs and reflective memory modules. The models and types are listed in Table 1.

Table 1: VME modules used in systems

Type	Model	Characterisitcs
CPU	VMIC[2] Pentium III	850MHz to 1GHZ, VxWorks OS
ADC	ICS[3]-110B	16 bit, 32 Channels
DAC	Pentek[4] 6102	16 bit, 8 channels
Reflective Memory	VMIC 5579	PMCbus, single mode and multi-mode fiber

All custom analog electronics used for signal conditioning and actuator drive are mounted in 19 inch rack mount chassis or Eurocard format crates located in the suspension racks. These custom electronics are typically very low noise (1-2nV/√Hz) high dynamic range circuits. The timing for the ADCs and DACs is derived from the same GPS (Global Positioning System) based timing system used by the LIGO data acquisition system [2].

Since the suspension controls act as the actuation point for the ASC and LSC systems, the suspension controls had to be integrated into their reflective memory networks. There are three reflective memory loops for the ASC and LSC controls, one for each arm of the interferometer and one for the corner station. In addition there is a reflective memory network for the LIGO data acquisition system. Each node of the network is connected in a loop using single mode or multi-mode fiber. Data written to a memory location at one node is passed to each node in the loop. In this way data can be shared among nodes.

3 LOCAL DAMPING LOOP

All of the software is modular and built around a block of code that contains gain, 3 filters (up to 4 second order sections each), an enable, offset adjust and an excitation input. Each of the 3 filters can be enabled or bypassed on the fly by the operator. Figure 1 shows the structure of a basic code building block (BCBB).

All filters are implemented using IIR filters formed by second order sections (SOS1, SOS2, SOS3). The coefficients for each filter are stored as a text file that is read by the code on start up. This allows the operator to select different servo filters on the fly by enabling or bypassing various filters, or editing the filter text file and restarting the system can change the entire configuration.

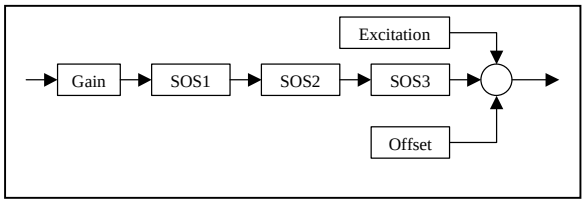


Figure 1: Basic Code Building Block (BCBB)

The basic code blocks are then combined to form the algorithms for each of the suspension local damping loops. The figure below is a block diagram of the code necessary to form the local damping loop for a single optic.

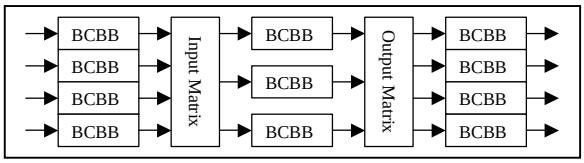


Figure 2: Block Diagram of Code For A Single Local Damping Loop

In the local damping loop for each optic, the four sensor inputs are combined to form the three degrees of freedom: position, pitch and yaw. These are shown top to bottom, respectively between the input and output matrices in Figure 2. The function of the output matrix is to form the four actuator signals from the three degrees of freedom. It should be noted that there is also a transverse degree of freedom for each optic that is not shown in the figure. This degree of freedom is handled separately by a single input, single output control loop formed by basic code building blocks. Not shown in the diagram is the addition of the length and alignment signals supplied by the LSC and ASC systems. These signals are added to their respective degrees of freedom just prior to the output matrix.

4 SYSTEM ARCHITECTURE

One of the functions of the suspension controls is to provide the LSC and ASC actuation on each of the mirrors. This function requires that the suspension controls be distributed throughout the LIGO site and also that they be tightly coupled to the LSC and ASC systems. This is accomplished by using multiple CPUs and VME crates coupled via fiber optic reflective memory loops. These reflective memory loops allow sharing of data from one processor to the next. The

[2] VMIC, Huntsville, AL, USA
[3] Interactive Circuits and Systems, Ltd., Gloucester, Ontario, Canada
[4] Pentek, Inc., Upper Saddle River, NJ, USA

higher bandwidth requirements of the LSC system drive the clock rates for the actuator outputs to 16384 samples per second, far higher than would be required for the local damping loop alone. For these reasons it has been decided to distribute local controls among five VME crates as shown in Table 2.

Table 2: VME Crate Locations and Functions

Location	Clock Rate	Functions
Vertex Crate 1	2048	Read sensor inputs for all vertex optics Perform input matrix and servo filters calcs for all vertex optics Perform output matrix calcs and output actuator signals for all SOS Perform mode cleaner length controls DAQ/GDS interface for all suspensions
Vertex Crate 2	16384	Output matrix calcs for up to three vertex LOS LSC and ASC interface for up to three vertex LOS
Vertex Crate 3	16384	Same as output crate 1
X End	16384	Input and Output matrix calcs for X end ASC X end functions
Y End	16384	Input and Output matrix calcs for Y end ASC Y end functions

The vertex input crate performs the equivalent of 750 second order IIR filter calculations 2048 times per second. Each of the output crates perform the equivalent of up to 80 sections at 16384 times per second.

5 INTERFACE TO DAQ, GDS AND EPICS

The LIGO DAQ and GDS systems are interfaced via reflective memory. One of the functions of the vertex input crate is to read and write data from and to this reflective memory. Using the DAQ and GDS, an operator can look at data from the system, measure transfer functions, plot power spectra and input diagnostic test signals in real time. A more complete description of the functions and operation of the DAQ and GDS systems are described in reference [2].

Routine operator interface is via EPICS.[5] Through EPICS, the operator can monitor the system, change matrices, gains and offsets, engage filters and invert servo polarity. A Motorola MVME 162 CPU is located in each of the VME crates listed in Table 2. This CPU processes the database records, state code, etc. for the controls located in that crate. Data is communicated between the front end CPU (Pentium III) and the EPICS CPU via shared memory across the VME backplane. This is done in an effort to minimize the burden on the front end CPU and allow it to handle only the time critical front end servo controls.

[5] EPICS- Experimental Physics and Industrial Control System

6 EXPERIENCE TO DATE AND FUTURE PLANS

At this time the digital suspension controls for the 4Km interferometer located in Hanford have been installed and tested and the interferometer is being commissioned. All components for the other two interferometers have been prepared and are ready for installation. To date, no major changes to the basic design are anticipated and the design appears to meet all of the system requirements necessary for the interferometers to reach their full sensitivity.

7 ACKNOWLEDGEMENTS

We thank the entire LIGO team for assistance and support. This work was supported by National Science Foundation Grant PHY-920038.

8 REFERENCES

[1] J. Heefner, 'Large and Small Optics Suspension Electronics Final Design', Internal LIGO Technical Note T980043-00-C, 1998.

[2] R. Bork, D. Barker, 'An Overview of the LIGO Control and Data Acquisition Systems', these proceedings.

[3] J. Heefner, R. Bork, R. Abbott, P. Fritschel, 'The LIGO Interferometer Sensing and Control System', these proceedings.

WEAP060
physics/0111027

THE LIGO INTERFEROMETER SENSING AND CONTROL SYSTEM

J. Heefner, R. Bork, R. Abbott, LIGO Laboratory, California Institute of Technology,
Pasadena, CA 91125, USA

Abstract

The LIGO Interferometer Sensing and Control System (ISC) is a large and highly distributed Multiple Input Multiple Output (MIMO) control system that is used to control the length and alignment degrees of freedom of the interferometers. The 4 kilometer Fabry-Perot cavity lengths are controlled to better than 10^{-13} meters (rms) and the angular degrees of freedom are controlled to better than 10^{-8} radians. This paper will describe the real-time digital servo control systems that have been designed, developed and implemented for the LIGO Length Sensing and Control (LSC) [1] and Alignment Sensing and Control (ASC) [2] systems. In addition, the paper will describe how these controls, along with the suspended optic controls [3], have been integrated into the overall LIGO control and data acquisition system [4].

1 INTRODUCTION

The LIGO interferometers located in Hanford, Washington and Livingston, Louisiana are Michelson laser interferometers enhanced by multiple coupled optical resonators. These coupled optical resonators are 4 kilometers long Fabry-Perot cavities in each arm of the interferometer. The mirrors that form the cavities are suspended from a single loop of wire mounted inside suspension cages that are in turn mounted on seismically isolated optical platforms within the LIGO vacuum system. Control of the optic is achieved using voice coil actuators that act on magnets attached to the surface of each optic. Shadow sensors are used to measure movement and orientation of the optic with respect to the suspension cage. This "local" damping of each optic is not sufficient to keep the optical cavities of the interferometer on resonance in the presence of disturbances. It is the function of the LSC and ASC systems to actively control the length and alignment of the optical cavities with respect to the "on resonance" condition. Figure 1 shows the LSC photodetectors and ASC wavefront sensors (WFS) and Quadrant photodetectors (QPDX, QPDY) in relation to the interferometer optical components.

There are many components that are common to both the LSC and ASC. The length and alignment degrees of freedom are detected and controlled by impressing phase modulated radio frequency sidebands on the light injected into the interferometer. Tuned photodetectors whose outputs are band pass filtered (BPF) and synchronously demodulated are used to detect the various degrees of freedom. The demodulated output is then low pass filtered (LPF), passed through a whitening filter and an anti-alias filter (AA) and on to an analog to digital converter (ADC). A "typical" input signal path is shown in figure 2.

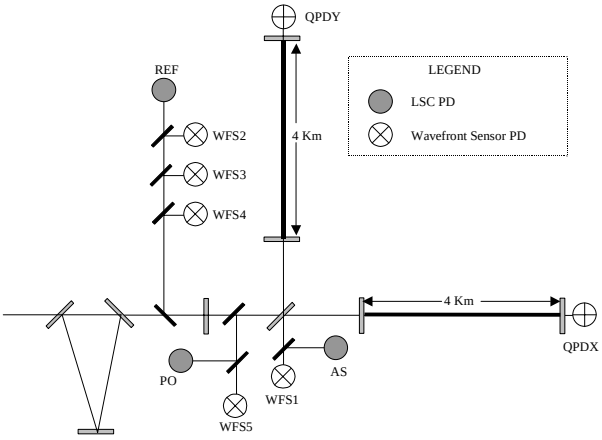


Figure 1: LSC and ASC System Layout

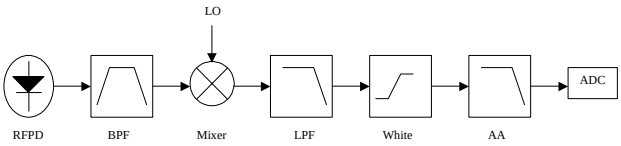


Figure 2: "Typical" Input Signal Path

Outputs from the LSC and ASC systems are handled by the suspension control system as described in [3]. The interface to the suspension controls is via fiber optic reflective memory links. There are three reflective memory loops for the ASC and LSC controls, one for each arm of the interferometer and one for the corner station. The loop to each arm is used to pass data between front end systems in the corner station and each end station. A loop is used for each arm in order to minimize and balance the 13-microsecond delay caused by the speed of light transit time in the 4-kilometer arms. The loop in the corner station is used to pass data between LSC, ASC and suspension front end

CPUs located at the interferometer vertex. In addition there is a reflective memory network for the LIGO data acquisition system. Each node of a particular reflective memory network is connected in a loop using single mode or multi-mode fiber. Data written to a memory location at one node is passed to each node in the loop. In this way data can be shared among nodes.

The analog electronics are very low noise (1-2nV/√Hz input referred) and housed in 6U height Eurocard format crates in the ISC control racks. All ADCs, DACs, processors and other commercial equipment are VME. The timing for the ADCs and DACs is derived from the same GPS (Global Positioning System) based timing system used by the LIGO data acquisition system [4].

2 LENGTH SENSING AND CONTROL

Figure 3 is a block diagram of the length control signal paths from each of the LSC photodetectors to each of the interferometer optics. As can be seen from the figure the length control system is a highly distributed multi-input, multi-output (MIMO) servo control system.

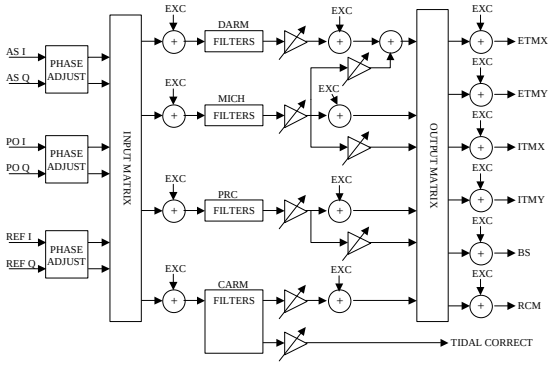


Figure 3: Length Controls showing signal paths from anti-symmetric, pick-off and reflected photodetectors (AS, PO, REF, respectively) to each optic.

The in-phase (I) and quadrature-phase (Q) component of each photodetector are demodulated and passed through a software block that can offset the overall phase of each signal. These phase adjusted signals are passed to the input matrix where each of the degrees of freedom to be controlled are calculated. These degrees of freedom are differential arm length (DARM), michelson length (MICH), power recycling cavity length (PRC) and common arm length (CARM), shown top to bottom, respectively in figure 3. Each degree of freedom is then filtered. The filters shown in figure 3 represent a block of code the can contain up to 10 infinite impulse response (IIR) filters. Each of these filters can contain up to 4 second order sections and each filter can be enabled or bypassed on the fly. Coefficients for each of the filters are read from a modifiable text filter on system startup, but an effort is currently under way to allow the coefficients file to be read and reloaded without restarting the system. Following the filters the signal is passed to an operator adjustable gain stage and then on to the output matrix where the appropriate actuation signals are produced. These actuation signals are passed via reflective memory to the digital suspension controls, which then add the signals to the appropriate degrees of freedom of each optic (ETMX, ETMY, ITMX, ITMY, BS and RCM) as described in [3]. Also shown in figure 3 are the points at which GDS (Global Diagnostic System) excitation signals (EXC) can be added into the signal path. These signals are used by the operators to measure transfer functions, and run other diagnostic tests as described in [4].

Not shown in figure 3 is the connection between the real-time servo calculations and the interferometer lock acquisition code [5]. The lock acquisition code looks at data from the system and determines in real time the filters, gains, etc. required to bring the optical cavities of the interferometer into resonance.

All sampling and servo calculations for the LSC are performed at 16,384Hz, the maximum sample rate for LIGO. This sample rate is necessary because many of the LSC servos have unity gain bandwidths greater than 100Hz. In addition, data collected and processed by a CPU located in the vertex area may be used to drive an optic located in one of the end stations. For these situations, the 13 microseconds required to transit the 4 kilometer arms, must be taken into account when the servos are designed.

3 ALIGNMENT SENSING AND CONTROL

Figure 4 is a block diagram of the alignment control signal paths from each of the ASC wavefront sensor photodetectors (WFS) to each of the interferometer optics. As can be seen from the figure, the ASC, like the LSC, is a highly distributed MIMO servo control system. Not shown in the figure are the connections from each of the QPDs located in the end stations to steering optics located in the interferometer vertex.

