

Figure 3 The upgraded database

In tests, we found that the I/O speed of the KSC2922/3922 is lower than that of the VCC. To transfer the same number of data when 30 main power supplies are ramping, the minimum interval between two QIOs is 30 ms for the VCC and 44 ms for the 3922. The reason is that the VCC is an intelligent module which can assemble F17 command transfers to CAMAC modules, while the 3922 is a dumb module. The F17 command has to be sent by the VAX computer, so the length of the 3922 packet chain is 1.5 times as long as that of the VCC. As it takes more time to transfer the data, we don't acquire the device status information during the ramping of the main PS to enhance speed. In accordance with the 3922 packet rules, the output data are placed in the packet chains and the readback is mapped onto the old VCC area.

## 5. CONCLUSION

The upgraded BEPC control system was completed in October 1994. The dedicated adapter VCC has been replaced by a commercial product successfully and the real-time response speed and the reliability of the system are improved. In the near future, we will upgrade the DVM acquisition with an analog scanning module.

## ACKNOWLEDGMENT

The author would like to thank Prof. Huiying Luo and Chunhong Wang who gave us a much support.

## REFERENCES

- [1] J. Zhao, X. Geng et al., Proc. of Conf. on High Energy Acc. (1995)
- [2] J. Zhao, Nation Conf.
- [3] X. Geng, Y. Yan, Nation Conf.

# The IHEP Accelerator Utilities Control System

S. Goloborodko, A.Kholodnyi, E.Oustinov, A.Sytin

Institute for High Energy Physics , Protvino, Russia

## Abstract

This paper presents the IHEP Accelerator Utility System ( AUS ) architecture and the hardware and software organization of the Utility Control System ( UCS ). The size and complexity of UNK requires a multi-level multi-processor control architecture. Low level control for utility subsystems is performed by distributed equipment controllers. The next high level of control is carried out by front end computers (FEC) spread around the main accelerator ring and Beam transfer line (BTL) and is interconnected with higher level computers and operator consoles by a control network. A review of the general features is given.

## Introduction

The IHEP Accelerator Utility System (AUS) includes safety equipment for personnel and machinery in the premises of the accelerator complex. The Utility Control System (UCS) is composed of the following sub-systems : electricity distribution network, gas analysis and compressed air, ventilation, fire-fighting and drainage. It is made up of hardware and software designed to automate the control and so increase the effectiveness of the accelerator operations staff and the utility control staff.

The UCS must transmit the necessary information to the dispatchers and to the operators of the respective services, as well as allow them to set, change or monitor utility systems parameters. It is also necessary to ensure the correct order of the actions of systems and operators to guarantee the security of personnel as well as the protection of the accelerator equipment. It is requisite to keep a record of AUS state changes and UCS operators actions.

To perform these functions, UCS must provide for AUS data acquisition, AUS data processing and archiving, the generation of control messages and transmission of these messages to the equipment of the respective control systems, the creation and update of the respective information databases, the presentation of the AUS state information to the operators, the generation of parameter-out-of limits warnings and provide the possibility for manual remote control of the equipment and equipment diagnostics.

## Organization of UCS Hardware.

A peculiarity of utility systems is the unusual diversity in the low-level data acquisition devices (pick-ups) and in the control devices in different sub-systems. Since the AUS sub-systems will be installed in all operational buildings of the accelerator, a hierarchical structure is the obvious one for the UCS hardware ( fig. 1 ).

The data acquisition and control hardware is built in Euromechanics (EM) crates with the MULTIBUS1 (MB1) backplane bus and in an industrial-designed servo-amplifier equipment crate (EC). This configuration (one EM and one EC) provides support for 256 transducer inputs, 32 electrically operated valves and 8 actuator drivers for AC electric motors. All the motors for the compressors, pumps and valves used throughout the utility equipment are interfaced by Digital Servo Amplifier cards (DSA). A PC-16 module with an i8086 microprocessor is used as a crate controller in this configuration.

An analysis of input/output signal requirements for the various AUS subsections shows that they, as well as the TPS connection, the generation of remote starting signals for the control devices in a given building and the control message transmission may be effected using the following basic set of electronic modules :

- a multi-channel digital-input module with 96 input lines with opto-coupler isolation: input signals: 24 V 10 mA.
- a multi-channel digital-output module with 16 output lines with opto-coupler isolation; output signals: 5V, 400 mA;
- a multi-channel analog-input module with 16 input lines with galvanic isolation: input signals: +10 V with 5 mV precision;
- a multi-channel analog-input module with 16 input lines with galvanic isolation; input signals: 0-20 mA (this may be combined with the previous module);
- a 2 to 4 channel IRPS interface module; this will enable the data to be exchanged with the higher level controllers.

The EM data acquisition and control crates in one building are connected to a Utility Control Computer (UCC) using the MIL-1553 bus. The latter will provide the inter-linking of all sub-systems within a building and the transmission of the control messages to the sub-system controllers installed in the EM crates. The UCC provides the communications through the network with the accelerator control computer, as well as with the central accelerator control console and local consoles in the operations buildings or utility consoles in the services buildings. The above requirements are satisfied within the conventional architecture of accelerator control systems by a front-end computer (FEC) and one based on a Motorola 680xx microprocessor is used as the UCC .

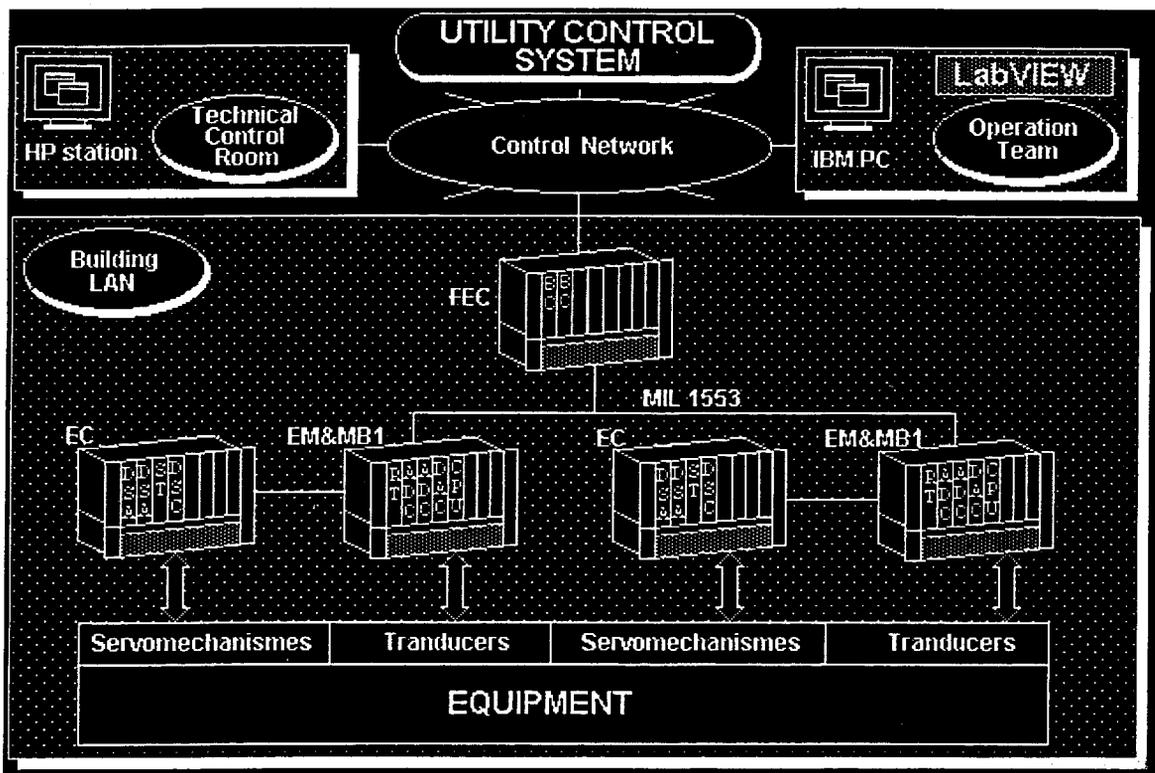


Fig. 1. Utility Control System.

A local console based on an IBM-PC computer is connected to the building LAN for start-up, tuning, repair and equipment diagnostics purposes and may also be used as a local operator console for all sub-systems in the building. In like manner, additional IBM-PC-based consoles, located in different buildings of the accelerator complex, may be connected, provided they have a Control Network connection.

Obviously this architecture for the UCS presupposes that reliable independent sub-systems will be

employed for the low level equipment control of each sub-system of the AUS. These independent control sub-systems must ensure the reliable operations of the AUS sub-systems even in case of malfunctions of the equipment at higher levels of the UCS (EM & MB1 and UCC) and the data links (MIL-1553 and computer network). Hence, these independent sub-systems for low-level control must comply with the current standards and must be attested accordingly.

In addition to the central accelerator control console, it is necessary to provide consoles for the UCS services in the premises of the respective services. Utility consoles provide information, displaying the state of the systems, while local consoles are mostly employed for test runs, diagnostics and repair.

## **The UCS software**

The MTOS may be used as the operating system for the EM & MB crate controller (PC 16 module), The remote terminal (RT) MIL-1553 bus driver, as well as a RS-232 serial port driver for possible later use, are to be built into the MTOS kernel. Furthermore, utility and application-level protocols are to be developed to download tasks from the higher-level computer (UCC), to the PC-16, to initiate or to suspend the execution of the given tasks, to remove the tasks from the controller memory through the MIL-1553 bus and/or through a RS-232 serial port, as well as to ensure the interaction in the command-reply mode.. The FEC operates under LynxOS and uses the standard means of UCS equipment access, adopted for all the accelerator control system, DEC workstations and ULTRIX are used for the accelerator control computers and consoles. The appropriate choice of operating environment for the local console is MS Windows, as this setting provides a uniform user interface for all applications. Since the development of applications under MS Windows is quite laborious work, the use of commercial software development kits and libraries seems to be the obvious solution to simplify the programming of applications and user interface. To interact with the higher-level computer through the network, the PC/TCP Development Kit of the Windows Sockets library is to be used.

LabVIEW for Windows, capable of creating an almost complete set of applications based on an IBM PC/AT, may be used for applications development. A set of VIs and servers which realize a protocol for remote access from upper level computers and consoles to the ECs has been developed. This software provides convenient tools for representation and access from LabVIEW applications to different accelerator equipment. The icon-based graphical programming system LabVIEW provides a powerful environment for control system design and gives significant gains in productivity with no sacrifice in performance and flexibility.

Such tools have already been used in the Beam Transfer Line (BLT) electricity control system and for creating a prototype for the BTL utility systems.

## **Conclusion**

The use of a hierarchical, networked architecture is suitable for this system. The distributed sub-systems can be incrementally upgraded - or even replaced - with a minimum of impact to the rest of the system. During the last accelerator run the BTL UCS was tested. The number of available menus on the local control console have been increased progressively to cover all the required functionality and their performance has been improved using feedback from the operators. Most problems occur when the different pieces of equipment are first connected to the network; mainly due to cabling faults. The main advantage expected from using the UCS is the fact that is an OPEN SYSTEM available for many services and the design is based on the key elements: integration, distribution, easy of use, expandibility and reliability.

## **References**

1. A.N.Alferov, B.L.Brook, A.F.Dunaitsev et al. "The UNK Control System", Proc. of the Inter. Conf. on Accelerator and Large Exp. Ph. Control Systems, Springer, KEK Tsukuba, Japan Nov. 11-15 1991, p 134.
2. Yu.I.Bardik, E.N.Kallistratov, A.A.Matushin et al., "Multidrop BUS Controller for IBM PC/AT", IHEP preprint, Protvino Russia, 1993, p 93-113.

3. J.Smolucha, A.Frank, K.Seino and S.Lackey "A New Architecture for Fermilab's Cryogenic Control System". Proc. of the Inter. Conf. on Accelerator and Large Exp. Ph. Control Systems, Springer, KEK Tsukuba, Japan, Nov. 11-15 1991, pp 96-100
4. C.Lemoine, M.Tyrrel "CAS and Alarm Console New Feature", SL/Note 93-102 ( CO ), November 1993. CERN.

# The Tesla Test Facility Injector Controls

F. Gougnaud, J.F. Gournay, Y. Lussignol  
CEA-Saclay DAPNIA/SIG 91191 Gif/Yvette France

## 1. INTRODUCTION

The goal of the Tesla Test Facility (TTF) project is to demonstrate the feasibility of a large superconducting positron electron collider. The collaboration includes several European and American laboratories. TTF is a linear accelerator with four cryomodules of eight cavities. The aim is to reach 500 Mev with a gradient of 15 MV/m. TTF will be installed at DESY. The Injector has been built by LAL (Orsay) and DAPNIA (Saclay) including its controls. There was an agreement between us and DESY in February 1993 to use EPICS for the injector controls [1]. This paper will present the architecture and the hardware of the system, the EPICS drivers, the EPICS tools used and some specific tools we have developed.

## 2. INJECTOR OVERVIEW

The TTF Injector comprises an electron gun with a high voltage system, a subharmonic buncher, two beam transport lines with magnetic lenses, triplets, dipoles and steerers. It also includes a superconducting capture cavity and a klystron modulator [2]. Different beam diagnostics are used : beam position monitors, beam pulse intensity monitors, a SEM-grid beam profiler, view screens and an optical transition radiation monitor.

## 3. HARDWARE

There are 4 dedicated VME crates (IOC in EPICS vocabulary) : one for the gun, one for the beam lines, one for the beam diagnostics and one for the capture cavity and the modulator. The CPUs in the VME crates are Motorola MVME162LX 222 (MC68040, 4MB DRAM). Most of the other VME boards are standard industrial products. These include:

- A/D boards ADAS ICV150 (12 or 14 bit)
- Fast A/D board OMNIBYTE COMET
- D/A boards ADAS ICV712 (12 bit)
- D/A boards ADAS ICV701 (16 bit)
- Binary Input /Output boards ADAS ICV196
- Stepping motor boards ADAS ICV914.

We chose ADAS boards because we have had good experience during the last years with these boards [3]. Furthermore, the manufacturer has a complete set of hardware to interface the VME boards to the equipment (isolated inputs/outputs, relay outputs, motor amplifier boards).

For SEM-grid and beam position monitors, the boards were designed by LAL. A programmable timing system designed by Fermilab, comprising a rep-rate generator and IP timer modules [4], delivers

synchronization to the gun, the capture cavity and the diagnostics. Lastly, we use an ESRF-designed VME board for video switching [5].

#### 4. EPICS DRIVERS

We have received and installed EPICS during the summer of 1993. After that, the first important step was to develop a set of drivers for all the VME boards we planned to use. There were no major difficulties in writing them, thanks to the EPICS manuals and to similar drivers which came with EPICS. In addition, discussions and E-mail exchanges with the original EPICS developers made the work easier. Initially, the standard structure consisting of a device and a driver layer was adopted as advised in the documentation [6]. But this structure was not always justified and we only kept it when necessary. Although this was probably obvious to the main collaborating laboratories, we didn't get the information and then wasted some time in reorganizing our drivers accordingly. This shows that even if a world-wide software sharing policy is very constructive, it may be difficult for a small, remote team to stay in phase with the major laboratories involved in the life of the product.

Most of the drivers were easily written except for the stepping motor which involves asynchronous processing. This type of driver should not block the database during the processing. Consequently, the driver puts the commands in a message queue. A readback task reads this queue, processes the commands on the VME board and allows the driver to hand the control back quickly to the database scanning tasks. The readback task updates the motor data in the database using a callback routine mechanism.

The free circulation of the EPICS source code is also a great advantage. For instance, the driver for the COMET board was first written at Los Alamos. It was then extensively modified at DESY and finally we made some minor enhancements to improve the software-event processing.

This gives some concern about the future life of our developments. As it is unlikely these drivers will be incorporated in the standard EPICS we will have to do this task each time we install a new release.

#### 5. USE OF EPICS TOOLS

Once the drivers were written, the displays for the operator interface were designed rapidly. We used MEDM (an EPICS tool that provides a graphical user interface) and KM (a tool assigning the SUNDIALS knobs to variables). Synoptic displays show the whole machine with diagnostics and magnets (Fig. 1). Steerers and lenses can be controlled with the SUNDIALS. The response times are very satisfactory. MEDM has all the general functions necessary for a control system and is used for most of the displays: the RF capture cavity, the diagnostics, the timing... For the RF capture cavity, with its cold tuning system, the display shows the very complex automation of the RF loops (Fig. 2). This was the first application delivered in June 1994. For the beam profiles, a MEDM display plots the raw data, the average smoothed data and shows additional information computed from the data.

SNL provides a simple tool for programming sequential operations based on the state transition diagram concept. It was used for different applications, such as to program the automation of the RF loops and for the calculations of the centre of gravity and the different widths of profiles. SNL was used for the standardization of user code in the IOCs and for simplicity in the database access method.

To build the databases of the different IOCs, the EPICS Database Configuration tool (dct) is used. Dct is a basic character-oriented tool which mainly lacks the ability to represent the links between the records (which could be rather complex) in a nice way. Nevertheless, this tool is adequate because we have split our databases in logical pieces small enough to be handled in this way.