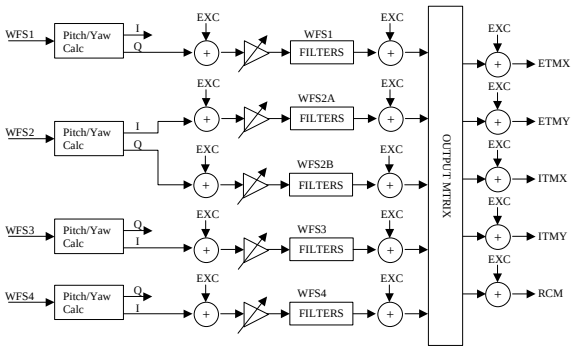


Figure 4: Alignment controls showing signal paths from each wavefront photodetector to each optic

Each of the WFS and QPD photodetectors are a 4 segment detector. Each segment of the detector is processed individually and then used to determine beam pointing (pitch and yaw). One difference between the WFS and the QPD is that the WFS is sensitive to the RF sideband modulation impressed on the interferometer light and the QPD is sensitive to the carrier.

As can be seen in figure 4, the in-phase (I) and quadrature-phase (Q) of each WFS are detected and used to determine an I and Q phase pitch and yaw signal for that WFS. Then, depending on which WFS it is, the I and/or Q pitch and yaw signals are passed to operator adjustable gain stages and then to filters. The filters shown in figure 4 represent blocks of code that can contain up to 10 IIR filters and each of these filters can contain up to 4 second order sections. Each filter can be enabled or bypassed on the fly and like the LSC the coefficients for each of the filters are read from a text file on system startup. Following the filters, signals are passed to an output matrix where the actuation signals for each of the interferometer optics are calculated. These actuation signals are then passed to the digital suspension controls via reflective memory where they are applied to the appropriate degrees of freedom as described in [3].

One difference between the ASC and LSC is that data for the servos is collected in both the end stations and the vertex. Data collected from the QPDs in the end stations is passed to the vertex via the reflective memory network for the respective arm and all of the calculations for the beam pointing servo are done by the CPU in the vertex. Also shown in figure 4 are the points at which GDS excitation signals (EXC) can be added into the signal path. All sampling and servo calculations for the ASC are performed at 2048Hz.

4 INTERFACE TO DAQ, GDS AND EPICS

The LIGO DAQ and GDS systems are interfaced via reflective memory. One of the functions of the vertex input crate is to read and write data from and to this reflective memory. Using the DAQ and GDS an operator can look at data from the system, measure transfer functions, plot power spectra and input diagnostic test signals in real time. A more complete description of the functions and operation of the DAQ and GDS systems are described in reference [4].

Routine operator interface is via EPICS[1]. Through EPICS, the operator can monitor the system, change matrices, gains and offsets, engage filters and invert servo polarity. A Motorola MVME 162 CPU is located in each of the LSC and ASC front end VME crates. This CPU processes the database records, state code, etc. for the controls located in that crate. Data is communicated between the front end CPU (Pentium III) and the EPICS CPU via shared memory across the VME backplane. This is done in an effort to minimize the burden on the front end CPU and allow it to handle only the time critical front end servo controls.

5 ACKNOWLEDGEMENTS

We thank the entire LIGO team for assistance and support. This work was supported by National Science Foundation Grant PHY-920038.

6 REFERENCES

[1] P. Fritschel, R. Bork, G. Gonzalez, N. Mavalvala, D. Ouimette, H. Rong, D. Sigg, M. Zucker, “Readout and Control of a Power-Recycled Interferometric Gravitational-Wave Antenna”, Appl. Opt. 40 (2001) 4988-4998.

[2] P. Fritschel, N. Mavalvala, D. Shoemaker, D. Sigg, M. Zucker, G. Gonzalez, “Alignment of an Interferometric Gravitational-Wave Detector”, Appl. Opt. 37 (1998) 6734-6747.

[3] J. Heefner, R. Bork, ‘The LIGO Suspended Optic Digital Control System’, these proceedings.

[4] R. Bork, D. Barker, ‘An Overview of the LIGO Control and Data Acquisition Systems’, these proceedings.

[5] M. Evans, et. al., “Lock Acquisition of a Gravitational Wave Interferometer”, submitted to Optics Letters.

[1] EPICS- Experimental Physics and Industrial Control System

WEAP061
astro-ph/0111140

VST TELESCOPE DYNAMIC ANALISYS AND POSITION CONTROL ALGORITHMS

P. Schipani, D. Mancini, Osservatorio Astronomico di Capodimonte, Napoli, Italy

Abstract

The VST (VLT Survey Telescope) is a 2.6 m class Alt-Az telescope to be installed on Cerro Paranal in the Atacama desert, Northern Chile, in the European Southern Observatory (ESO) site. The VST is a wide-field imaging facility planned to supply databases for the ESO Very Large Telescope (VLT) science and carry out stand-alone observations in the UltraViolet to Infrared spectral range. So far no telescope has been dedicated entirely to surveys; the VST will be the first survey telescope to start the operation, as a powerful survey facility for the VLT observatory. This paper will focus on the axes motion control system. The dynamic model of the telescope will be analyzed, as well as the effect of the wind disturbance on the telescope performance. Some algorithms for the telescope position control will be briefly discussed.

1 VST TELESCOPE DYNAMIC MODEL

The telescope coupling dynamic is so slow that it is possible to study independently the two main axis behaviors. The coupling of the two axes is usually negligible in a telescope, especially in the most important operating condition, i.e. the tracking phase, in which the axes trajectory is most of the time regular and not interested by strong accelerations.

Both axes structures are modeled by a number of inertias joined by stiffnesses and structural dampings (Figure 1 shows the altitude axis simplified model, the meaning of the symbols is reported in table 1). The axis gear is represented by the motor and teeth contact stiffness and by the damping. The inertia of the motor itself is taken into account, properly scaled by the transmission ratio. The structural data are derived from a Finite Element Analysis of the mechanical structure of the telescope.

The dynamic of the mechanical system can be described by second order differential equations in matrix form as:

$$J\ddot{\Theta} + F\dot{\Theta} + K\Theta = T$$

where J, F, K, T are the inertia, viscous damping, stiffness and torque matrix respectively, and Θ is the angular position vector. Figure 3 shows the open loop response of the altitude axis electromechanical model. Figure 4 shows the open loop transfer function of the azimuth axis. The first notch in the bode gain plots represents the Locked Rotor eigenfrequency (~10 Hz for both axes).

Table 1: Altitude axis structural parameters

Parameter	**Symbol**
Center Piece inertia [kg·m^2]	J2a
M1 (Primary Mirror) inertia [kg·m^2]	J2b
Top Ring Inertia [kg·m^2]	J3a
M2 (Secondary Mirror) box Inertia [kg·m^2]	J3b
M2 Inertia [kg·m^2]	J3c
Motors inertia [kg·m^2]	J1
Motors viscous friction [Nm/(rad/s)]	F1
Viscous friction [Nm/(rad/s)]	F2
Transmission damping [Nm/(rad/s)]	F12
M1 pad damping [Nm/(rad/s)]	F2ab
Structural damping [Nm/(rad/s)]	F23
Top Ring - M2 box damping [Nm/(rad/s)]	F3ab
M2 box - Mirror damping [Nm/(rad/s)]	F3bc
Transmission stiffness [Nm/rad]	K12
M1 pad stiffness [Nm/rad]	K2ab
Structural stiffness [Nm/rad]	K23
Top Ring - M2 box stiffness [Nm/rad]	K3ab
M2 box - Mirror stiffness [Nm/rad]	K3bc
Transmission ratio	R

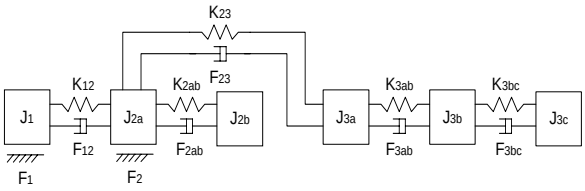


Figure 1: Altitude axis model

2 REQUIREMENTS FOR AXES CONTROL

Both VST azimuth an altitude axes are controlled in a double (position and speed) feedback control loop. The speed and position controllers have to be tuned to guarantee the two most important requirements:

- an extremely low tracking error
- a good disturbance rejection.

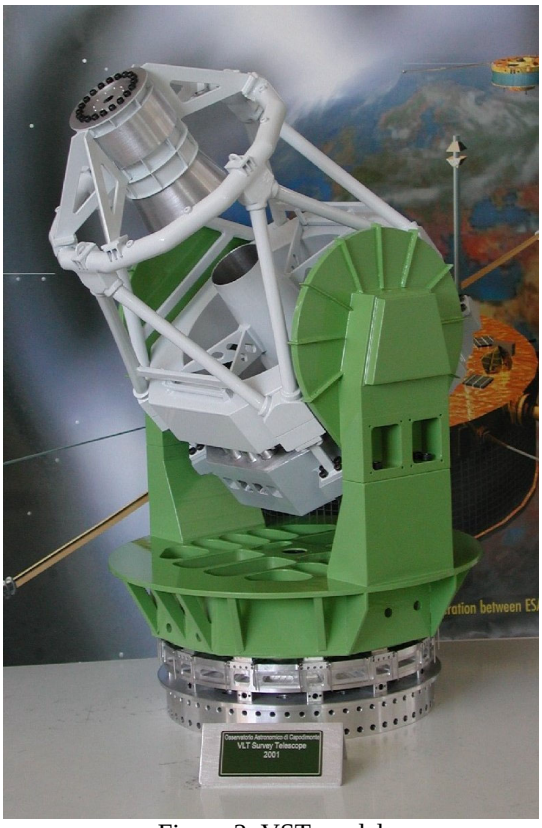


Figure 2: VST model

The absolute RMS tracking error must be as low as possible in order to guarantee the image quality during the observations. Furthermore, the stability robustness of the closed loop must be guaranteed with proper gain and phase margins, to reject structural parameter changes of the controlled system and/or environment modifications.

Particular attention must be paid in the synthesis of the controllers, in order to improve the guide and to assure a proper rejection against the disturbances. The main external disturbance to consider in this analysis is the wind shake. The wind mainly affects the altitude performance, because the altitude axis is exposed to a greater wind torque. The azimuth rotation is much more protected by the co-rotating enclosure. Actually most of the wind disturbance effect comes from the open slit of the enclosure in the observation direction, which affects mainly the altitude axis, and usually the influence of the wind disturbance on azimuth is neglected. The site chosen for VST, Cerro Paranal in the Atacama desert, is sometimes windy. Therefore a wind effect analysis has been carried out taking into account the Chilean site weather statistics.

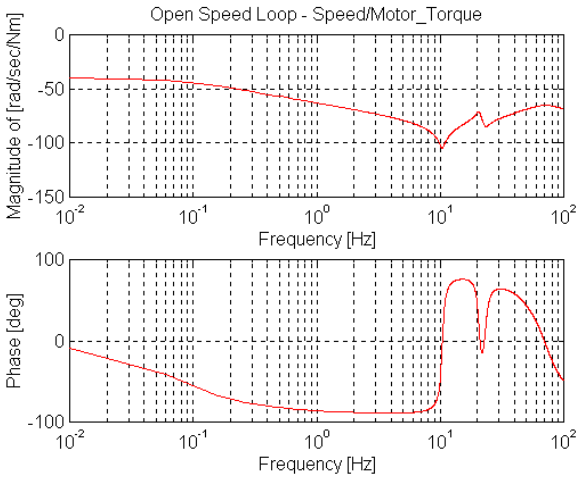


Figure 3: Altitude axis open loop transfer function bode diagram

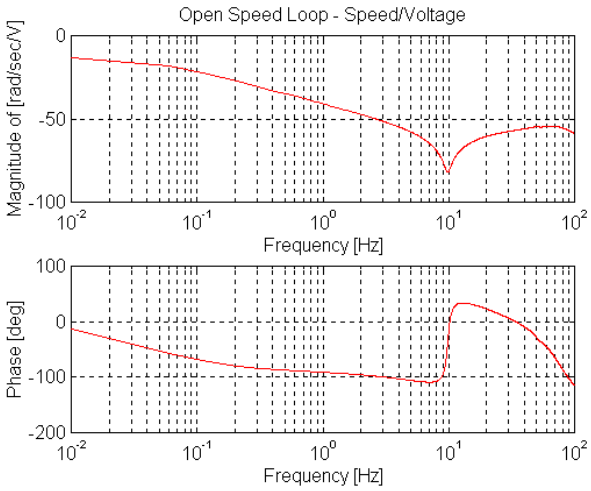


Figure 4: Azimuth axis open loop transfer function bode diagram

3 CONTROLLERS

In this analysis the speed loop controller is a PI. The complexity of the real telescope controller could be increased by adding a series of Notch 2nd order filters in order to attenuate eventual resonant spectral components.

The position controller chosen for this simulation is a standard Proportional+Integral (PI) controller, which ideally guarantees zero error to a ramp input (similar to the usual real observation conditions) after the transient phase. The real controller parameters will not be constant; pointing and tracking phases need different values for the proportional and integral constants. The problem can be circumvented or with a rough switch between different controller structure for small, medium and large errors, or providing the PI controller with anti-windup capability, or better with a variable structure controller (Scali, *et al.*, 1993) in which the

parameters $K_p(e)$, $K_i(e)$ depend on the instantaneous position error value. This last solution has been proven to be very effective with other telescopes with similar dynamics (Mancini, *et al.*, 1997; Mancini, *et al.*, 1998). A detailed analysis of the dependence of the position controller on the tracking error is beyond the scope of the present simulation, which is limited to the "small" error case, i.e. to the tracking phase control. After the tuning of the position control parameters for both altitude and azimuth a -3db bandwidth of about 3 Hz has been obtained.

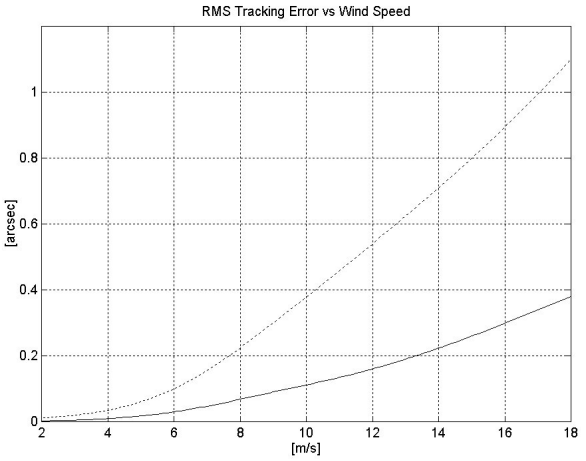


Figure 5: Expected errors vs wind speed, with wind speed reduction factors α=0.98 (upper dotted line) and α=0.63 (lower continuous line)

4 WIND DISTURBANCE EFFECT

The wind disturbance effect on the telescope performance has been studied using the Von Karman spectrum to model the wind speed power spectral density. Some simulations have been carried out at different wind speeds up to 18 m/s, the maximum operational speed in Paranal, in two case studies:

- considering a good wind speed reduction factor inside the enclosure, thanks to a proper setting of the wind-screens, and not observing in the wind direction (α=0.63)
- in a pessimistic condition with almost no wind speed reduction (α=0.98)

According to this simulation results the disturbance effect would increase with wind speed as foreseen; in the more realistic (α=0.63) case, up to 12 m/s the effect could be considered not really performance limiting even in good seeing conditions, while at higher wind speeds in very good seeing conditions a negative effect could be noticed. Wind problems should be limited to a low percentage of telescope usage time; the wind speed is above 12 m/s in about 11% of the time, above 15 m/s only in the 5% of the time.