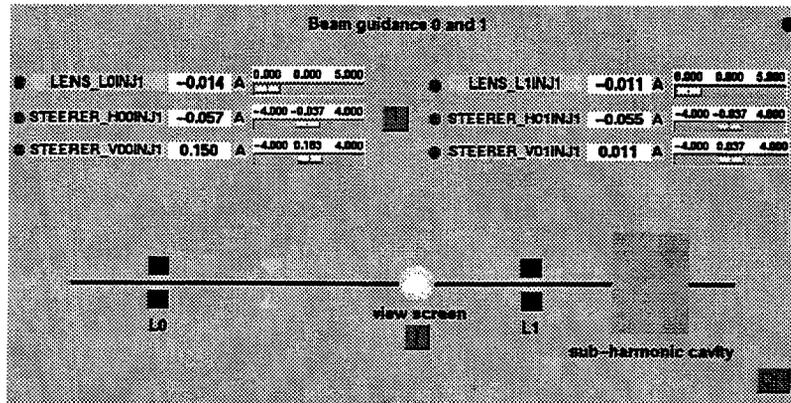


Fig. 1 A Synoptic Display for the Injector

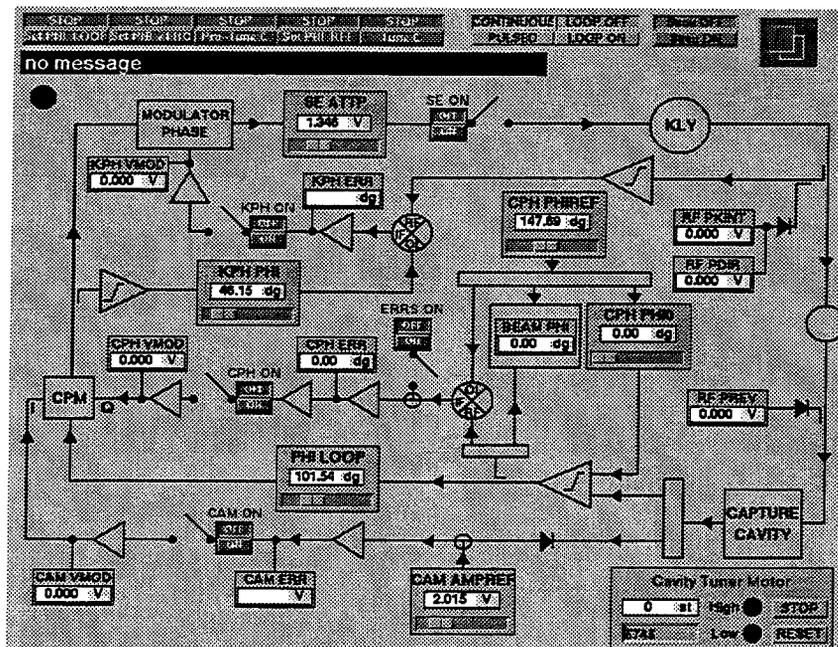


Fig. 2 Superconducting Capture Cavity Control Screen

## 6. NOMENCLATURE CONVENTIONS

To name parameters, the following convention is used : *device location property*. In general, the device name identifies the equipment (steerer, dipole etc.). For complex equipment like the gun or the capture cavity, the device name is, in fact, the name of a subsystem (high voltage for the gun or phase loop for the cavity). The meaning of the location field is obvious. In general, it consists of a number followed by the name of a logical sector of the machine. For complex devices, the location name is the name of these devices (gun or capture cavity). The property field refers to the physical parameter that is controlled like "intensity" or "voltage" or "status"... This property is followed by two lower case characters following the EPICS convention: ai for "analog input", ao for "analog output", bi for "binary input", bo for "analog output". All the characters are upper case except the last two. For instance, we have: LENS\_1INJ1\_iai (readback value), LENS\_1INJ1\_ONbo (on/off control), KPHLOOP\_CAPCAV\_PHIao (phase control for the capture cavity).

## 7. SPECIFIC TTF TOOLS

At DAPNIA we have developed some additional tools on the IOP side. The goal was to replace some existing tools (parameter page, BURT GUI) by tools which are more convenient for our own use, or to fulfill some specific needs (virtual oscilloscope, journal printing tool, error messages display).

The parameter page program is a tool which allows the user to control accelerator devices (e.g. power supplies). Its specifications are inspired by a similar tool used at Fermilab [7].

The configuration of a page is defined in a simple ASCII file with the names of the devices. In this file, it is also possible to specify pointers to a "next" or "previous" file in order to rapidly switch between pages. Finally, SUNDIALS knobs number can also be specified to automatically assign the knobs to the parameters. Each line displays the information for one device. The notion of device is implemented in a very simple way : all channels belonging to the same device are identified with the DESC field of the database. The parameter page has some additional features :

- push buttons to save a set of parameters and to restore them
- undo push button to return to the value before the last change
- "init" buttons to set the database alarm limits around the actual readback values. The intervals used to compute the limits can be set individually for each parameter
- trend plots for each readback value.

This tool needs to access the databases located in the IOCs. This is done in the EPICS world through the so-called Channel Access facility. A small library was developed which enables this communication with a maximum efficiency : support of asynchronous operations and monitors, cache to avoid repeated connections with the channels, support of complex get functions ( retrieval in one call of all information concerning a channel). A rather similar library was provided by APS but it came after our own development.

This tool is based on OnX, a GUI builder developed at LAL [8]. OnX is easy to use for developing complex user interfaces, but it has limited performance for updating such screens with dynamic numerical values and trend plots. Nevertheless, the extended functionalities and the nice look and feel (Fig. 3) will surely make this tool very useful for the control of the machine.

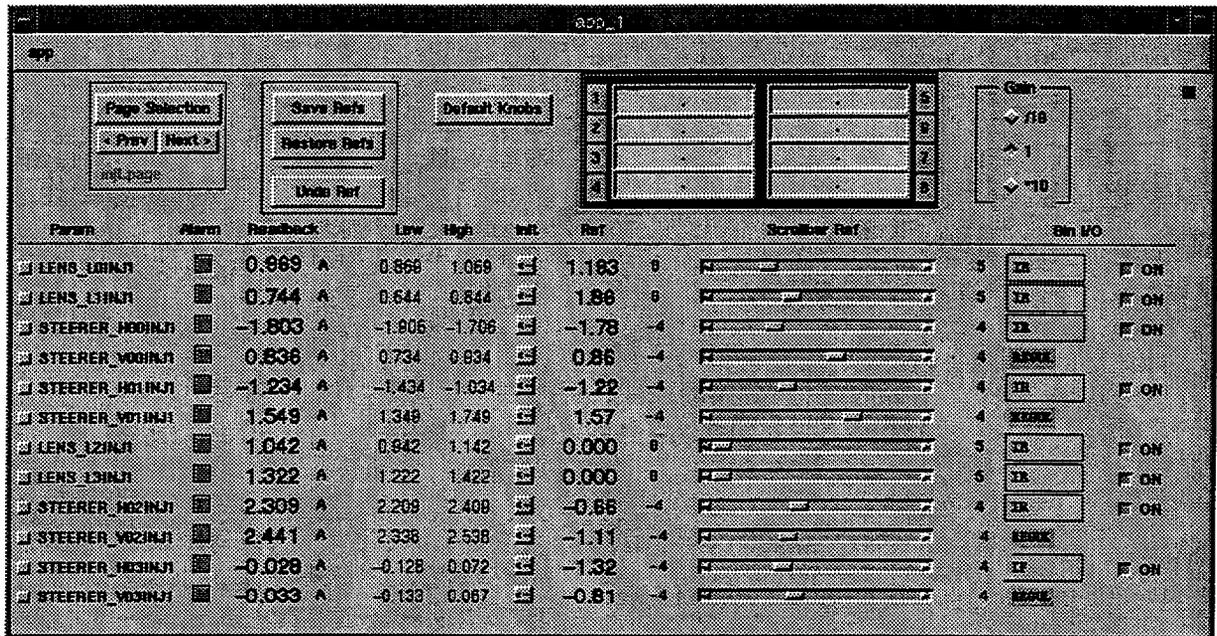


Fig. 3 Parameter Page Display

Some other tools were also developed :

A virtual oscilloscope used to display simultaneously the four channels of the COMET board. It has the basic functions which exist on a real oscilloscope : continuous sampling or one-shot mode, vertical and horizontal scale setting. For the injector, the COMET board is used for the beam pulse intensity measurements. This tool uses our in-house developed channel access library and is also based on OnX.

A journal printing tool : such a journal is composed of basic keywords and parameters known from the database. These keywords are used to define formatting instructions. When a request is made for printing a journal, the parameters are read from the IOCs, the journal is composed and sent to troff for formatting. The printout is used for the log book of the machine. The user interface is based on Tcl/Tk.

A front-end GUI for the backup and restore facility (BURT). This facility is used to save a set of parameters in a file and to restore this setup when needed. This program is extremely important during the startup and the commissioning phases. One such user interface already exists but it does not completely correspond to our requirements. In particular, it lacks the most common selection procedure we use : selection by the comments that are attached to the setups rather than selection by filenames. Furthermore, our interface has the minimal functionalities required, hence it is very simple to use.

A tool which displays the error messages of the system. EPICS provides a UNIX system-wide error server. All the incoming error messages go into a file but no tool is available to display these errors in real-time. Our tool shows the incoming error messages in a scrolled list. It is possible to enter a given string of characters in order to select messages containing this string (useful to check a given IOC for example). This tool was made with Tcl/Tk.

## 8. CONCLUSION

At the end of 1995, the TTF Injector control system runs for most of the components of the machine. The performance and reliability are very satisfactory. This system was developed with limited manpower, approximatively 3 man-years. The use of EPICS is indeed largely responsible for this good result. Our experience in developing control systems has convinced us for a long time of the benefit of standardization (the use of the graphical environment SL-GMS for several projects) and software sharing (the use of a

database inspired from the SLAC SLC database for 3 projects). However, with EPICS, one goes a big step further with the power of the database, the clean interface between this database and the clients through Channel Access and the large set of powerful tools available.