Table 2: RMS tracking errors due to wind disturbance

V [m/s]	**Time % With Wind Speed > V**	**RMS Tracking Error (α=0.98) [arcsec]**	**RMS Tracking Error (α=0.63) [arcsec]**
3	77	0.02	0.005
6	50	0.10	0.03
9	24	0.30	0.09
12	11	0.54	0.16
15	5	0.80	0.26
18	2	1.10	0.38

5 REFERENCES

[1] Andersen, T. (1995), "The Servo System of the EISCAT Svalbard Antenna", Proceedings of SPIE, 2479, 301

[2] Mancini, D., Brescia, M., Cascone, E., and Schipani, P (1997), "A variable structure control law for telescopes pointing and tracking", Proceedings of SPIE, 3086, 84

[3] Mancini, D., Cascone, E., and Schipani, P. (1998). "Telescope control system stability study using a variable structure controller", Proceedings of SPIE 3351, 341

[4] Mancini, D., Schipani, P. (2000), "Tracking performance of the TNG Telescope", Proceedings of SPIE, 4009, 355

[5] Ravensbergen, M. (1994). "Main axes servo systems of the VLT", Proceedings of SPIE, 2199, 997

[6] Sarazin, M. (1999), "Wind at Paranal: Nighttime Velocity Histogram (1985-1999)", http://www.eso.org

[7] Scali, C., Nardi, G., Landi, A., and Balestrino, A. (1993), "Performance of variable structure PI controllers in presence of uncertainty and saturation nonlinearities", XII IFAC World Congress, Vol. 8, 532, Sidney

WEAP063
physics/0111036

COMPUTATIONAL MODELING IN SUPPORT OF NATIONAL IGNITION FACILITY OPERATIONS

M. J. Shaw, R. A. Sacks, C. A. Haynam, and W. H. Williams
Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Abstract

Numerical simulation of the National Ignition Facility (NIF) laser performance and automated control of laser setup process are crucial to the project's success. These functions will be performed by two closely coupled computer codes: the virtual beamline (VBL) and the laser operations performance model (LPOM).

1 INTRODUCTION

Success on many of the NIF laser's missions depends on obtaining precisely specified energy waveforms from each of the 192 beams over a wide variety of pulse lengths and temporal shapes. The LPOM automates inter-beam power balance, provides parameter checking for equipment protection and diagnostic setup, and supplies post-shot data analysis and archiving services. The LPOM relies on a NIF VBL physics model for fundamental modeling of individual beamline performance.

2 NIF VIRTUAL BEAMLINE

VBL is a natural consolidation and development of a number of previously distinct laser simulation efforts at Lawrence Livermore National Laboratory (LLNL). The PROP92 [1] extraction and propagation code has been the mainstay of design, verification, and component selection for the NIF laser system. The joint LLNL/Commissariat a L'Energie Atomique pumping code AMP [2] has recently been extended to model thermal deposition throughout the laser cavity. Thermal/mechanical/optical response calculations translate this heating into optical aberrations that can be applied either deterministically or stochastically to influence beam intensity statistics and focusability. The optical aberration effects of gas-path turbulence are modeled, and extension to the rest of the laser transport system has begun. Optical imperfections in the many components comprising each NIF beamline also influence beam propagation. The VBL includes these effects either as measured metrology data or as power spectral density-based simulated phase screens. The NIF adaptive optic system is critical for controlling farfield beam characteristics. Simulation played a major role in design and design-certification of this system, and this model is included in the VBL. Understanding of optical damage has progressed from early notions of threshold damage fluence through heuristic memory function treatment of pulse-shape effects [3], to today's emerging picture of stochastically distributed damage initiators coupled with thresholded exponential growth [4] of existing damage spots. This model will be included and updated as understanding develops. Conversion of optical energy from 1053-nm to 351-nm wavelength is required for NIF's scientific mission. Since the efficiency of this conversion is a major factor in NIF energetics, considerable effort has been focused on understanding and modeling this process [5]. The conversion model will be integrated into the VBL in support of the LPOM's laser-setup mission.

All facets of the VBL model are under intense development. One particular advance deals with the facility for deriving the low-power temporal shape required to generate a specified high-power output shape. For speed, the algorithm in PROP92 was based on a single-ray, energetics-only calculation. A new approach was recently implemented that uses multiple rays and an improved iterative algorithm to achieve a more accurate solution while still holding the computation time to roughly 10% of that needed for the diffractive "forward" calculation. The new method improves the solution of the input pulse solver by roughly an order of magnitude.

3 LPOM

The LPOM will be one of the NIF Integrated Computer Control System (ICCS) high-level software supervisors. The primary role of the LPOM is to automate the setup of the 192 individual NIF laser beams. Figure 1 illustrates LPOM's interaction with the laser system. LPOM will maintain a current model that includes the optical paths, configurations, and frequency conversion characteristics for each beam, as well as a database of diagnostic measurements, laser energy, and power at various locations along the beamline.

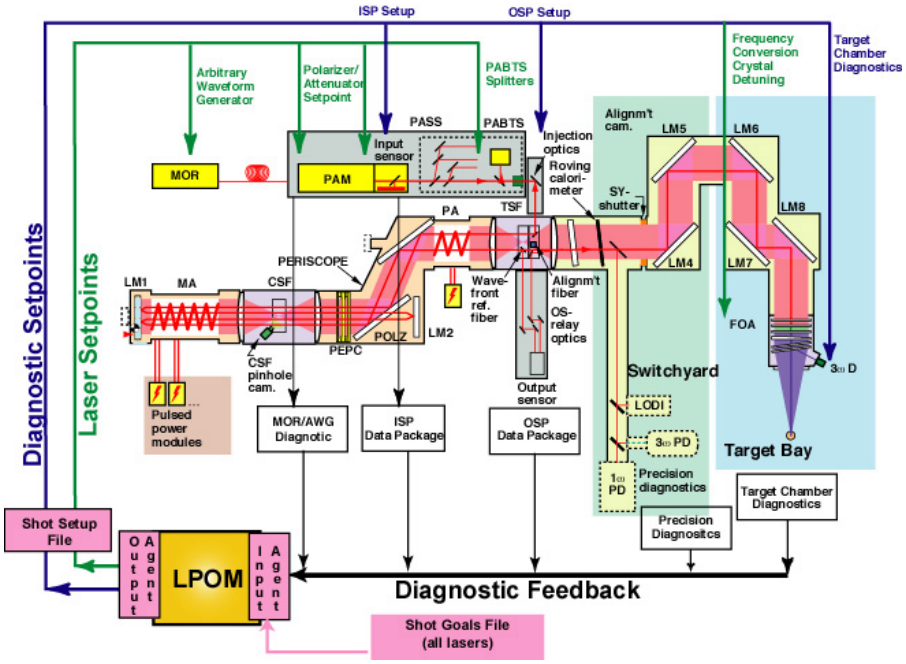


Figure 1: Laser equipment set points determined by LPOM and the diagnostic signals used for feedback. The LPOM communicates with the front-end system through a communication agent with the ICCS.

From these, it will estimate the amplifier gains and optical losses. These parameters will be used in VBL to calculate the laser equipment set points required to produce the requested output laser pulse. These set points include the initial energy and pulse shape produced in the Master Oscillator Room (MOR) and various attenuation settings for waveplates located in the preamplifier module (PAM) and the Preamplifier Beam Transport System (PABTS). LPOM will also predict the power and energy at each of the laser diagnostic locations to aid in setting up the diagnostics system (attenuation levels, temporal range, etc.) in such a way as to optimize the clarity of the signal. Post-shot, this data is fed back to the LPOM in an automated closed-loop fashion to provide updates to the predictive model.

The LPOM Equipment Protection Module has two components. The first module, the Setup Assessment Code, checks the calculated system setup before the ICCS Laser Supervisory System implements it. In this role, LPOM will compare its calculated setpoint values against allowable ranges provided by offline VBL calculations that evaluate the danger of excessive optics damage. The second module, the Setup Verification Module, evaluates actual pulses generated and measured during a series of low-energy (~1 J) shots prior to the countdown of a full-system shot. Its role is to verify that these low-energy pulses match the expected pulse shape and energy. Both modules serve as administrative controls for equipment protection; i.e., LPOM will alert the NIF Shot Director of the status of the shot with respect to the impact upon damage-susceptible elements.

The third LPOM component is a data analysis and archiving module. This module serves two related roles. First, it provides analytical tools required to measure system performance. These include algorithms to calculate laser power balance, compare predicted laser pulses to those measured, and calculate laser system trends. Results from this module are used both for post-run evaluation and for input to future LPOM calculations. Second, it provides archiving of LPOM-generated data, user access to LPOM and its related archives, and graphical tools for trending and data reporting.

4 CONCEPTUAL DESIGN

A conceptual design of the LPOM system has been completed. Because of time constraints, it will be implemented on a parallel architecture, with a single processor devoted to each beamline. A master CPU will control the communication between LPOM and ICCS, coordinate laser setup and equipment protection calculations, and provide graphical user interfaces (GUIs) and data analysis tools to users.

Over the past few months, a prototype or emulation of the LPOM laser setup module has been developed. The prototype is focused on the setup of NIF's main amplifier section and was designed to provide proof of principle of the laser setup strategy, to test parameter optimization schemes, and to develop GUIs. A PROP92 calculation with a prescribed set of amplifier gains and optical losses is used as a surrogate laser

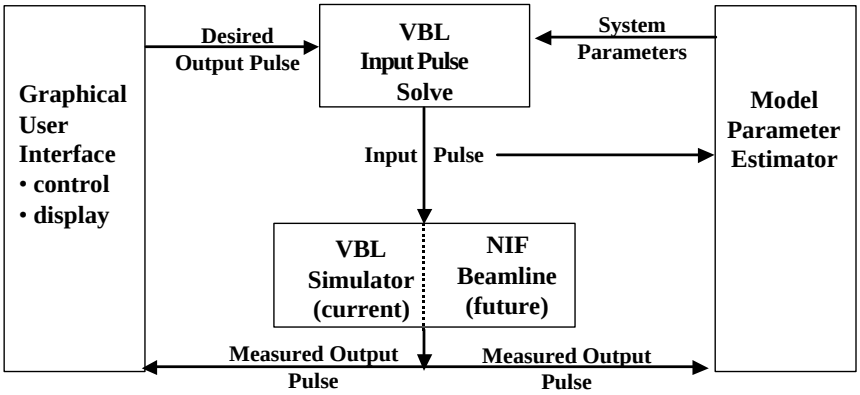


Figure 2: Conceptual design of the LPOM shot setup emulator system. The model parameter estimator consists of VMCON, using a simplified C++ laser model.

shot. A parameter optimizer uses simulated input and output laser diagnostic data (including realistic amounts of systematic and stochastic noise) to estimate a set of gains and losses that best matches the simulated data. This set of parameters, along with the Input Pulse Solver, is used to generate the input pulse shapes and energies required for subsequent shots. The emulator package is diagrammed in Figure 2.

A nonlinear parameter optimization package determines the amplifier gains and optical losses of the laser as a best fit to archived diagnostic data. The parameter adjustment scheme is required to accurately track variations in the system, without introducing appreciable uncertainties in the laser output. A series of tests are under way to determine the most reliable operation strategy. In the immediate future, the emulator will be extended to include the entire laser system and then to include multiple laser beamlines.

5 SUMMARY

The LPOM will be an integral part of the NIF laser's supervisory control software, providing setup, equipment protection, analysis, and archiving services. It will rely on the VBL for physics modeling of the beamline performance and damage risk.

This work performed under the auspices of the U.S. DOE by LLNL under contract No. W-7405-Eng-48.

REFERENCES

[1] R. A. Sacks et al., *1996 ICF Annual Report*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-LR-105821-96. (1996), pp. 207–213.

[2] K. S. Jancaitis et al., "A 3-dimensional ray-trace model for predicting the performance of flashlamp-pumped laser amplifiers," Proceedings of the Second Annual International Conference on Solid State Lasers for Application to Inertial Confinement Fusion, Paris, France, October 20–25, 1996.

[3] J. B. Trenholme, "Damage from Pulses with Arbitrary Temporal Shapes," Lawrence Livermore National Laboratory, Livermore, CA, memo LST-LMO-94-001 (Ver. 2) L-18179-2 (1995).

[4] C. Widmayer, M. Runkel, and M. Feit, "The Effect of LAT Beam Shape, Energy, and Pointing Fluctuations on Damage Initiation Experiments," Lawrence Livermore National Laboratory, Livermore, CA, memo NIF-0064364, April 2001.

[5] J. M. Auerbach et al., *1996 ICF Annual Report*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-LR-105821-96. (1996), pp. 199–206.

WEAP066
cs.DB/0111004

THE RELATIONAL DATABASE ASPECTS OF ARGONNE'S ATLAS CONTROL SYSTEM

D. E. R. Quock, F. H. Munson, K. J. Eder*, S. L. Dean*, ANL, Argonne, IL 60439, USA

Abstract

Argonne's ATLAS (Argonne Tandem-Linac Accelerator System) control system comprises two separate database concepts. The first is the distributed real-time database structure provided by the commercial product Vsystem [1]. The second is a more static relational database archiving system designed by ATLAS personnel using Oracle Rdb [2] and Corel Paradox [3] software. The configuration of the ATLAS facility has presented a unique opportunity to construct a control system relational database that is capable of storing and retrieving complete archived tune-up configurations for the entire accelerator. This capability has been a major factor in allowing the facility to adhere to a rigorous operating schedule. Most recently, a Web-based operator interface to the control system's Oracle Rdb database has been installed. This paper explains the history of the ATLAS database systems, how they interact with each other, the design of the new Web-based operator interface, and future plans.

1 ATLAS CONTROL SYSTEM

ATLAS is a linear accelerator system containing 3 separate ion source injectors and 64 superconducting resonators. This facility accelerates low-energy, heavy-ion elements ranging from hydrogen to uranium. These ions are accelerated up to a maximum of 20% the speed of light mostly for the purpose of nuclear physics research. Typical operation at ATLAS is to start a new experiment every three to five days, with beam being delivered to one of three target areas 24 hours a day.