## ACKNOWLEDGEMENTS

We acknowledge the work done by M. Bernard, M. Mencik and M. Taurigna for the gun and the diagnostics. We would like also to thank all the members of the EPICS community who have helped us in some way to design our system and especially B. Dalesio from LANL and M. Kraimer from ANL.

## REFERENCES

- [1] M. Clausen. Controls for the Tesla test facility status and future plans (ICALEPCS'93, Berlin, October 1993).
- [2] D.A. Edwards (Editor) TESLA TEST FACILITY LINAC Design Report (DESY, March 1995).
- [3] J.F. Gournay et al. Le Système de Contrôle du Projet MACSE (Windays, Paris, April 1991).
- [4] M. Shea et al. IP 8-Channel Timer Module (Fermilab, September 1994).
- [5] J. Cerrai. Notice Multiplexeur Vidéo MUXV (ESRF, Grenoble, October 1990).
- [6] M. R. Kraimer. EPICS Input/Output Controller. Application Developer's Guide.
- [7] J. G. Smedinghoff. The Fermilab Accelerator Control System Parameter Program and Plotting Facilities (2nd International Workshop on Accelerator Control Systems, Los Alamos, October 1985).
- [8] G. Barrand. OnX : a tool to build interactive graphical applications (LAL Orsay).

# Accelerator Magnet Control from an SMP Workstation:

S. Herb and H. G Wu  
Deutsche Elektronen Synchrotron,  
Notkestrasse 85, 22603 Hamburg, Germany

## Introduction:

We report on the development of a prototype accelerator magnet control system based on an Axil 311 (Sparc 10 clone) multiprocessor workstation running the Solaris 2.4 operating system, a UNIX variant supporting soft real time scheduling and response in an SMP (symmetric multiprocessing) environment. The motivation for this work comes from several related considerations:

- The introduction of real-time capabilities in desktop operating systems may reduce the need for specialized real-time systems for driving front end hardware and permit task development and execution in more comfortable and standardized environments.
- Control of large groups of 'tightly-coupled' equipment, such as accelerator magnets, can be localized in self-sufficient modules which perform device coordination and error handling and present integrated device interfaces to clients, rather than having to export thousands of 'channels'.
- Multiprocessor workstations appear to be an excellent platform for combining the high-rate real-time I/O required for the front-end control of a large number of devices with the flexible higher-level tasks concerned with 'integration' and communications. One benefit of using more than one processor is that the complications and inefficiencies resulting from context switching and pre-emption can be drastically reduced. Another is that efficient intertask communication using shared memory can be available to a larger group of tasks.
- Commercial developments in this direction are very rapid, which ensures competitive costs and effective software support.

## System Description:

The Sparc station has at present two 55 MHz HyperSparc CPUs, each with 256 kB of external cache. It is connected to a VME crate via an SBUS to VME adapter (Performance Technologies SBS 915). In the VME crate are Motorola MVE 162 CPU cards, each driving four of the DESY specific 'Sedac' fieldbus lines used for control of the HERA power supplies. The present test setup has four lines; 16-20 lines will be required to drive all of the ~1300 magnet power supplies in the HERA electron and proton rings.

The MVE162 cards run simple queue processing tasks written for convenience using VxWorks. The Sparc VME interface is the sole VME bus master; it writes packets of fieldbus 'telegrams' into the VME memory queues, and later (asynchronously) collects the results returned. The writes and reads are performed on the Sparc side by a *Device Server* task. This task, described later in more detail, monopolizes one of the Sparc processors by running with Real Time priority in a polling loop.

Controls tasks running on the other processor include a *Network Server* which provides access for remote clients and '*Bump*' tasks which permit the control of groups of magnets. It is planned later to also include alarm and data logging tasks. These tasks communicate with the Device Server using two mechanisms built on Unix System V shared memory. The first is a system of shared memory queues and data buffers used for passing API messages (commands) between the tasks. The second is a real time shared memory database, which is continually refreshed by the Device Server and can be read out, via API calls, by the other tasks (read time 4  $\mu$ sec). The architecture is sketched in Figure 1. An important point is that the Sparc station will not serve as a console machine; whether X-windows access will be permitted at all during real operation is an open question.

The structures and tasks have all been implemented in C++. The shared memory queues permit write access from multiple tasks and therefore include mutex protection of the queue tail. Most of the low level structures are persistent, to avoid performance issues associated with dynamic object management. Experience so far with C++ is that it is well suited to this relatively low level programming and that it is superior to C (at least) for producing reliable and understandable code. A minor problem is that the creation of shared C++ objects in the shared memory becomes non-trivial once virtual methods are defined.

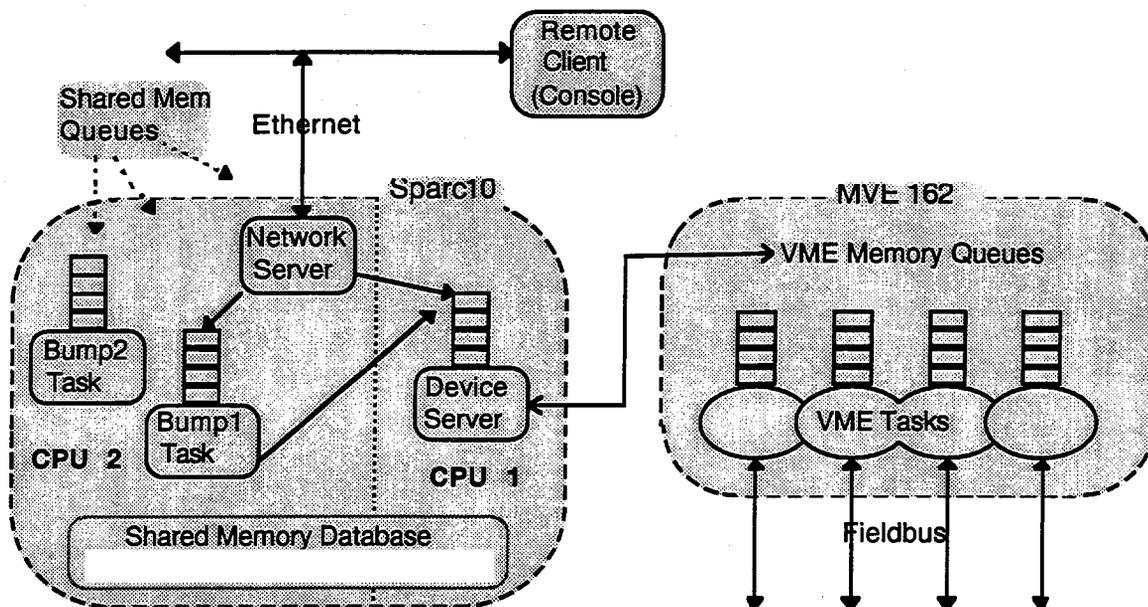


Figure 1 - Schematic drawing of the system architecture.

### Device Server Description:

The Device Server contains C++ objects representing each of the hardware devices controlled over the fieldbus lines and is responsible for the timely activation of each of these objects when a command for it is received. This is achieved by a 'command processing engine' operating on a single FIFO input queue, which for each command calls the requested device and method of generating the fieldbus operations and later, asynchronously, hands back the fieldbus results. 'Timely' activation is ensured by limiting commands at this level to single devices and single step operations and maximizing throughput. The goals are a maximum command throughput of roughly 10 kHz (including 3 kHz of data poll commands for the real time database refresh) and typical return times for requests from local clients (with contact to the shared memory) of less than 20 msec, permitting local clients to control groups of devices with a 50 Hz rhythm, if desired. An obvious limitation of such a system is that this rhythm may contain significant jitter, which is not a problem for our magnet control applications.

This architecture matches well the polling loop operation of the Device Server and could be characterized as 'loosely deterministic' rather than 'tightly deterministic', in that timings and priorities of individual commands are not rigidly ordered. The complete absence of context switching and interrupt handling permits significant simplification in the software design and the 'wasted' time in the polling loop is of little interest if the remaining CPU(s) can satisfy the other system needs.

### System Design Considerations and Goals

There is a great deal to be said about the considerations which led to this design; we content ourselves here with with a mixed list summarizing (or repeating) some of the points we found important. We should:

- like to control ~1300 Power Supplies for the HERA e and p rings from a single platform
- be able to drive the ~16 fieldbusses at a significant fraction of their full bandwidth
- be able to build magnet bumps from any of the various magnet types sitting on arbitrary fieldbusses
- have many magnet bumps active at once - use 'linear superposition' rather than 'ownership' as principle
- keep error handling for multi-magnet groups close to the hardware
- use command communication structure based on message passing and queues
- satisfy most reads from real time database rather than fieldbus access (esp. display update traffic to consoles!)
- use object description of hardware with common functions implemented via inheritance and polymorphism
- support local 'compound devices' (such as bumps) which offer remote clients control over groups of magnets
- support local supervision tasks (avoid unnecessary data transfer over the network.)
- use API protocol close to the 'standard model'
- build a Network Server as an independent module to permit easy modification or replacement

## System Performance:

A typical test run configuration has been included:

- a *Device Server* running in a polling loop in the RT (real time) class with the highest priority, with all pages loaded and locked into memory and with 100 ms time slices (essentially the rescheduling frequency)
- a *Network Server* running in the RT class with lower priority and 'waking up' every 50 ms to send data refreshes (read from the real time database) to a remote client.
- several local '*Bump*' tasks running in the RT or TS (time-sharing) class, 'waking up' every 20 ms to test the return status of commands submitted during the previous cycle, then submitting new commands to the *Device Server*.
- Assorted user jobs running in the TS class, either for supervision, or in an attempt to test for interaction with the controls jobs (performing a compile or starting xemacs is a good way to generate some disk activity).