1.1 Real-time Control

Real-time control of the accelerator is achieved by the interface of Vista's Vsystem software running on a Compaq AlphaServer [4] computer to a CAMAC (Computer Automated Measurement And Control) Serial Highway. Vsystem graphical user interface displays are provided on workstations located throughout the accelerator facility for operator real-time control and readback of accelerator devices.

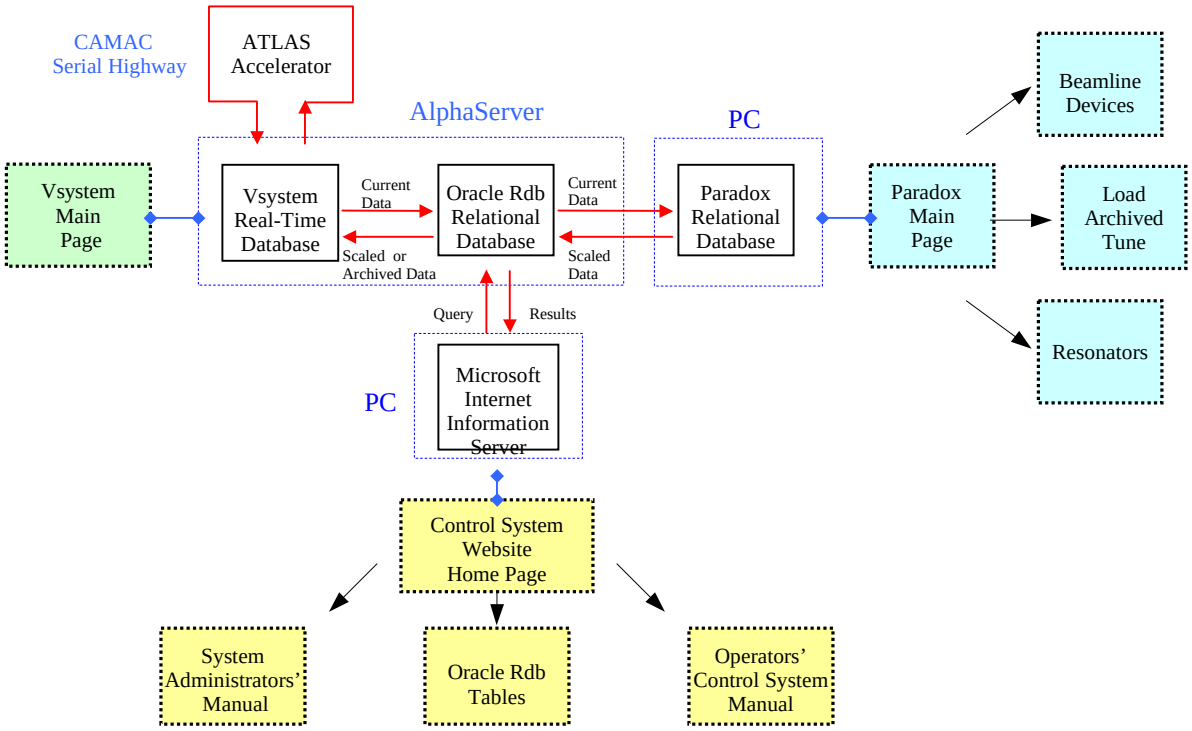


Figure 1: ATLAS Database Management System Software Configuration

* Undergraduate Research Participants

2 ATLAS DATABASE SYSTEMS

2.1 Historical Development

The ATLAS Control System real-time database, Vaccess, is augmented by two relational database management systems: Oracle Rdb and Corel Paradox. A relational database is a collection of tables connected in a series of relationships that, in the case of ATLAS, reflects the organization of accelerator devices and control system processes. The Oracle Rdb relational database was initially developed to store information about the accelerator that changed very seldom. In-house written processes retrieve information from the Oracle Rdb tables for specific functions that they perform as part of the accelerator control system. An example of this is a process that retrieves predefined stepper motor operating positions for a specific slit, foil, or other beamline insertion device. Another in-house written process continuously scans Vsystem's real-time database, and writes critical values to Oracle Rdb database files for data archiving and acceleration configuration restoration purposes. Examples of tables stored in the Oracle Rdb database are:

- Resonators
- Beam Measurement
- Cryogenic Alarms
- CAMAC Crates and Modules

With the later addition of the Paradox tune archiving database application, some of the Oracle Rdb tables began to also serve as an intermediate storage location for the transfer of data from and to Vsystem and Paradox.

2.2 Tune Archiving System

To expedite the startup of new ATLAS experiments, a tune archiving relational database system was developed. This system provides a means for reconfiguring accelerator components by scaling component values of a previously archived experiment according to the charge states, energies and mass requirements of the new experiment. The scaled values are then loaded into the Vsystem real-time database.

At the start of the archiving system development work, Oracle's graphical user interface software was too costly. Consequently, Corel's PC-based Paradox software was selected. Since its inception in 1994, the ATLAS accelerator-tune archiving system has expanded to include several other significant operator functions. The archiving system records accelerator operating parameters on an automated 4-hour schedule, and provides a means for monitoring and analyzing the performance of an ongoing experiment. Details of the Paradox tune archiving database application can be found in a previous paper [5].

3 WEB-BASED USER INTERFACE

3.1 ATLAS Control System Website

The most recent software application addition to the ATLAS Control System is the ATLAS Control System website. This website contains the following three major utilities:

- Interactive query of the Oracle Rdb database
- Operators' Control System Manual
- System Administrators' Manual

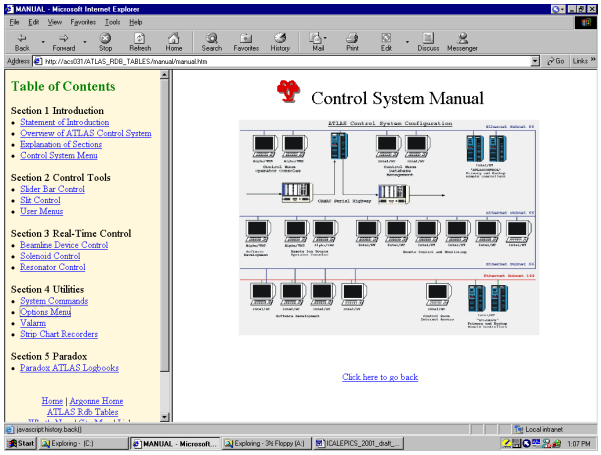


Figure 2: Online Operators' Control System Manual

The Web server software for the ATLAS Control System website is Microsoft IIS (Internet Information Server) [6]. Microsoft IIS is part of the Windows NT 4.0 Option Pack, and is integrated with the Windows NT Server operating system. The control system web pages were developed with HTML, JavaScript, and Microsoft ASP (Active Server Pages) languages. ODBC (Open Database Connectivity) is used to establish communication with the Oracle Rdb database that is installed on a Compaq AlphaServer computer. The Web server PC, AlphaServer computer, and Paradox PC reside on an isolated Ethernet local area network (LAN), and use the TCP/IP protocol for transferring information. Currently, eight PCs are available to ATLAS personnel for accessing the control system website.

3.2 Online Database Access

In the past, control system developers used interactive SQL (Structured Query Language) on the AlphaServer's OpenVMS operating system to view the contents of the Oracle Rdb database. This method is cumbersome, and requires knowledge of the database

structure and SQL language. An Internet browser-based user interface to the Oracle Rdb database was created to provide a quick and simple querying procedure.

3.3 Methodology of Web Browser Query

Standard HTML pages and Microsoft ASP files provide query access to the Oracle Rdb database. The HTML pages use HTML forms that include check boxes, text entry, and radio buttons. These graphical tools provide the interactive interface to the user. The ASP files contain server-side script commands that build an SQL search string based on the user's selections and input. After a user submits a query request from the HTML web page, the ASP script commands execute on the Microsoft IIS Web server. The results of the query are returned in tabular format as an HTML page to the client browser. As shown in Figure 3, the interactive query web pages consist of two frames. The top frame allows filtering and sorting, while the lower frame displays the query results.

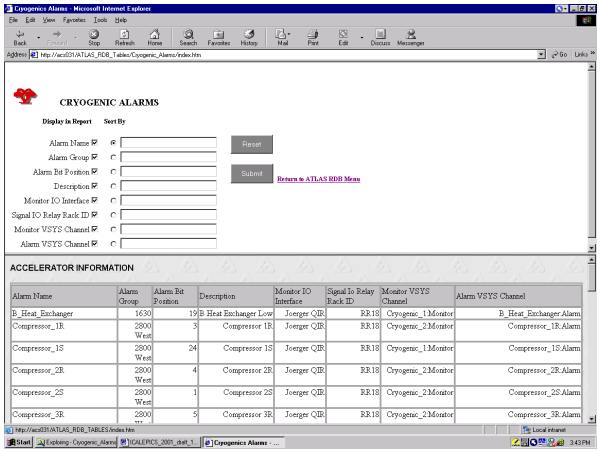


Figure 3: Oracle Rdb Interactive Query Web Page

4 CONCLUSION

The core of the ATLAS Control System database system is Vsystem's Vaccess real-time database. The two relational database applications, built with Oracle Rdb and Corel Paradox, are invaluable accelerator configuration and restoration utilities. The new Web-based user interface to the Oracle Rdb database has become a convenient and frequently used diagnostic tool. The online control system manual is easily accessed from remote PCs located throughout the accelerator facility.

Future plans will shift focus away from the databases themselves to improving the various interfaces to the databases. One improvement is to add a context-sensitive help system to the ATLAS Control System website. Ongoing maintenance of the website requires the update and addition of new control system documentation. It is planned to eventually open the website to TCP/IP networks outside of the ATLAS Control System network, but confined to inside Argonne National Laboratory.

This work is supported by the U.S. Dept. of Energy, Nuclear Physics Div., under contract W-31-109-ENG-38.

REFERENCES

[1] Vista Control Systems, Inc., Los Alamos, NM, USA.
[2] Oracle Corporation, Redwood Shores, CA, USA.
[3] Corel Corporation, Ottawa, Ontario, Canada.
[4] Compaq Computer Corporation, Houston, TX, USA.
[5] F. Munson and D. Quock, "Argonne National Laboratory ATLAS Accelerator Tune Archiving System", 3rd International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC 2000), DESY, Hamburg, Germany, October 9-13, 2000.
[6] Microsoft Corporation, Redmond, WA, USA.

PRODUCTION PHASE DATABASE FOR STAR SILICON STRIP DETECTOR

M. Janik, W. Peryt, S. Radomski, P. Szarwas, Faculty of Physics Warsaw University of Technology, Poland
J. Baudot, Institut de Recherche Subatomique, Strasbourg, France

Abstract

The silicon strip detector (SSD) of STAR is a large silicon detector reaching almost half a million channels and made of thousands of components. In order to help in the long production process and store detailed information of the state of the detector a database system has been set up that contains the main characteristics of all the elements of the SSD. This system includes procedures to enter data from production test benches, to consult, modify and analyse these results as well as to sort or set qualities for tested objects. It is based on a well known database system and usual tools for the WWW interface. We describe here the the architecture of this system, its main features and possibilities and some applications.

1 INTRODUCTION

The silicon strip detector (SSD) [1] is an upgrade of the current STAR experiment setup, adding a fourth layer to the Silicon Vertex Tracker (SVT). It consists of double sided silicon sensors over a surface of around one square meter. To match the high track densities reached at RHIC with Au+Au collisions at $\sqrt{S_{NN}} = 200$ GeV the granularities of the strips is such that the total number of channels reaches 491520.

The detector itself is a barrel of 320 detection modules, each of them made of one silicon sensor (or wafer), twelve readout integrated circuits ALICE 128c, two hybrids that support the readout circuits, and two COSTAR integrated circuits dedicated to the slow control of the detector. The barrel is built out of twenty carbon ladders, each one supporting sixteen modules. To be able to read all the channels in a relatively short time, (below 5 ms), the readout system uses twenty sets of two boards called ADC and connection board. For the sake of a smooth building process all elements were produced in 10% excess, making in total around 7300 objects.

The production is actually spread among several laboratories from research institutes and industries in Europe. For instance, while the individual components of modules are tested in Strasbourg, the module itself is assembled in Paris, then tested again in Strasbourg and eventually fixed on its supporting ladder in Nantes. To cope with all these geographical movements and changes of object status, it was decided to store this information in a database accessible by every actor of the project.

On top of that, to produce a detector of this scale, with the best quality achievable, requires to set qualities for individual constituents and sort among them, the goal being to evaluate the number of good channels and to calibrate them. It is obvious that this task for a half million channels detector can only be done through a computerized system which can store all characteristics of all objects of the SSD.

The database system[1] was then designed with the following main requirements:

- store all characteristics of all objects involved in the building of the SSD in a central place,
- give access to these data to all actors through a web interface,
- allow the geographical and status follow-up of objects,
- display data in simple tables or graphics if needed,
- allow elaborate search through objects.

In this paper we describe first the architecture of the system, then we review all the facilities available to the users and administrators and finally we stress some of the applications where this system was used.

2 TECHNICAL ARCHITECTURE

In essence the database system has to be a distributed system since many laboratories contribute to the project. Nevertheless, one of the main advantages of a database system is to centralize in a unique place all the data. It was decided to build upon a central one-location database server an open system which interacts with this server from different clients, mainly a web application but also LabView and C programs. This architecture is displayed on Fig. 1. A correlated requirement was that it was preferable to use free packages to avoid to pay for licences for a great number of sites.

The MySQL database system was chosen for its simplicity. It is free and provides interfaces for a large number of application languages: C, JAVA, PHP, even ROOT which is a popular tool among the physicists. In addition it allows secure login to the database as well as an automatic procedure to mirror the data on other servers than the central one.

[1] http://wwwstar-sbg.in2p3.fr (guest access possible)

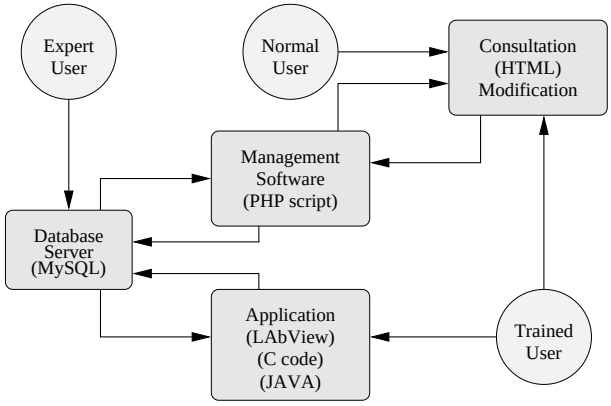


Figure 1: Schematic drawing of the different elements of the database system.

In MySQL data are organized into tables for which we can consider that one entry or line is an object and one column is a property of this object. All the tables needed by the system can be classified in few categories according to their goal. First a few tables are dedicated to administrate the whole project. They contain users and laboratories definitions and their privileges. Then general tables used for all kinds of objects are required to store information like location or status. Two other tables are also useful to all types of object but they fill a specific role. One logs all the changes made in the database, allowing a historical mechanism. The other stores the links between objects (parent-child for objects made of several components). Finally dedicated tables are used for each type of object which properties are not the same.