These tests indicate that we can achieve the desired command throughput and that there is no problem in running many 'bump' tasks at the 50 Hz rate. We also looked in more detail at the behaviour of the tasks running in the RT class; we measure and histogram the time for the device server to pass through the polling loop using the *Solaris 2.x gethrtime()* call, which returns (in 3  $\mu$ sec) a 64-bit non-wrapping time with sub-microsecond resolution. The maximum computational work for one pass through the loop takes certainly less than 500  $\mu$ s. We may therefore assume that loop transversal times of 1 ms or longer represent deadtimes related to system activity.

The response of Solaris 2.4 is usually specified in terms of the interrupt dispatch latency, which should be between one and two ms, and is related to the non-pre-emptible kernel code sections. What this means for a polling task in an SMP environment is less than clear; the RT task has a priority higher than all system tasks, but lower than all interrupt handlers. In any case, we see a significant number of traversal times in the 1-2 ms range, representing roughly a 0.002% duty factor. More disturbing are measurements taken over time intervals which include heavy disk access activity for TS class tasks, occasioned by the start of Xemacs etc. when we then see several events in the 10-20 ms bin. Occasional 10 ms gaps would not be a severe problem for our magnet control but could be extremely unpleasant in some other real time environments. A recent exchange on usenet suggests that this is an artifact from a scheduler problem which has been identified and fixed in Solaris 2.5.

Another point is the performance of time-share class jobs in the presence of real-time jobs using one processor and part of the second. Performance in executing tasks seems reasonable; an amusing point is that the response of the windowing system on the parent console, on which the real time tasks have been started, is rather sticky, while the response for remote X-windows login is normal.

## Conclusions:

The term 'Distributed Control System', covers a wide range of possibilities. The model proposed here is of a node powerful enough to drive a large group of devices which functionally belong together, while in addition supporting a variety of higher level tasks. This permits running flexible supervisory and maintenance tasks, as well as tasks which provide remote clients with interfaces to groups of devices, as opposed to 'channel oriented' systems which emphasize low level data transfer. In general, we are trying to move more intelligence and self-sufficiency down close to the front end.

SMP workstations with real time capabilities appear well suited to this model. Very interesting in this regard is the progress of the POSIX.4 (more properly, 1003.1b + c,d,...) standards for real-time calls, which have already been incorporated in hard real-time systems such as LynxOS and VxWorks and are now gradually moving into the mainstream operating systems so that real-time programming is becoming a less esoteric activity.

## Acknowledgements:

We would like to acknowledge the ESRF, CESR, and HERA PKTR control systems as sources for some of the prejudices which have gone into the system here; obviously they are in no way to blame for the results. Special thanks to K. H. Mess and F. Willeke at DESY for their support, to K. Rehlich, M. Clausen and P. Duval for discussions and to comp.unix.solaris and comp.lang.c++ for many useful hints.

# THE CONTROL SYSTEM FOR HIRFL

Tuanhua Huang  
 Institute of Modern Physics, Academia Sinica  
 Lanzhou, Gansu, 730000  
 The People's Republic of China

## I INTRODUCTION

The Heavy Ion Research Facility in Lanzhou (HIRFL) consists of an Electron Cyclotron Resonance (ECR) ion source, a Sector Focusing Cyclotron (SFC), a Separated Sector Cyclotron (SSC), and 8 experimental terminals; there are 3 beam transport lines to link them. The subsystems of the HIRFL control system are divided in a corresponding manner [Figure 1].

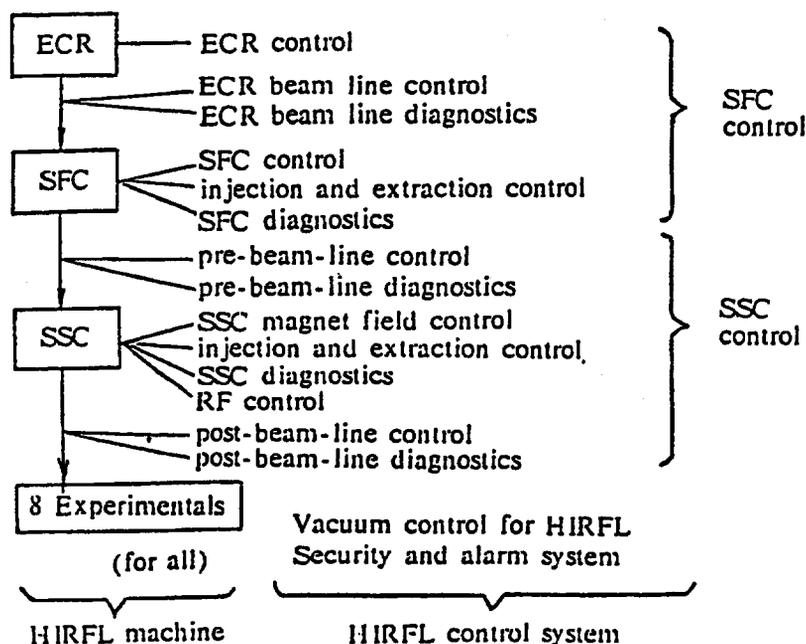


Fig.1 The Layout of HIRFL Control System

Basically the HIRFL control system may be divided into two parts - the SFC control part and the SSC control part. The main engineering construction of the subsystems in the SSC control part was finished by 1988 before the first ion beam was successfully extracted from the SSC. These subsystems had primarily been used for beam tuning for 4 years. The main engineering construction of the SFC control part was put into use by April 1992, not long after the end of construction of the ECR ion source and upgrade of the SFC.

The centralization of SSC control was completed in 1993 and is now functional and stable (Figure 2)

## II SUB-CONTROL-SYSTEMS IN SSC CONTROLS

The subsystems in the SSC control part are described as follows:

### 1. Power supply control for SSC magnets

This subsystem consists of a CAMAC crate with I/O modules connected to circuits in NIM crates. One NIM crate is located near the CAMAC crate and the others are near the power supplies. The circuits in the NIM crates are multiplexers for adjusting and measuring the currents of the power supplies, and for switching and monitoring them. DACs and voltmeters are employed in adjusting and measuring channels.

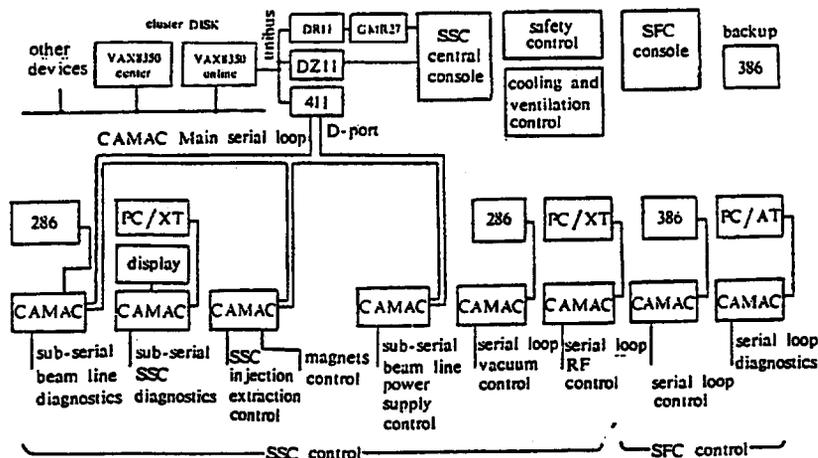


Fig.2 The Control System for HIRFL

## 2. Control of SSC injection and extraction elements

In the control room ER16VM modules are located in a CAMAC crate. At the sites of the injection and extraction devices there are NIM crates with circuits to perform adjusting and measuring of the currents (or voltages) via DACs and ADCs, and to perform switching and state monitoring of the power supplies. Also performed are position adjustments and the alarms and interlocks for cooling water in these elements.

## 3. Power supply control for the beam lines from SFC to SSC and from SSC to the 8 experimental terminals

A CAMAC crate linked to a VAX8350 is employed to operate the CAMAC sub-serial loop via a 3992 serial highway driver and a 3936 U-port. There are 4 CAMAC crates near the power supplies on the sub-serial loop. The I/O modules in these crates connect to the circuits in NIM crates inside the power supply enclosures to perform adjustment and measurement of the currents, and for switching and monitoring power supply states.