As stated in the first requirement on this system, tables can contain any kind of data that the production test procedure [2] will generate. Mainly single numerical values are involved here like a current or a voltage. But other kinds of information with a more complex meaning are also used like dead or alive status, graphics in the form of a list of values, position or location (in the form of a string from a predefined enumeration of strings).

From the beginning it was clear that the web will be the main interface, so that the web pages should provide as much facilities as possible. To build HTML pages interacting both with the database server and the user we rely on PHP scripts and JavaScript. A dedicated machine was set up as the central web server for this application. The web pages are mainly dedicated to the consultation of the results but allow also to change some of the properties of objects like location, status and some basic values from tests which are not automatically sent to the database.

The last layer of the system are the programs that allow to enter data directly from the test benches. Most of the tests are automated through LabView Virtual Instruments (VI). There is no direct connection from LabView to MySQL but this difficulty was overcome by using an intermediate C program which is interfaced with MySQL. This little program runs as a daemon on a Linux machine. It receives through TCP/IP socket SQL commands from LabView and then sends them to MySQL. So that only some LabView VIs to build MySQL commands incorporating the data to enter in the database have been created. Alternatively, all LabView VIs also write their data on simple text files with a known format. For every type of file a dedicated C program parses the file and sends the data directly to the correct MySQL tables. This method was actually used at the beginning of the production process.

3 SYSTEM FACILITIES

The very first and readily available pages for consultation after the login step are lists per type of object which show every currently registered item in the database. General information is first displayed in an array with one object per line. From there, the user can access, for a given object, a specific test results window with tables and graphics if some are attached, an editor form to change some of the properties, an assembly window which displays whether the object has some parents or some children and a history window with the log of every change made.

In addition to these simple list displays, it is possible to show on the fly the distribution of any column (properties of the object).

The other main feature is the search engine. On top of the usual search for general properties like name, location or status, a technical search is proposed. It means that for a given type of object the user can build a query to select adjustable ranges for some properties. For example one can look for silicon wafers with a depletion voltage between 25 and 50 volts and a number of dead strips below 3. The result of this search consists of the same kind of list as before with the same facilities. Such a system allows to show detailed distribution for a given type and quality of objects.

Other interesting items proposed in the menu are the summary pages. They review the number of all kinds of objects registered in the database with respect to different properties like location, quality, status. They also provide for each object type the distributions which are particularly important for estimating the quality of the production (distribution of the number of dead strips on silicon sensors for instance).

On these pages one has also access to an always-up-to-date status of the production flow in graphics form. This displays a box for each state/location that the components of the SSD have to go through during the production process. The boxes indicate how many objects are in such a state allowing to judge whether the production flow is in good shape or if there is a critical path.

To be complete, we have to mention that the rest of the menu consists of forms to enter new objects, information on the user privileges and preferences concerning the language to use (the whole software is automatically proposed in English, French and Polish) documentation on the project, version of the software and a 'how to' section.

4 APPLICATIONS

Such a database system is a powerful tool to precisely characterize a production and even individual objects because it offers both the possibility to average values on the whole sample of data and to pinpoint individual parameters for a given object. Let us illustrate this point by considering silicon sensors for which an important property is the depletion voltage at which it starts to operate properly (good signal to noise ratio). This voltage can only be set by viewing a two dimensional histogram like leakage current versus voltage where a clear break in the curve occurs at the depletion. A dedicated JAVA applet, connected to the MySQL server by means of the Java DataBase Connectivity interface, has been built to display this plot along with a cursor that the user can fix at the depletion point. This point is then stored in the database as the depletion voltage. This method was known to work but it was possible to tune it on a large sample insuring a precise determination of the depletion for each detector.

Another example of the benefit of the system is the possibility to set a quality for each object. This quality serves then as the primary characteristic to decide to accept (pay) the object from the producer and to insert it in the SSD. Using the technical search allows actually to check how many objects of a given type will fulfill some criteria, then make some distributions to help in tuning the selection. Once the quality is defined, a simple C code incorporating the selection updates the quality value for each object.

As explained in the introduction, the SSD production phase is still going on. Thus it is clear we have not yet created all the needed applications. The last example worth describing is the way we have chosen which sensors will go in which place in the detector. Each of the 440 detectors produced and tested exhibits different characteristics. Consequently it is highly desirable to distribute wafers by groups of 16 on ladders carefully. For instance, in order to be able to operate one ladder at a given voltage, detectors with similar depletion voltage should be gathered. It is also important to avoid the creation of large dead zones by spreading as much as possible dead strips. To perform such a choice the powerful C++ interpreter analysis framework of the ROOT package was used. A dedicated ROOT macro to read the crucial characteristics of all sensors from the database and then distributes them among ladders while checking the property of ladders (bias voltage, dead strips,...) remains in the specification.

5 CONCLUSION AND OUTLOOKS

The creation of this database system started a little bit earlier than the production phase of the SSD itself. The core packages have been created in a few months by a very restricted team of developers. Then it got enriched by user feedback and new requests. This process of requests from detector testers to the developer is going on for more than a year and is necessary for each new phase the production.

The final application of this system will be to provide the most complete and detailed technical status of the SSD once produced and installed in STAR. It will serve as the primary source of information for the first calibration and condition databases [3] which are required to operate the detector. Meanwhile it will still be available to retrieve the properties of the components during the running life of the SSD.

6 ACKNOWLEDGEMENTS*

The authors want to stress that this development has been made possible thanks to the active participation and interest of all the members of the STAR SSD group whose names follow:
At the Faculty of physics of the Warsaw University of Technology: J. Grabski, A. Kisiel, P. Leszczynski, A. Maliszewski, T. Pawlak, J. Pluta, M. Przewlocki.
At the IReS in Strasbourg : L. Arnold, D. Bonnet, J.P. Coffin, M. Germain, C. Gojac, M. Guedon, B. Hippolyte, C. Kuhn, J.R. Lutz, C. Suire, A. Tarchini.
At Subatech in Nantes : A. Boucham, S. Bouvier, J. Castillo, C. Drancourt, B. Erazmus, L. Gaudichet, G. Guilloux, L. Lakehal-Ayat, F. Lefevre, L. Martin, T. Milletto, C. Le Moal, O. Ravel, C. Renard, G. Renault, L.M. Rigalleau, C. Roy, D. Roy.

7 REFERENCES

[1] The STAR SSD Collaboration, "Proposal for a silicon strip detector for STAR", STAR note SN-0400, June 1999.

[2] J.R. Lutz et al., "Production tests of microstrip detector and electronic front-end modules for the STAR and ALICE trackers", Proceedings of the 5^{th} workshop on electronics for LHC experiments, Krakow, Sept. 2000.

[3] D. Bonnet et al., "Control System of the silicon microstrip layer of the STAR experiment", Proceedings of the ICALEPS'99 conference, Trieste, October 1999.

WEAP070
physics/0111094

AUTOMATED CHECKING AND VISUALIZATION OF INTERLOCKS IN THE ISAC CONTROL SYSTEM

R. Keitel and R. Nussbaumer, TRIUMF, Vancouver, B.C., V6T 2A3, Canada

Abstract

The EPICS based control system of the ISAC radioactive beam facility supervises several sub-systems, which are controlled by PLCs. Most of the devices are protected by non-trivial interlocks, which are implemented with ladder-logic software. Detailed information on interlock state and the individual interlock conditions are accessible for each device at the EPICS operator interface level. With the increasing number of ISAC devices, the interactive generation and maintenance of these displays with standard EPICS tools was too labor-intensive and error-prone. Software was developed, which uses a printout of the PLC program as well as reports from the ISAC device database to a) check the interlock implementation in the PLC against interlock specifications in natural language and b) generate device displays with a graphical representation of interlock state and details of interlock conditions.

1 BACKGROUND

The control system of the ISAC Radioactive Beam Facility at TRIUMF is implemented using the EPICS toolkit [1,2]. Control of the vacuum sub-systems and the sensitive target ion source sub-systems is implemented with Modicon Quantum series PLCs, in order to maintain close to 100% up-time for these systems. The PLCs are peer nodes on the controls Ethernet and are supervised by the EPICS input/output computers (IOCs) using EPICS TCP/IP drivers [1].

Most of the devices in these sub-systems have non-trivial interlocks, which are implemented in the ladder logic of the PLC program. The detailed interlock state of each device can be displayed on the operator console by calling up device control panels from named buttons on synoptic sub-system displays.

Initially, these device control panels were generated interactively using the EPICS *edd* display-editing tool. The compliance of the ladder logic implementation with the interlock specifications was checked by formal walk-throughs of the ladder code. With the number of ISAC devices growing quickly, these methods proved to be quite unsatisfactory and error-prone.

A set of Perl tools was developed, to automate both the interlock checking and interlock visualization processes. It should be noted, that the approach presented here is not a general solution to the problem, but relies on some ladder programming conventions adopted at ISAC and also on some features of the MODSOFT ladder programming software.

2 ISAC INTERLOCKS

2.1 Interlock Specification

The ISAC controls software group realized early in the project, that interlock specifications could be obtained from the equipment specialists much faster, if the controls group provided draft documents for mark-up by the specialist. Interlocks specifications are entered as device instance data in the ISAC relational device database (RDB). The database contains logical expressions for "on" interlocks, "off" interlocks, and "trips". The logical expressions do not use control system variables, but express conditions in natural language, so that they make intuitive sense to the equipment specialists. In order to match the PLC implementation (see below), a maximum number of eight primary conditions are allowed. If this is not enough, placeholder names are used for the primary condition, which are then detailed separately. An example of an interlock specification from the ISAC device database is shown in Fig. 1.

Interlock specification documents are generated for each sub-system as reports from the RDB.

2.2 Interlock Implementation

ISAC uses Modicon PLCs of the Quantum series. These are programmed in Ladder Logic using the Modsoft programming package. PLC devices are

Interlock to turn on:
(DTL:CG1A < HEBT:CG2 OR DTL:CG1A < 50 mTorr)
AND (HEBT:CG2 < 150 mTorr OR rough mode)
Trip:
none

rough mode: HEBT:IV0 closed AND HEBT:TP2 off AND DTL:PV1 open

Figure 1: Interlocks specification from the ISAC device database

interfaced to the supervisory EPICS systems with soft status and command registers. The minimum interface is a 16-bit wide status register of which one bit is an "interlock ok summary" status bit (STATINT) and 8 bits are reserved for detailed interlock status, i.e. "interlock 1 bad" (BAD1), "interlock 2 bad" (BAD2), etc. As a convention, the ladder implementation of the interlock ok summary uses only the BAD1.. BAD8 bits. In addition, the device "on" and "off" status bits (STATON, STATOFF) are used to distinguish "on" conditions, "off" conditions, and "trip" conditions.

Figure 2 shows a screen capture from the Modsoft ladder editor displaying the ladder logic implementation of the specification from Fig. 1.

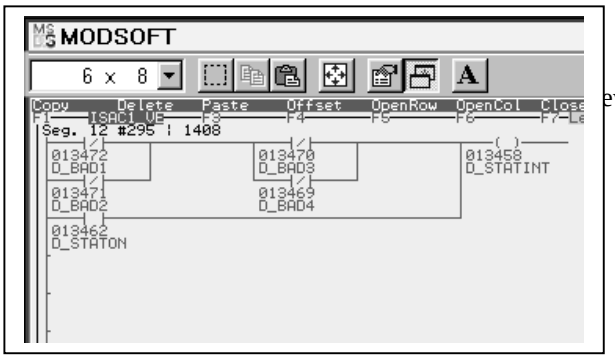


Figure 2: Ladder logic implementation of the interlock summary specification from Fig. 1, containing on "on" interlocks

2.3 Interlock Checking

The ISAC PLC programs are implemented using the visual ladder editor of the Modsoft programming package. Although the binary program format is proprietary, the Modsoft programming package can print the program content to a file using (pretty ugly-looking) ASCII character graphics.

In order to automate the checking of the interlock specification against the PLC program implementation

- Code was added to the ISAC device RDB application to report the interlock specification to an ASCII meta-file. For simplicity, any interlocks more complex than a series of ANDed nodes consisting of ORed primitive conditions were flagged as complex and referred to manual checking.
- A Perl tool was developed which analyses the PLC ladder program. The tool generates an ASCII meta-file in the same format produced by the RDB and reports all discrepancies between the ladder program and the interlock specification.

With this - admittedly quite simplistic - approach, the interlocks of 97% of the ISAC vacuum and ion source system devices could be verified to be correctly implemented. The remaining devices are checked by code inspection. The interaction of the different tools is shown in Figure 3.

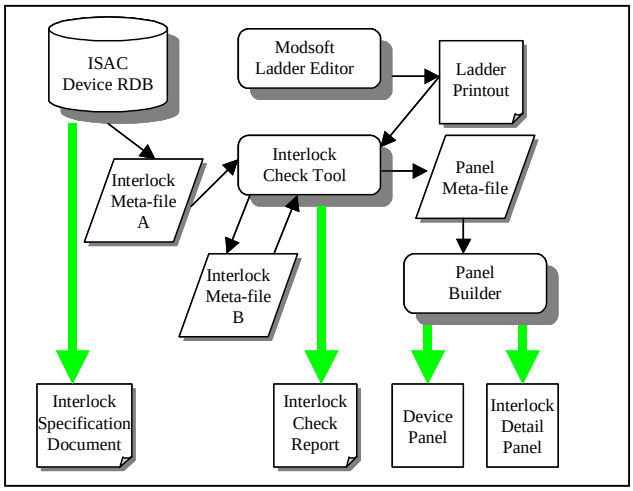


Figure 3: Interaction of RDB and tools

2.4 Interlock Visualization

The presentation of the device interlocks to the ISAC operators evolved in three stages.

a) Interactive generation: Initially, panels were created with the EPICS edd display editor from parameterized display pages, using cut-and-paste. Then the logic was drawn and the interlock condition texts were added according to the interlock specification. This stage established the general look and feel of the device control panels. It was decided to display the logic with flow lines like in a ladder program, changing the colour of the condition texts depending on the ok/bad state of the interlocks. In order to make better use of screen real estate, the flow lines were flipped to run top-to-bottom on the panels instead of the left-to-right direction in the ladder program.

b) Automatic generation from the interlock specification: Device panel section templates were generated interactively with edd and exported to .adl (ASCII) files. From the EPICS device RDB, ASCII meta-files were generated with the interlock information (see 2.3). A Perl script combined these meta-files and the section templates into complete device panel .adl files. The .adl files were converted with EPICS utilities into binary .dl files for the dm display manager. The meta-files could express only limited interlock complexity, but 97% of the ISAC vacuum device panels could be produced this way and have been in production use for the last 15 months. A few confusing situations arose when RDB specifications were out of synch with the ladder

program and it was decided to move on to the next stage.

c) *Automatic generation from the ladder program:* The interlock checking tool (see previous paragraph) had progressed at this stage to the point that it could easily be extended to generate meta-files that capture all the information in the interlock summary logic. Perl modules were written, which use these meta-files and generate device panel .adl files for all ISAC vacuum and ion source device classes. Also generated are detail panels for interlocks, which are only summarized on the device panel. This approach guarantees now absolute synchronization between the PLC ladder program and the EPICS device panels. "Hyperlink" buttons were added to each interlock condition, allowing quick system-wide call-up of device panels for offending devices. Figure 4 shows an automatically generated device panel displaying the interlocks shown in Figure 2, an automatically generated interlock detail panel, callable from the "rough mode" hyperlink in the device panel, is shown in Figure 5.

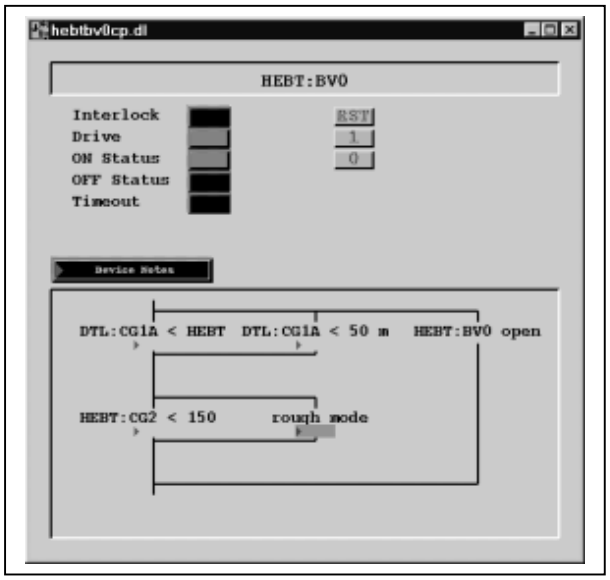


Figure 4: Automatically generated device control panels showing the interlocks of Fig. 1

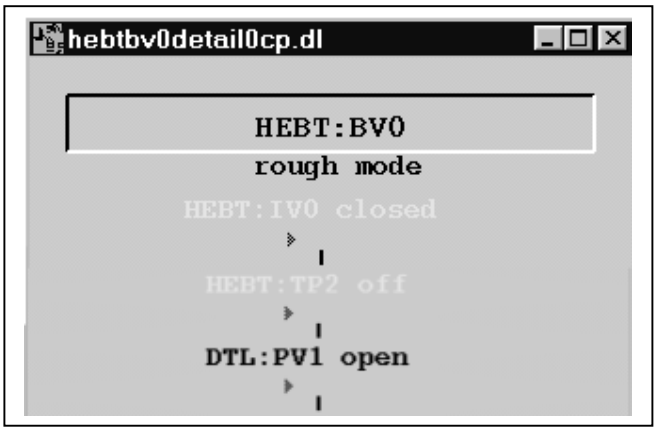


Figure 5: Automatically generated interlock detail panel

3 SUMMARY

The automation of the interlock checking and interlock visualization improved the quality of the ISAC control system considerably. In addition it saves the controls group members from spending a lot of time on repetitive and error prone tasks. It remains to extend the automatic panel generation to the beam optics and beam diagnostics sub-systems. This should be less challenging as the interlocks are usually simple and all pertaining information is contained in the EPICS databases.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the help of M. Leross who undertook the arduous task of checking the PLC interlocks, thereby finding the bugs in our utilities.

REFERENCES

[1] R. Keitel et al., 'Control System Prototype for the ISAC Radioactive Beam Facility at TRIUMF', ICALEPCS97, Beijing, November 1997, p. 372

[2] R. Keitel, et al., 'Design and Commissioning of the ISAC Control System at TRIUMF', ICALEPCS99, Trieste, October 1999, p. 674

WEAP071
physics/0111048

GENERATING EPICS IOC DATABASES FROM A RELATIONAL DATABASE – A DIFFERENT APPROACH

R. Keitel, TRIUMF, Vancouver, B.C., V6T 2A3, Canada

Abstract

The EPICS based control system of the ISAC radioactive beam facility uses the CapFast schematic editor to construct the IOC function-block databases. This allows a self-documenting graphical representation of the IOC software using a hierarchical, object-like software structure with reusable components. On the other hand, the control system is quarterbacked by a relational database, which contains all device information. Using database reports and Perl scripts, a device instantiation method was developed which generates top-level schematic files in CapFast format. This method exploits the advantages of device data entry and reporting using a relational database system while maintaining the graphical representation of the IOC database.

1 INTRODUCTION

One of the powerful features of the EPICS control system toolkit is the method with which the software functionality of the front-end Input-Output Computer (IOC) is generated. Instead of writing conventional code, a run-time database is configured from standard function blocks.

In "EPICS-speak", these function blocks are called "records". The collection of all records in an IOC is called "database". Although these terms are badly chosen, because they suggest the passive behaviour of a relational database, they will be used in this paper in the forms "EPICS record" and "IOC database" or "runtime database" to distinguish them from their relational counterparts.

2 EPICS DATABASE GENERATION

Any IOC database generation starts with assigning field values to EPICS records with the help of an IOC database configuration tool. Field values include hardware addresses, execution links to other EPICS records, and execution information such as membership in periodic scan groups.

Completely configured runtime databases are loaded into the IOC at start-up. The EPICS runtime system, iocCore, establishes the required data structures and resolves the links between EPICS records within the same and between different IOCs.

As the IOC database files are ASCII files, a text editor is in principle sufficient to configure simple databases. More sophisticated, interactive tools add convenience to this process. With increasing complexity of a system, manual maintenance of the IOC database degenerates from cumbersome to error-prone to impossible. It is obvious to try and harness the power of a relational database system (RDB) for this process.

Different approaches have been taken, depending on the design of the underlying IOC database. If the IOC database consists of only a few EPICS records per device, it is possible to generate the IOC database files directly as reports from the RDB. If the device functionality expressed by EPICS records is more complex, EPICS database *templates* can be generated manually with a database configuration tool and instance parameters from an RDB can be inserted by some merging tool [1,2], or the whole process can be done by a more sophisticated RDB application [3].

3 EPICS DATABASES AT ISAC

For the ISAC project, the CapFast[1] schematics editor was chosen as the IOC database configuration tool. It provides a graphical representation of the EPICS records and their interactions and – most importantly – allows the implementation of a schematics hierarchy. This is done by introducing "symbols", each of which represents, but hides, the detailed functionality of a corresponding schematic. The symbol contains instance macros and hierarchical connections. Symbol instances are then used at the next higher level of the schematic hierarchy. Top-level schematics are translated into IOC databases using standard CapFast/EPICS tools. During this translation process, the schematic hierarchy is flattened.

Using CapFast, two types of IOC databases are configured at ISAC:

a) Sub-system databases which contain all device instantiations
b) "Other" IOC databases, which contain no device instantiations.

This paper is only concerned with sub-system databases.

[1] CapFast is a trademark of Phase Three Logic, Beaverton, Oregon

3.1 Hierarchy

ISAC devices are implemented in an "object-like" way, using the hierarchical capabilities of the CapFast tool. Device behaviour, which is common to more than one device class, is implemented in "component" schematics. The components are abstracted and represented by component symbols. This leads to the following schematic hierarchy in the ISAC control system:

- Sub-system schematics – contain device instantiations (device symbols). They are not abstracted and are translated into an IOC database which is loaded into an IOC.
- Device schematics – contain component symbols and may contain EPICS record symbols. They are abstracted by a device symbol.
- Component schematics – they may contain other component symbols and may contain EPICS record symbols. They are abstracted by a component symbol.
- Primitive component schematics – they contain only EPICS record symbols. They are abstracted by a component symbol.

Initially at ISAC, all sub-system schematics were configured interactively.

4 AUTOMATIC EPICS DATABASE GENERATION

In moving to automatically generated IOC databases, it was deemed very desirable to maintain the graphic representation of the sub-system databases, so that all run-time databases, which are loaded into IOCs, can be visually inspected the same way.

4.1 ISAC Relational Device Database

All ISAC device information is maintained in a relational database. For historical reasons, this device database is implemented using the Paradox RDB system. The device database is organized around the concept of a "logical device", which represents all "physical devices" of an operational unit. Examples of logical devices are:

- A magnet, consisting of the physical devices coils, power supply, water switch, and temperature switch.
- A gate valve, consisting of the physical devices solenoid valve and limit switches.

Each logical device is a member of one and only one functionality class. All members of a functionality class share the same behaviour and are distinct by their instance parameters, such as device name, hardware type, hardware addresses, unit conversions, etc. Each functionality class is represented by one and only one CapFast device schematic. Relevant device symbol parameters, such as instance macros and hierarchical port names, are automatically imported into the RDB from the CapFast device symbol files. Fig. 1 shows an entity relationship diagram of the relevant part of the ISAC device RDB.

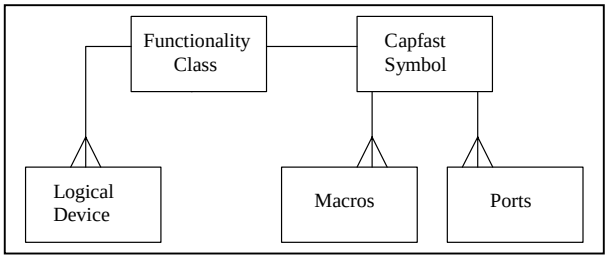


Figure 1: Partial entity relationship diagram of the ISAC device RDB

4.2 Automatic EPICS Database Generation

For the automatic generation of CapFast sub-system schematics, a tool *schgen* was implemented in Perl. This was greatly facilitated by the fact that both CAPFAST schematic and symbol files are ASCII text files, which can be easily interpreted and re-engineered. The *schgen* tool uses device information files, generated by the RDB, which contain all necessary device instance information such as device names, CapFast device symbol names, device instance macros and device instance hardware addresses. Additional information is extracted from CapFast device symbol files. Finally, a sub-system schematic is produced in CapFast format

The ISAC control system developer starts the generation of a sub-system IOC data from the Paradox RDB application main form. (S)he selects the desired ISAC sub-system as well as the subsystem I/O hardware type. Default values for the sub-system schematic name, scan parameters, etc can be overridden at this stage. The RDB then generates the device information files and spawns the schematic generation tool *schgen*, which produces the sub-system CapFast schematic file. An example of an automatically generated sub-system schematic viewed with CapFast is shown in Figure 2. This file is then translated into the sub-system IOC database by the standard CapFast/EPICS tools sch2edif and e2db.

5 SUMMARY

The move from interactive to automatic generation of CapFast sub-system schematics significantly reduced the number of errors in the ISAC runtime databases, mainly due to parameter checking in the underlying

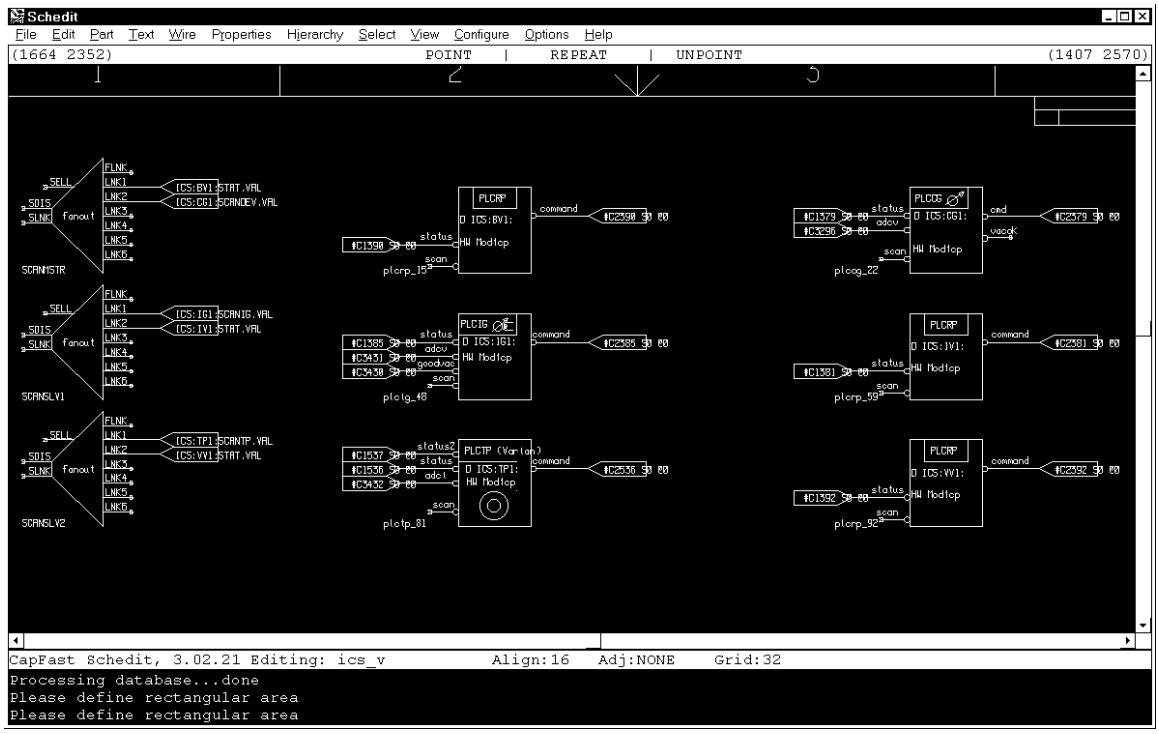


Figure 2: Example of an automatically generated sub-system schematic viewed with CapFast

relational database application. The retaining of a unified, self-documenting graphical representation throughout the control system's IOC databases is well appreciated by all developers. The conversion of older sub-systems from interactive to RDB supported generation was recently completed.

ACKNOWLEDGEMENTS

The author wants to thank J. Richards for helpful discussions about relational database design issues. Special thanks to E. Tikhomolov, who undertook the effort of integrating older sub-systems into the RDB with admirable diligence, and to R. Nussbaumer who provided IOC database comparison tools.

REFERENCES

[1] R. Bakker et al., 'Significance of a Comprehensive Relational Database System for Modern Accelerator Controls', ICALEPCS97, Beijing, November 1997, p. 239

[2] M. Kaji et al., 'Relational Database System in the KEKB Accelerator Control System', APAC 98, Tsukuba, March 1998, p.501

[3] A. Loukyantsev et al., 'The use of Oracle for Development and Maintenance of EPICS Databases', ICALEPCS99, Trieste, October 1999, p 460

WEAP072
physics/0111146

AN INTEGRATED ENTERPRISE ACCELERATOR DATABASE FOR THE SLC CONTROL SYSTEM

T. Lahey, J. Rock, R. Sass, H. Shoaee, K. Underwood
SLAC, Stanford, CA 94309, USA

Abstract

Since its inception in the early 1980's, the SLC Control System has been driven by a highly structured memory-resident real-time database. While efficient, its rigid structure and file-based sources makes it difficult to maintain and extract relevant information. The goal of transforming the sources for this database into a relational form is to enable it to be part of a Control System Enterprise Database that is an integrated central repository for SLC accelerator device and Control System data with links to other associated databases.