## 4. The beam diagnostic subsystem in the beam lines from SFC to SSC and from SSC to the 8 experimental terminals

The beam diagnostic elements, such as multiwire chambers, Faraday cups, slits and phase probe units are connected to circuits in NIM crates, and then to I/O modules of 6 CAMAC crates distributed on the sites of the beam lines. All the crates are linked as a sub-serial loop to the control room where a T&W computer is employed as a front end. A 3922 crate controller is used as an Auxiliary Crate Controller (ACC) which connects to a T&W microcomputer, and a 3952 is used as a Main Crate Controller (MCC) which connects with the central VAX8350. A 3821 memory module is included to act as a mailbox linking the two computers, VAX8350 and T&W. In this way the centralization of subsystems with PCs is realized.

## 5. Beam diagnostic subsystem in the SSC

In the control room an IBM PC/AT computer is connected to a CAMAC crate via a crate controller. A 3992 CAMAC serial highway driver in the crate drives a serial loop which links 2 CAMAC crates in the SSC hall. Some I/O modules in the 2 crates link to circuits in NIM crates where the signals from the SSC beam diagnostic elements, such as radial differential probes, Faraday cups and position probes, are gathered and the motion control signals to these probes are sent. This subsystem will be centralized through the VAX8350 via the same mechanism described in 4 above.

## 6. The fire alarm and security interlock system

Smoke detectors are distributed throughout the whole of HIRFL. They are all connected to the control room where alarms will be heard. For safety of both personnel and hardware, an interlock system was constructed for all HIRFL, and its status will be displayed on the central console.

### 7. *RF control and vacuum control*

RF and vacuum are controlled by a CAMAC system run from a PC. They once worked independently, but will be centralized and operated from the console.

### 8. *Others*

Cooling and ventilation control run independently in a traditional style. They also will be connected to and shown on the console.

## III SFC CONTROLS

In the SFC control room there is a console which consists of an AT&T 6386/25 computer, 2 touch panels, 2 color monitors, a storage oscilloscope, etc. and a CAMAC crate which is connected to the computer via a 2925 module and a 3920 crate controller. In the CAMAC crate are modules to operate the devices in the console, and a 3992 serial highway driver and a 3936 U-port to drive a CAMAC loop, with 4 crates distributed on the sites where the controlled devices are located. The CAMAC modules in 4 crates operate the devices via circuits in NIM crates and designed by our staff.

The functions of SFC control include:

1. ECR ion source control
2. Power supply control for the beam line from ECR to SFC
3. Power supply control for the SFC
4. Position control for injection and extraction in SFC
5. Beam diagnostics in the beam line from ECR to SFC
6. Beam diagnostics in the SFC

The circuits for ECR ion source control and power supply control were designed similarly, while those for position control for injection and extraction are different. The beam diagnostic subsystems are similar to those in the SSC control part and temporarily use an IBM PC/AT computer rather than the AT&T one. The SFC console has not been used as yet except for the AT&T computer.

The operating programs to perform the 6 listed functions are written in FORTRAN and run separately on two computers.

## IV CENTRALIZED CONTROL

Two VAX8350s are clustered as the central computers for HIRFL control. There is a CAMAC parallel and a JY411 serial driver in the SSC control room. The JY411 drives several CAMAC crates to link all the SSC sub-control-systems. Three subsystems for power supply control are linked without microcomputers, and the 2 sub-control-systems of beam diagnostics are linked with them; communication between VAX8350 and microcomputer is via CAMAC dataway as explained above.

In the SSC control room there is a console with 6 touch panels, 6 color monitors, 6 displays, a storage oscilloscope and a TV. A VAX8350 links touch panels via a DZ-11 interface, and monitors and displays via a DR-11 and a GMR-27 controller; thus we may operate the SSC control system on this console. In 1993, after having done preparatory work, the sub-control-systems were connected into the central computer and the operation of the SSC was moved to the console. All the data sent out to operate SSC devices come from the touch panels. The 6 sets of panels, color monitors and displays are functionally the same, which is convenient for operation. The TV runs independently to look at specific sites of HIRFL while the facility is being operated.

The VAX8350 and PCs will be linked by Ethernet (Figure 3). Communication between them via the DECnet-VAX and DECnet-DOS transparent task-to-task technique has been undertaken. The console for SSC control will be used for the whole HIRFL control system when communication between SFC and SSC controls via Ethernet/DECnet is completed.

## V SOFTWARE

The operating programs of HIRFL consist of two parts, one for operating the sub-control-systems and another for operating the console devices. Most of the programs are written in FORTRAN developed on VAX/VMS 4.5, some are in MACRO and new ones in C. In order to access CAMAC circuitry easily, the CAMAC software package developed by the Fermilab controls group was used.

The operating program of each sub-control-system was written by our staff member who was responsible for that subsystem itself. The programs had been run separately for several years.

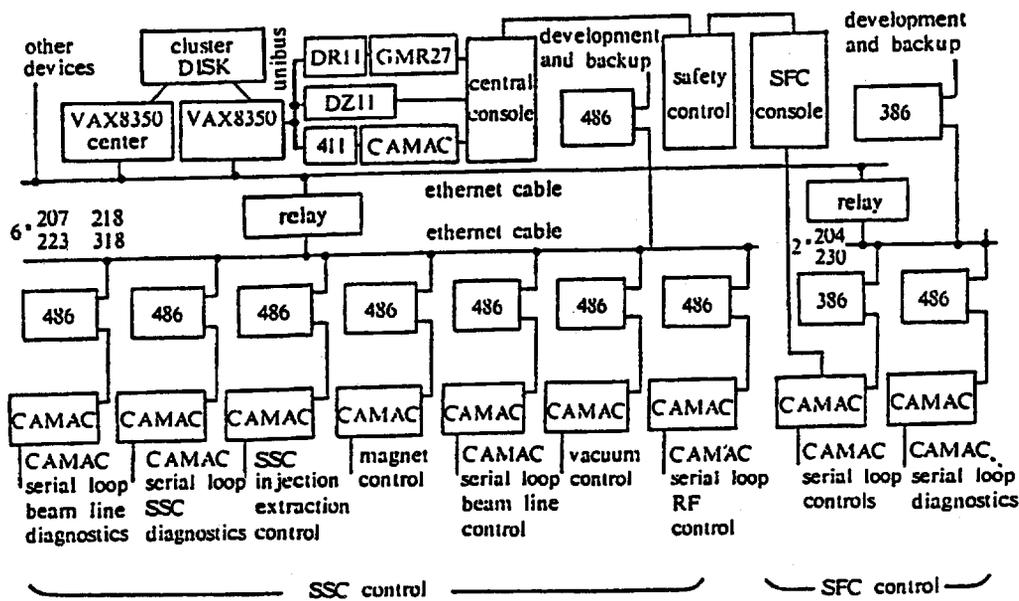


Fig.3 The Fracture of HIRFL Control System

The operating programs for console devices can be divided into two - manager programs and display-control programs. They are developed by our staff, and are convenient for linking existing operating programs of sub-systems by a variety of techniques.

In order to perform centralized operation on the console, the following steps are taken:

1. Run the console management program to select touch panel and color display, to select the sub-system under control and to operate the touch panel as if operating on a keyboard.
2. Set up sharable memory for data and status commands. The central management program and sub-system programs equally access it by calling subroutines to exchange the operating information; they do not influence each other.
3. Run the start-up program to make all the applications active processes by DCL commands. Thus the whole system is put into service.

# Design for the Control and Data-Acquisition System for the TAMA300 300m Laser Interferometer

Norihiko Kamikubota

National Laboratory for High Energy Physics (KEK), 1-1 Oho, Tsukuba, Ibaraki 305, Japan

Nobuyuki Kanda

Institute for Cosmic Ray Research (ICRR), Univ. of Tokyo, 3-2-1 Midori-cho, Tanashi, Tokyo 188, Japan

Ei-ichi Mizuno

The Institute of Space and Astronautical Science (ISAS), 3-1-1 Yoshinodai, Sagamihara, Kanagawa 229, Japan

Masatake Ohashi, Ryutaro Takahashi, Toshitaka Yamazaki

National Astronomical Observatory (NAO), 2-21-1 Osawa, Mitaka, Tokyo 181, Japan

## Abstract

A five-year project, "TAMA300", was started in 1995. The purpose of the project is to construct a 300m laser interferometer in Tokyo to search for gravitational waves. The studies of the control and the data-acquisition system for TAMA300 have been carried out and a conceptual design is presented. We are evaluating an EPICS system for devices which are well distributed along the interferometer arms and need to be supervised at a rate of 1–100 Hz. We also need a data-acquisition system with high-speed ADCs to record 14 interferometer signals at the rate of 20 kHz. In addition, we are investigating a scheme to merge the signals from both the EPICS and the data-acquisition systems into a raw-data archiver. The scheme includes a data server to provide sampled data for online analysis.

## I. Introduction

Although the existence of gravitational waves is predicted by the general theory of relativity, there has so far been no direct observation. Much effort has been put at many places in the world into the direct detection of gravitational waves.