We have taken the concepts developed for the NLC Enterprise Database and used them to create and load a relational model of the online SLC Control System database. This database contains data and structure to allow querying and reporting on beamline devices, their associations and parameters. In the future this will be extended to allow generation of EPICS and SLC database files, setup of applications and links to other databases such as accelerator maintenance, archive data, financial and personnel records, cabling information, documentation etc. The database is implemented using Oracle 8i. In the short term it will be updated daily in batch from the online SLC database. In the longer term, it will serve as the primary source for Control System static data, an R&D platform for the NLC, and contribute to SLC Control System operations.

1 DESIGN DECISIONS

The decision to stay with Oracle was made easily and early: it is in widespread use by other laboratories; there was substantial in-house expertise; the relational model is well known and understood; experience with OO databases has produced ambiguous results. These factors eliminated the possibility of using another relational or OO database.

Selection of the relational methods used to represent device attributes and their relationships was a more difficult issue [1]. To date, each laboratory has developed their own schema [2] with some, like SNS, building on previous work done at CERN [3]. We had already developed a prototype design for the NLC Enterprise Database [4]. Building on that work, we extended the model to incorporate the device attributes and relationships in the SLC database. In the process, we considered two methods of handling device attributes.

Attributes as named rows: In this model, each attribute value and its name are stored as a row in a relational table. This is the method used by the PEPII device database at SLAC [5]. Its main advantage is flexibility – the task of adding devices and/or attributes is a simple matter of adding data, not changing structure. However, since each attribute is a row, queries tend to retrieve large numbers of rows and attribute tables can become large, adversely affecting performance. In addition, this model does not allow optimal use of database features such as constraints and joins, nor the standard set of database query and reporting tools.

Attributes as table columns: This is the "standard" way to construct a normalized relational database. Each attribute is a column in a table. The advantages of this approach are clarity of structure, maximal usage of relational features (e.g. constraints and joins) and tools, and faster queries returning fewer rows. Of course, adding devices and attributes requires changes in database structure, a disadvantage in a rapidly changing R&D environment.

We decided on a standard relational model but organized the tables into an OO-like class hierarchy. In addition, every "object" in the database has a unique ID and can be linked to any other object. We can use the standard Oracle tools for design and reporting, and most integrity checking and relationships are implemented in the database structure. The hierarchy of tables allows us to define devices in an Object Oriented-like manner and isolate database changes for new attributes and devices. At the top of this hierarchy is a Linkable Objects table. Using this table we can also link devices together in arbitrary ways. However, addition of new devices and attributes still requires database changes, a maintenance issue that we will need to deal with as development continues.

2 DESIGN DETAILS

2.1 SLC Database Characteristics

The SLC database consists of a set of flat structures called primaries. Each primary has a set of related data and pointers or references to other primaries. Special programs are required to traverse links and find relationships. Since there is no language or standard way to traverse these links, it can be difficult to extract interesting data from this complex web of connections. All sources are kept in versioned text files. The SLC database presently contains over 1 million data items, each equivalent to an EPICS channel.

2.2 Main Relational Tables

Figure 1 shows a few of the main relationships in the database. Every "object" in the database has an entry in the Linkable Objects table. This includes all devices and signals. Everything in the database is a subtype of a Linkable Object. The Accelerator Device table inherits from Linkable Objects and is the parent for most of the usual devices. Controller inherits from Accelerator Device and is the parent for all of the control devices in the system, a large subtype being magnets. One subtype of the Magnets table is Quads. Many controllers also have power supplies associated with them.

IOCs, Crates and Modules inherit directly from Accelerator Device. There are one-to-many mappings of IOCs to Crates to Modules to Signals.

Standalone analog and digital signals inherit directly from the Linkable Objects table. Non-physical information like TWISS parameters, polynomials etc. also inherit directly from the Linkable Objects table.

Database views that join data throughout the hierarchy hide some of the complexity from applications or users, allowing them to obtain data without explicitly joining many tables

In the present design, all of the links e.g. magnet to power supply are explicitly implemented in the tables using database constraints. In the near future, we would also connect Linkable Objects into a hierarchy of related devices, thus a magnet, its power supply (now explicitly connected) temperature thermocouple and BPM would all be associated at the Linkable Object level. Because the SLC database only contains some of this associative information, this aspect of the database has not yet been implemented.

Associations between Linkable Objects would be contained in a single table that could be traversed using an sql select statement with a connect-by clause. This makes possible a "parts-explosion" hierarchy of devices. The cost of this approach is maintenance of the links and increased query time.

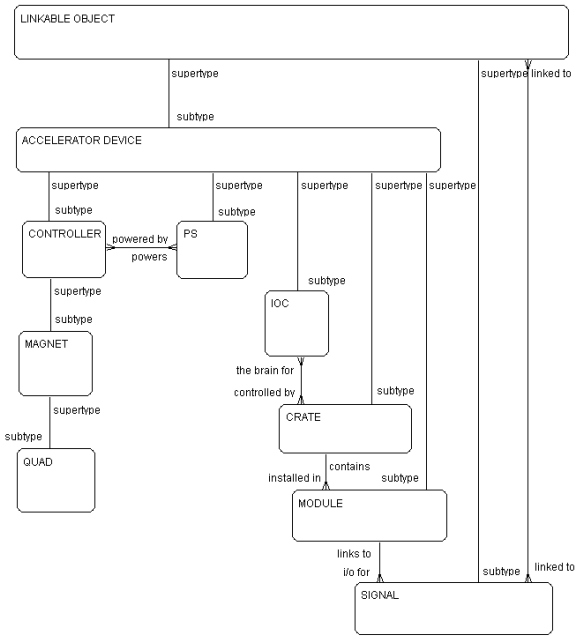


Figure 1 Main Relational Tables

The database presently has about 130 tables. These store most of the information in the SLC database.

3 DATABASE LOADING AND MAINTENANCE

The database is maintained on central SLAC UNIX servers. Updating of the database takes place in three stages:

1. Extract the SLC data to ASCII files on VMS.
2. Load into flat Oracle tables in the UNIX database.
3. Populate the full relational structure in the UNIX/Oracle database from the flat tables.

Updating of the relational structure from the SLC database will be initiated automatically whenever there are changes to the SLC database structure. A single script is executed on VMS that starts a series of actions that run on both VMS and UNIX. Because reloading the entire database structure from scratch can take up to two hours, the scripts detect changes since the last time and just update data that has changed. Of course we also have a reload everything from scratch option as well.

4 CURRENT TASKS

Our first application of the relational database is a replacement for an application called CAMDMP that extracts and reports CAMAC allocation information in the SLC database.

We are in the process of testing the relational model using queries from SLC applications, and will modify and enhance the structure as needed to accommodate these applications.

The Aida project, [6] which is a CORBA based method to access any data in the Control System, uses an extension of the database to access SLC data.

5 FUTURE PLANS

Adding the linking of devices at the level of the Linkable Objects table will provide us with a complete hierarchy of device relationships that is not presently represented in the SLC database.

Other EPICS installations have done various implementations to represent an EPICS database in a relational model. Since EPICS has been loosely integrated into our existing Control System and is used to control several subsystems, we must also integrate it into this database.

Eventually we want to replace the existing text files with this database as the configuration repository for the SLC Control System.

As this database becomes more central to the operation of the SLC Control System, we'll need to develop a set of user interfaces and specialized tools for data retrieval and database maintenance.

6 DESIGN SHARING?

Since most labs are using Oracle and we all have a similar problem to solve, can we share basic relational designs between labs? Do the differences overwhelm the similarities? How and where are schemata published? How modified? Who's in charge?

REFERENCES

[1] B. Franksen, "Update on BESSY RDB", EPICS Collaboration Meeting, Paul Scherrer Institut, May 2001.

[2] N. Yamamoto, "KEKB Static Database Design", EPICS Collaboration Meeting, KEK, May 2000.

[3] S. Sathe, "SNS Database Status", EPICS Collaboration Meeting, Oak Ridge National Laboratory, November 2000.

[4] J. Rock, "Enterprise Database for the NLC", EPICS Collaboration Meeting, Stanford Linear Accelerator Center, May 24-28, 1999.

[5] S. Meyer, "A Components Database Design and Implementation for Accelerators and Detectors", Particle Accelerator Conference, Vancouver, B.C., Canada, 12-16 May 1997.

[6] R. Sass, "Aida - Accelerator Integrated Data Access: An NLC Middleware Concept. These Proceedings.

WEAP074
physics/0111065

A RELATIONAL DATABASE MODEL FOR MANAGING ACCELERATOR CONTROL SYSTEM SOFTWARE AT JEFFERSON LAB*

S. Schaffner, T. Larrieu, JLAB, Newport News, VA 23606, USA

Abstract

The operations software group at the Thomas Jefferson National Accelerator Facility faces a number of challenges common to facilities managing a large body of software developed in-house. Developers include members of the software group, operators, hardware engineers and accelerator physicists. One management problem has been ensuring that all software has an identified maintainer who is still working at the lab. In some cases, locating source code for 'orphaned' software has also proven to be difficult. Other challenges include enforcing minimal standards for versioning and documentation, segregating test software from operational software, encouraging better code reuse, consolidating input/output file storage and management, and tracking software dependencies. This paper will describe a relational database model for tracking the information necessary to solve the problems above. The instantiation of that database model provides the foundation for various productivity- and consistency- enhancing tools for automated (or at least assisted) building, versioning, documenting and installation of software.

1 BACKGROUND

Over a number of years, the Controls Software Group at Jefferson Lab has had good success effecting formal procedures to manage software running on single board processors called input/output controllers (IOCs) that are used for the accelerator slow controls [1]. The procedures include use of the Concurrent Versioning System (CVS) to track software changes, and use custom utility scripts to standardize the management of the software and make policy-adherence less burdensome for software developers. One outcome of the system has been a decrease in accelerator downtime attributable to control software errors. The system simplified tasks such as performing system software upgrades and even Y2K auditing, however, the tools mentioned above were not enough on their own to address issues such as tracking application ownership and interdependencies.

A relational database was integrated into the system to provide the missing functionality. With information stored in the database it became possible to manage the more complex aspects of IOC configuration.

2 CONTROL SYSTEM UNIX ENVIRONMENT

Given the success managing IOC applications, a team was formed to develop the Control System UNIX Environment (CSUE) to similarly improve management of software running on UNIX platforms. One goal of the CSUE team is to develop and implement policies and procedures for managing operational software to ensure that all operational software is registered with an identified maintainer, and that interdependencies are documented. A second goal is to ensure that installation and modification of operational code is performed in a manner that minimizes disruption of control system operations. The system must support the segmented structure of the Jefferson Lab control system which is divided into subdomains (referred to locally as "fiefdoms" [2]), each of which has its own subnet, file and application servers, and control functions, and must typically have its own installed replica of any operationally-required software.

Several other objectives include allowing for consistent management of source code across architectures and operating systems, producing formal data file management policies, and stipulating what documentation is required before an application gets installed. As with the IOC software, scripts are used to implement the policies where ever possible. For example, all applications must be installed via a script; this script checks to make sure that any software listed as a prerequisite is already installed on the target system. Similarly, the application deletion script first checks to make sure no other applications on the target server have a dependency on the application to be deleted.

* Supported by DOE Contract #DE-AC05-84ER40150

3 CSUE DATABASE MODEL

In order to enforce integrity rules as described in the previous section, the utility scripts need access to information about the state of servers and software across all fiefdoms. All of the information necessary for the scripts is stored in an Oracle relational database.

The CSUE database models information in four broad categories: Developers (Staff), Applications, Systems (Servers), and Data (FileIO). The relationships among these categories are shown in Figure 1 below:

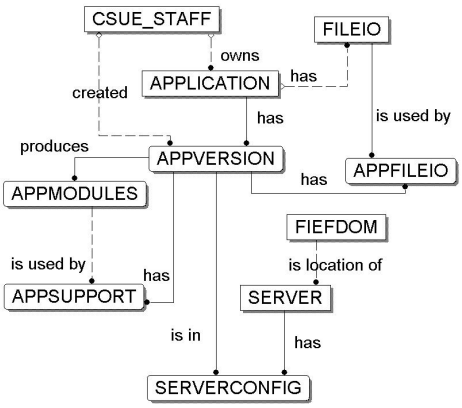


Figure 1. CSUE Database Entity-Relationship Diagram (collapsed). The child end of the relationship is denoted by the dot. Relationship phrases are read from parent to child. Solid lines denote primary key/foreign key relations. Non-identifying relations are shown with dashed lines.

3.1 Staff

Staff information is maintained as a link to an external database that is kept in sync with Human Resource Department information. In this manner up-to-date staff information is always available, be it the most current email address for a developer, or the fact that a staff member has departed employment at Jefferson Lab.

3.2 Applications

In CSUE, applications are sets of source code managed as a unit. Applications may consist of mixtures of code types (e.g., C/C++, Perl, TCL, etc.), both compiled and interpreted. Functionally, an application can be code to perform an operational task or code to be used by other applications (e.g., a C/C++ library, Perl module, Tcl packages). An application may be subdivided into smaller applications, each managed separately.

Classes of applications may be defined where the details of application management differ. CSUE currently supports 2 classes of applications. The first class consists of applications whose source code is managed directly (locally written or customized applications). The second class is licensed or public-domain software (such as the Perl or Tcl interpreters) where the source is not locally modified. Different management policies apply to these types of software, but from the standpoint of the relational database model, the same set of tables can be used to store information about the software.

The Application category in the CSUE database uses several tables. The APPLICATION table stores information that is version-independent. An application owner (taken from the CSUE_STAFF table) is associated with an application. The portion of the path to the base of the source directory is also stored.

Information specific to each version is stored in the APPVERSION table. For example, the path component to the version source relative to the base source path is an attribute of this table. The relationship between the CSUE_STAFF table and the APPVERSION table is used to track the creator of the version since a team member rather than the application owner can create versions.

The products of an application are tracked in the APPMODULE table. In CSUE terms, a module is any product that will be executed or used as support by another application. Examples of products are a script, a header file, a library, or a compiled binary. Since the products of an application can vary from version to version, the APPMODULES table is related to the APPVERSION table rather than directly to the APPLICATION table.

Entire applications or portions of an application can be referenced by other applications. An example is a shared header file for compiled applications or a script application that makes an internal reference to another script from a different application. The APPSUPPORT table tracks the specific modules used by an application version from another application version (called the supporting application). The database makes it easy to determine the interdependencies between programs so that when one program is upgraded it is possible to determine in advance what other programs might be adversely affected.