Recent interest has been concentrated on long-baseline laser interferometers [1]. Virgo (the French-Italian project to construct a laser interferometer with 3-km arms in Pisa [2]) and LIGO (two interferometers with 4-km arms in Washington and Louisiana [6], [7]) are typical examples, which are now under construction and are expected to make their first observations in 2000 or 2001. In Japan, a five-year project, "TAMA300" (Tokyo Advanced Medium-scale Antenna), was started in 1995.

The aim of the TAMA300 project is to construct a 300-meter laser interferometer in Tokyo [9]. Though the TAMA300 has a shorter baseline compared with the other km-size interferometers and accordingly less sensitivity, it has two significant advantages. First, the operation of TAMA300 is expected to start in 1998, which is earlier than other projects. Second, the location of TAMA300 in Asia is quite important to complete the world-wide coverage with gravitational antennas, since coincidence studies between separated interferometers are necessary to determine both the direction and phase of the arriving gravitational waves.

Some studies concerning the control and data-acquisition system for large interferometers have already been carried out. The Virgo group has studied control, monitoring and data taking systems [3], [4] as well as a simulation code [5]. For the LIGO project, a system called CDS (Control and Data System) has been developed to control interferometers and to manage various data [8]. We also carried out early studies of interferometer control systems in 1991 [10], [11].

In order to develop a control and data-acquisition system for TAMA300, we started a collaboration between NAO, KEK, ISAS and ICRR in 1994. The primary aims of the system are as follows:

- a) data acquisition for interferometer signals with a 20-kHz sampling rate
- b) remote control, monitoring and data logging of interferometer devices
- c) online pre-analysis for gravitational-wave candidates.

Although the systems used for offline analysis and the simulation of gravitational signals are also of interest, they will be discussed elsewhere in the future.

## II. Signals to be recorded

As a first step, we discuss the signals to be recorded (or at least monitored). Signals can be divided into two groups: The first group contains the high-rate signals. Interferometer signals, laser signals and signals related to online analysis belong to this group. Since the shapes of gravitational waves may have a structure with components of up to a few kHz, we have set the sampling frequency at 20 kHz. As given in Table I, 14 signals are planned to be recorded.

The second group contains the low-rate signals. The major part of the second group contains signals related to mirrors. Alignment control signals to keep the mirrors stable against seismic noise and the signals of the mirror positions will be recorded at a rate of 100 Hz. Signals from vacuum controllers and from environmental monitors are also included in this group. Those will be recorded at a rate of 1 Hz or lower. A summary of the signals belonging to the second group is also given in Table I.

The total amount of signals is estimated to be 600 kbytes per second. The major contribution comes from the high-rate signals.

Table I  
List of signals to be recorded with the TAMA300 gravitational-wave antenna.

signal category	signal name	sampling	channels
interferometer	main signal	20kHz	1ch
	visibility	20kHz	1ch
interferometer feedback	main	20kHz	2ch
	beam splitter	20kHz	1ch
	recycling mirror	20kHz	1ch
	mode cleaner	20kHz	1ch
	power	20kHz	1ch
laser	frequency stability	20kHz	1ch
	visibility	20kHz	1ch
	h	20kHz	1ch
online analysis	matched filter	20kHz	3ch
mirror control	main(4deg.freedom)	100Hz	4ch
	recycling(4deg.freedom)	100Hz	1ch
	mode cleaner(4deg.freedom)	100Hz	2ch
mirror isolation	main(5deg.freedom)	100Hz	4ch
	recycling(5deg.freedom)	100Hz	1ch
	mode cleaner(5deg.freedom)	100Hz	2ch
environment	beam splitter(5deg.freedom)	100Hz	1ch
	seismic noise, acoustic noise, etc.	100Hz	30ch
vacuum	temperature, etc.	1Hz	100ch
	vacuum status	1Hz	20ch
GPS monitor	valve status	1Hz	50ch
	clock	1Hz	2ch

## III. Proposed system for control and data acquisition

### A. EPICS for low-rate signals

Since most of the signal sources are well spread along the interferometer arms, it is obvious that a network-oriented control and data-acquisition system is necessary. In addition, we do not have much manpower and have only three years remaining to complete the project. We have thus looked for a platform on which to develop our own system, and have found that EPICS (Experimental Physics and Industrial Control System) [12] seems to be the best choice among several commercial products and public-domain tools.

We have installed an EPICS test bench at ISAS and are now evaluating its performance with dummy signals. At present,

the EPICS seems to satisfy our requirements for low-rate signals. The proposed system consists of a few Unix workstations, X-terminals and roughly ten VME-bus computers, which are connected through a network to each other. We expect EPICS to be an easy supervisor of the low-rate signals.

### *B. High-rate data acquisition*

It is not easy to collect high-rate signals with EPICS, mainly due to network capacity. Since all of the high-rate signals are localized near to the central control room in our case, we have looked for another high-rate data-acquisition system. At present, we are interested in a commercial data-recorder with VXI-ADC modules and an archiver. A serious problem is that we need to record signals continuously. In order not to interrupt an observation, we require an intelligent raw-data storage, such as a tape-robot. In addition, we must decide on the archiver media among several candidates such as tapes (DAT, Exabyte, DLT, etc.) or removable disks. Since progress in this field is very fast, we will leave the final decision until 1997, which is one year prior to the start of observations.

### *C. Event builder and Data server*

We need an "event builder" to merge both the high-rate and low-rate signals together. As shown in Figure 1, the event builder is in charge of creating a data package, called an "event packet", every two seconds, which contains the raw data of both the high-rate and low-rate signals. Each event packet exists in memory for roughly 10 seconds and is then overwritten with a new one. During this interval it remains in memory and is transferred into a raw-data archiver. The data transfer from the high-rate data-acquisition system to the archiver is made through a dedicated high-speed communication channel.

An event packet is used as a basic unit during the various data treatment, including the online and offline analysis. For easier handling of the data, the size of one packet has been designed to be on the order of 1 MB. It may include an additional header, such as a time stamp, a serial number, and so on, for later analysis<sup>1</sup>. The detailed structure of a packet is to be determined in the future.

The data server is also shown in Figure 1. Its role is to provide the latest data for an online analysis. Sampled packets, as many as the CPU power allows, are copied into the files on large scratch disks. The size of the disks is roughly 400 GB in total, so as to keep the data for one week.

We are investigating a physical scheme to realize the mechanism mentioned above. The most promising idea is to use a reflective memory in order to share on-memory packets with many nodes, including the high-rate data-acquisition, the event builder and the data server. A feasibility study is in progress.

### *D. Network*

The network is an essential part of our system. We have the idea of dividing it into a few segments. We will install at least two networks along each arm. One will be an FDDI network having a sufficiently high capacity for data transfer for the EPICS system. This network will be used only for data transfer. Another is the popular Ethernet, which is to be used for very low-rate data transfer, such as the vacuum-monitoring signals, and for general purposes, including X-window protocols. In order to avoid troubles caused by lightning during the summer season, optical isolation is to be introduced.

### *E. Schedule*

The schedule of the TAMA300 project for the next 5 years is summarized in Table II. The time limit for completing the control and data-acquisition system is 1998.

<sup>1</sup>Early consideration was found in [13].

Table II  
Schedule for developing the system for TAMA300.

Year	Goal(s) for each year
1995	Evaluation of EPICS with a test bench
1996	Construct the EPICS Supervisory system for low-rate signals Install network environment
1997	Construct the data-acquisition system for high-rate signals
1998	Install systems for actual observation and analysis Short observations for performance check
1999	Start the continuous observation for gravitational waves

#### IV. Acknowledgement

We wish to thank Dr. Kazuro Furukawa and Dr. Yujiro Ogawa for various discussions and suggestions. We also express our thanks to our colleagues in the TAMA300 group for kind encouragement. The present work is supported by a Grant in Aid of Scientific Research, Ministry of Education, Science, Sports and Culture, Japan.

#### References

- [1] For instance, K.S. Thorne, Proc. of the Snowmass 95 summer Study on Particle and Nuclear Astrophysics and Cosmology, August 1995, (World Scientific, 1995), in press
- [2] A. Vilenkin, Phys. Rev. D24(1981)2082
- [3] Virgo final conceptual design (1992), chapter VI and chapter VII
- [4] Virgo note PJT93-16
- [5] B. Mours and I. Wingerter-Seez, Proc. the Sixth Marcel Grossmann Meeting on General Relativity (MG6), Kyoto, June 1991, (World Scientific, 1992) p.1576-1578
- [6] A. Abramovici et.al., Science 256(1992)p.325-333
- [7] S.E. Whitcomb, Proc. of the Fourth Int'l Symp. on Particles, Strings and Cosmology (PASCOS94), New York, May 1994, (World Scientific, 1995) p.300-307
- [8] private communication, [http://www.ligo.caltech.edu/LIGO\\_web/overview/CDS.ps](http://www.ligo.caltech.edu/LIGO_web/overview/CDS.ps)
- [9] K. Tsubono, Proc. First E. Amaldi Conf. on Gravitational Wave Experiments, Frascati(Rome), June 1994, (World Scientific, 1995) p.112-114
- [10] N. Kamikubota, K. Furukawa, K. Nakahara, Y. Ogawa and A. Asami, Proc. the Sixth Marcel Grossmann Meeting on General Relativity (MG6), Kyoto, June 1991, (World Scientific, 1992) p.1480-1482
- [11] N. Kamikubota, K. Furukawa, K. Nakahara and I. Abe, Proc. Int'l Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'91), Tsukuba, November 1991, KEK Proceedings 92-15, p.318-321
- [12] For instance, <http://epics.aps.anl.gov/asd/controls/epics/home.html>
- [13] J.R. Shuttleworth, Univ. Coll Cardiff, private communication