3.3 Servers

In CSUE terms, a server is a host for executables. A server is a Unix machine, running a version of HP-UX, Linux, or Solaris on PA-RISC, Intel, or Sparc hardware. A server resides in a particular fiefdom [2] and has installed on it a subset of the applications tracked in CSUE. For any server it is possible to

determine when a particular application version was in beta testing, when it was in production, when it became a rollback, and when it became obsolete.

3.4 File I/O

Any files produced by or used by software other than the source and its makefiles, are considered file I/O. Rather than attempt to keep track of all the files produced or used by applications, classes of file I/O are defined (in the FILEIO table) and a fixed set of file classes are defined for each application class. Versioning of file I/O is independent of application versioning because file formats, and very often documentation, change more slowly than the source code. The version of file I/O being used by a particular version of an application is tracked in the APPFILEIO table.

One major file I/O issue is management of data files: where to store data files, how long to retain them in primary storage, whether to delete old files or move them to secondary storage, what to do when a file format changes, etc. are left up to individual software developers or software users to manage. A goal of CSUE is to formalize the management of file I/O. The FileIO portion of the CSUE database provides a central store of information about the location, version, and storage policy for files produced by all applications. This information can be used to automate archival processes for file I/O. The developer is still responsible for managing the files belonging to a certain class, however, in the absence of specific instructions from a developer, all file I/O classes have a default storage/archival policy that will be applied to help the system administration team manage disk usage more effectively.

4 TOOLS

With the database infrastructure in place, emphasis is being placed on the refinement of tools to aid in-house development. Given the large pool of staff writing operational software, scripts, utilities, etc. it is important that capturing information about dependencies and requirements be as automatic as possible. Requiring software writers to register their applications after-the-fact using some foreign interface such as a web-form seemed likely to ensure that such registration would often not occur. So, emulating the methods for developing IOC applications [1], the team decided to develop scripts for CSUE developers to use to manage their applications. While performing useful tasks such as laying out skeleton directories for a new application, or installing a finished package, the tools capture and store critical information in the back-end database.

The tools fall into 3 categories: creation and version management tools, utility tools, and information tools. The tools interact not only with the Oracle database, but also with the file system and with the CVS repository. The tools currently available allow creating applications, specifying modules, adding support, and creating new versions. The installation tools are to be developed next. After that, work will be done to develop tools for managing the file I/O. A web-based interface to browse or search all the documentation belonging to registered applications is also in the works.

5 CURRENT STATUS AND FUTURE PLANS

As of this writing, the database infrastructure has been largely finalized. Nearly a dozen new applications under development have been registered in the database (including the development tools themselves) and an approximately equal number of existing applications have been imported and registered. The next challenge will be to import all of the existing software, scripts, and utilities, numbering in the hundreds, often without identified maintainers, into the CSUE structure. It will require significant effort to identify this software and import it since internal code references to other software and to files will have to be modified to reference standard CSUE locations and paths. In addition, developers will be forced to think about such issues as file management and proper documentation. This may prove to be as difficult as actually developing the database and the tools.

REFERENCES

[1] S. Schaffner, M. H. Bickley, et al. “Tools for Application Management at Jefferson Lab”, ICALEPCS 99, Trieste, Italy Oct. 1999.

[2] K. S. White, M. H. Bickley “Control System Segmentation”, PAC 01, Chicago IL, June 2001.

WEAP076
physics/0111083

RESONANCE CONTROL COOLING SYSTEM FOR A PROTO-TYPE COUPLED CAVITY LINAC[1]

C.A. Treml[2], S.K. Brown, J.D. Bernardin
LANL, Los Alamos NM, 87545, USA

Abstract

Los Alamos National Laboratory (LANL) is designing a coupled cavity linac (CCL) for the Spallation Neutron Source (SNS) being built at Oak Ridge National Laboratory (ORNL). As part of the design process, a proto-type, Hot Model, consisting of two segments of the SNS CCL was manufactured and installed at LANL. An RF source was applied to the Hot Model to determine the success of the cavity design, the effectiveness of the control algorithms and cooling systems, and the validity of the Resonance Control Cooling System (RCCS) model used during the Hot Model design. From a system controls perspective, an overview of the hot model, a description of the RCCS, and the pertinent data from the RF tests will be presented.

1 THE CCL HOT MODEL

The Hot Model is an experimental proto-type of the first two segments of the SNS CCL, and its associated subsystems. These sub systems include water cooling and control, vacuum, RF, and hardware system control. The water cooled components of the CCL are the copper cavities, the short coupling cells between cavities (CCC), and one long coupling cell (LCC) between the two CCL segments. Each of the segments contains eight cavities with their corresponding CCCs.

The desired resonant frequency of the CCL cavities is 805.0 MHz. Under normal operation approximately 70% of the RF Power is dissipated in the CCL cavity walls. For a CCL, electromagnetic field resonant frequency is primarily a function of the geometry of the cavities, side coupling cells, and bridge couplers. As RF power is dissipated in the cavity wall, the copper wall will heat, then expand, and it's resonant frequency will decrease. This frequency shift will be controlled by introducing an opposing frequency shift in the cavity design and a closed loop cavity cooling system.

The CCL cavities have been designed and will be manufactured and pre-tuned to have a resonant frequency of 805.140 MHz with 20•C coolant flowing through the structure surrounding the RF cavity and no RF heating. On the back of the copper structure from which the CCL cavity is machined, there will be four parallel annular paths around the cavity nose (see Fig. 1). These paths will be fed and collected by internal plenums. The internal plenums will be connected to supply and return manifolds that are in turn connected to a water supply, cooling, and control system. Preliminary analysis indicated that for expected operating conditions, i.e. nominal steady RF load, a cooling water flow rate of 2.4 gpm, and an initial water temperature of 20°C, the frequency shift for a high energy (β=0.54) CCL cavity is a approximately -140 kHz and the temperature rise in the water is approximately 3.0°C. These results prompted the design frequency of 805.140 MHz. The water temperature rise and other deviations in frequency due to errors in manufacturing and power fluctuations are compensated for by changing the water temperature via the RCCS.

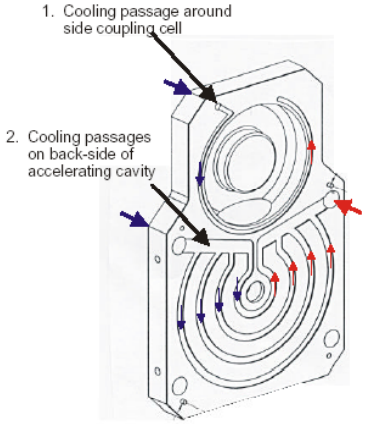


Figure 1: Diagram of the water cooling passages in the CCL RF structure

2 RESONANCE CONTROL

The RCCS removes waste heat from the copper structure around the CCL RF cavities and maintains resonance within these cavities through active temperature control. The importance of maintaining the

[1] Work supported by US Department of Energy Contract W-7405-ENG-36
[2] email: treml@lanl.gov

CCL resonant frequency is that it provides for the efficient acceleration of the proton beam with minimal klystron energy. The RCCS is comprised of multiple, closed-loop water cooling systems and the hardware/control for those systems.

2.1 Hot Model/RCCS Layout

The water cooling system for the CCL consists of four main components; the RF structure, transition plumbing, a water skid, and a facility chilled water source. A generalized schematic for the RCCS configuration is shown in Figure 2. The cooling features of the Hot Model RF structure consist of a three main paths; one for each of the cavity coupling cells, one for each of the cavities, and one for the long coupling cell. The CCC path enters on the side of the coupling cell from the inlet manifold and flows around the circumference of the cell and exits on the side of the outlet manifold. Each of these paths removes approximately 500W/cell of waste heat. Similarly, the cavity path enters then branches into four parallel annular paths on the back side of the RF cavity. This path removes the bulk of the waste heat (~1.87kW/cavity) and is the conduit for the temperature, thus resonance control effort. The water skid is a self-contained unit with all of the necessary plumbing, water treatment hardware, and instrumentation/control features required for delivering water at a desired temperature and constant flow rate to the CCL RF structure [1].

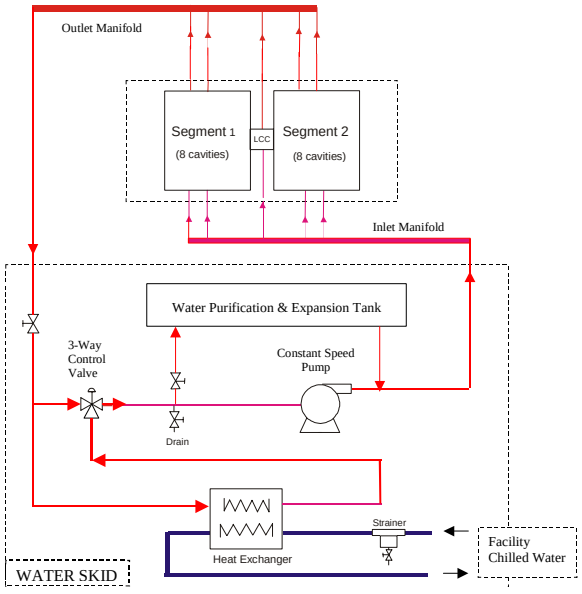


Figure 2: Generalized CCL Cooling System Schematic

2.2 Control Scheme

The RCCS control scheme is to maintain the water temperature in the flow loop supplied to the RF structure such that resonance can be maintained in the RF cavities. The water temperature in the flow loop is manipulated by adjusting the proportion of hot water returning from the CCL that is diverted through the water skid heat exchanger. The mechanism for this manipulation is a "3-Way" control valve, whose control effort is calculated via a multi-mode Proportional, Integral, Derivative (PID) control algorithm. All of the instrumentation/control for the Hot Model was implemented via a PC system running *LabVIEW*® and integrated to the hardware/sensors using *Field Point*®.

For the Hot Model, the multimode PID algorithm consisted of manual, temperature feedback (TFB), and RF error feedback (RFEFB) modes. The manual mode allowed an operator to position the control valve by entering values manually into a *LabVIEW*® screen. Manual mode served as a startup, maintenance, and override feature, rather than a mode for normal operation. The RFEFB mode was intended for normal operation, to control resonance under RF Power fluctuations. The TFB mode was used during the loss of RF power, and to preheat or maintain the cavities temperatures close to their resonant temperatures during RF loss.

The RF error is a measure of how far off the cavity RF is from resonance. The RF error signal is calculated using a moving average of the reflected RF power. Whenever the reflected power is measured at zero, i.e. the cavity is in resonance or there is no power, the RF error signal is 5.9V. Hence, 5.9V is the set point for the PID feedback loop when in RFEFB mode. A dead band of oscillations about the resonance frequency of 805.0 MHz (5.9V) of +/-10kHz (+/-0.3V) is specified for the SNS CCL [2].

In the event of RF power failure or trip, hence, the loss of heat load to the water cooling system, the control mode must be switched from RFEFB to TFB mode. This transition is necessary, since the RF error signal is forced to the set point in the event RF power is lost. With the process variable at the set point, all control effort goes to zero regardless of the temperature of the cavities. In the case where RF power has been lost, the control valve would remain at its current position, and the temperature of the RF cavity would begin to decrease due to the loss of the RF heat load. The motivation for switching to TFB mode is to keep the temperature of the cavities as close as possible to the temperature for which its resonant dimensions are achieved. Hence, there is little time lost returning to resonance once RF power is restored.

While in TFB mode, the desired temperature to use as the process variable would have been the actual cavity temperature; however, it was not possible to place a sensor directly on the cavity wall. Several surface temperature sensors were placed on the outside of the RF structure and water temperature sensors were placed in various locations throughout the water cooling system. Given the sensor information available, the desired process variable for the TFB mode was the water temperature on the transition line leading from the four parallel annular paths on the back side of the RF cavity to the output manifold. This parameter was the most representative of cavity temperature, and had the smallest time delay between it and the cavity temperature changes.

2.3 Results

Overall, the RCCS for the Hot Model worked well. The water skid, water transfer lines, orifice plates, manifolds, and the water cooling pathways distributed water evenly and as expected. Both the TFB and RFEFB RCCS control modes brought the process variable into an acceptable region about the set point with settling times on the order of 3 - 5 minutes.

In particular, the results in Figure 3, show the success of the RFEFB mode. This plot shows the RF error tracking the set point of 5.9V over a series of RF power fluctuations, severe drops, and total losses. The

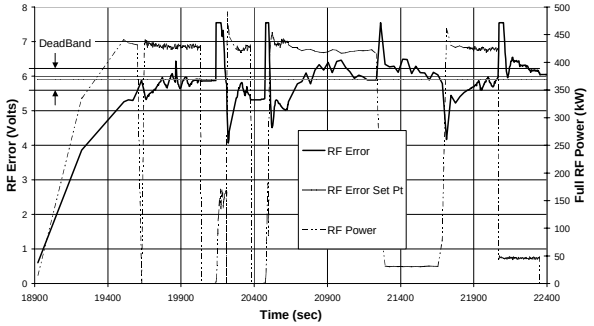


Figure 3: Hot Model Data: RF Error, Set Pt., Dead Band, and RF Power (08/16/01 PM)

RF error tracks inside the dead band within 3 or 4 minutes. A slight drift from the dead band does occur. On analysis of the related data, the cause of this drift was determined to be fluctuations in the facility chilled water. The oscillations of approximately 0.3V about the set point resulted from the coarseness of resolution in the 3-Way control valve. Both the facility chilled water and valve resolution findings will be applied to the SNS CCL.

3 MODEL VALIDATION

As part of the RCCS design/tuning process, the data collected from the Hot Model experiments will be used in a model validation scheme to validate the RCCS simulation used during the design of the RCCS. The motivation is to use the validated RCCS model to aid in tuning the RCCS PID control loops for the SNS project.

The validation technique that will be used is being developed as part of the authors dissertation research. Within a model reference adaptive control (MRAC) framework, the nominal model is considered the plant to be controlled or adapted. The model reference, i.e. the desired output of the model is for a specific mode of operation and is inferred using a Bayesian belief network (BBN) utilizing the relationships between the mode or operating conditions of a desired run and experimental data from runs with similar mode or operating conditions. The control logic is provided by a model parameter updating BBN augmented with cost and decision nodes. The updater BBN utilizes expert knowledge of model behavior and statistical knowledge of model input/output relations. The BBN/MRAC scheme minimizes model parameter uncertainty by reducing the difference between the simulation model outputs and the operational run specific reference model outcomes [3].

REFERENCES

[1] J.D. Bernardin, et al., “Spallation Neutron Source Coupled Cavity Linac Water Cooling and Resonance Control System Preliminary Design Report”, SNS-104040500-DA0001-R00, August 2, 2000.

[2] D. Hayden and C. Martin, “Instrumentation and Controls System for the SNS Hot Model RCCS 1999 Final Report”, Energy Process Engineering, Nov. 1999.

[3] C.A. Treml and R.M. Dolin, “Model Parameter Updating Using an As-Built/As-Is Philosophy and Bayesian Belief Networks”, 16th IMACS World Congress, Lausanne, Switzerland, 21-25 August, 2000, Proceedings.