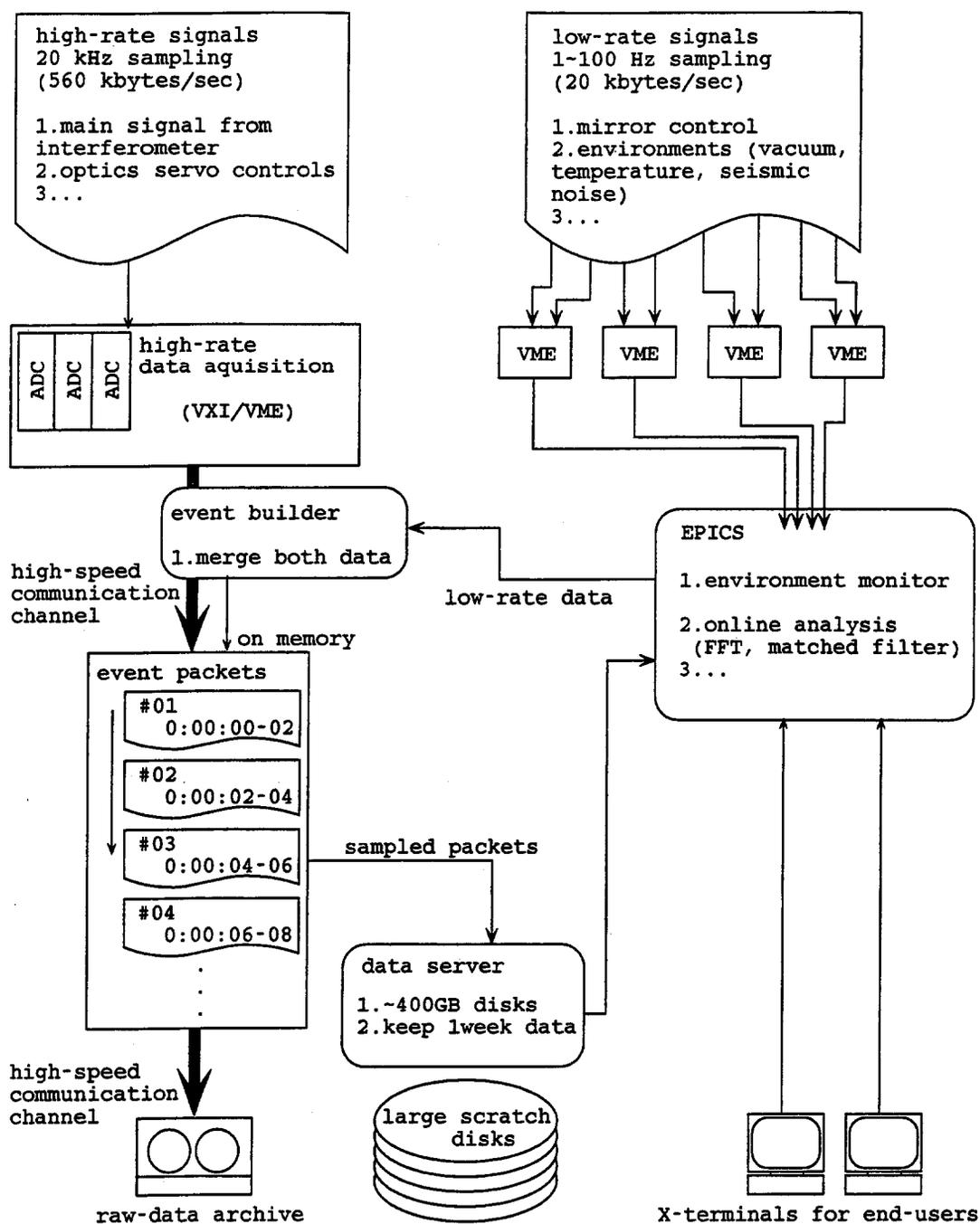


Figure 1. Schematic view of the data flows between the high-rate data-acquisition, the EPICS, the event builder, the data server, and the raw-data archiver.

# STATUS OF THE KEKB ACCELERATOR CONTROL SYSTEM DEVELOPMENT

T. Katoh, T. Mimashi, T.T. Nakamura, N. Yamamoto, A. Akiyama, S. Araki,  
T. Kawamoto, K. Komada, K. Kudo, T. Naitoh, T. Takashima and S. Kurokawa  
KEK, National Laboratory for High Energy Physics  
1-1 Oho, Tsukuba-shi, Ibaraki-ken 305, Japan

## INTRODUCTION

KEKB is a 5-year project started in 1994 to convert the existing TRISTAN electron-positron accelerator/collider into an asymmetric one.[1] TRISTAN is a three-stage cascade accelerator/storage ring with a 2.5Gev electron/positron injector Linac, 8GeV Accumulation Ring (AR) and 33GeV accelerator/collider Main Ring (MR). For the KEKB project the Linac will be upgraded to provide intense 8GeV electron and 3.5GeV positron beams to inject directly into the KEKB rings. The KEKB rings, the High Energy Ring (HER) and the Low Energy Ring (LER), will be installed side by side in the tunnel where the TRISTAN MR is at present. The TRISTAN MR will be shut down at the end of 1995 and the equipment will be removed from the MR tunnel. As the 8GeV AR will not be used by KEKB, it will be used as a synchrotron light source and for accelerator experiments. At present, TRISTAN is controlled by a system using mini-computers and CAMAC [2], but modern technologies will be introduced for KEKB controls. To save money and manpower, the CAMAC system will be reused after rearrangement. The present control system for TRISTAN was designed about 15 years ago and has become very hard to maintain. It is clearly important to have a good control system for a modern intense beam asymmetric collider. The quality of the control system required by KEKB is much higher and the quantity greater than that of the present system. However, the progress in the fields of computer hardware and software seems to give us the chance to have a more sophisticated control system and to make it possible for us to construct a control system which satisfies requirements in both quality and quantity. The preliminary design has been reported previously [4].

## FUNCTIONAL REQUIREMENTS

The control system design must be looked at from as many points of view as possible. Usually the suppliers of the system and the users tend to have different points of view. Therefore, we first started to review functional requirements of the control system for the KEKB collider. The hardware groups that install the accelerator components to be controlled want to monitor magnet current values, voltages of the power supplies and so forth, and if a failure occurs they want it to be reported and recorded as soon as possible. The operations group wants the recording of the operations sequence of the collider and to have tools to analyze the previous operations and make correlations between parameters and measured values for tuning purposes. Accelerator physicists need the connection between beam monitoring, simulation and beam correction systems. The controls group itself has its requirements.

The principal functional requirements are as follows:

- 1) All the data that can be taken should be taken.
- 2) All the data taken should be saved for later analysis
- 3) All the operations should be recorded for later inspection.
- 4) All the machine parameters and data about the machine components should be saved in the database.
- 5) The control system should be operator-friendly.
- 6) The programming system for application programs should be programmer-friendly.
- 7) The overall response time to an operator's request should be less than a few seconds, ideally one second, unless the progress of the process is indicated.

## CONSTRAINTS

The KEKB collider is a reconstructed machine and there are many constraints in constructing its control system economically and effectively. The first of them is CAMAC. We have already accumulated a large number of CAMAC modules in the TRISTAN control system and CAMAC is still thought to be the only well-defined standard for interface modules which can be used for process input/output. The old mini-computers will be removed from MR sub-control buildings and the CAMAC equipment configuration will be redone and some new CAMAC modules will be added. On the other hand, there is no constraint to introducing a new software environment, because all the application programs will be new.

The final commissioning of KEKB is scheduled to take place in 1998, so the control system should be installed at least one year before then. The basic system for the hardware and software development will be installed at the beginning of 1997. It will include some workstations, VME computers and X-terminals for the operator's consoles. Another constraint is that there are only a dozen people in the KEKB controls group.

## BASIC DESIGN CONCEPTS

Considering the requirements stated above and the restrictive conditions, the basic concepts are to use:

- 1) the so-called Standard Model for the accelerator control system.
- 2) international standards for the interfaces between the three layers, to facilitate later upgrading and maintenance.
- 3) international standards such as CAMAC, VME, VXI and GPIB as the interfaces between the control system and the equipment to be controlled
- 4) products either from international collaborations or that are commercially available, to minimize the manpower and effort required
- 5) the object-oriented technique or abstraction to hide the hardware from application programmers,
- 6) high-speed networks to connect the computers with each other to get a quick response,
- 7) the linkmen system as in the construction of the TRISTAN control system, in which the linkmen make the equipment database and code device drivers for the application programs because they know the equipment best.

Finally, we decided to use EPICS as the KEKB control system environment, and we joined the EPICS collaboration.

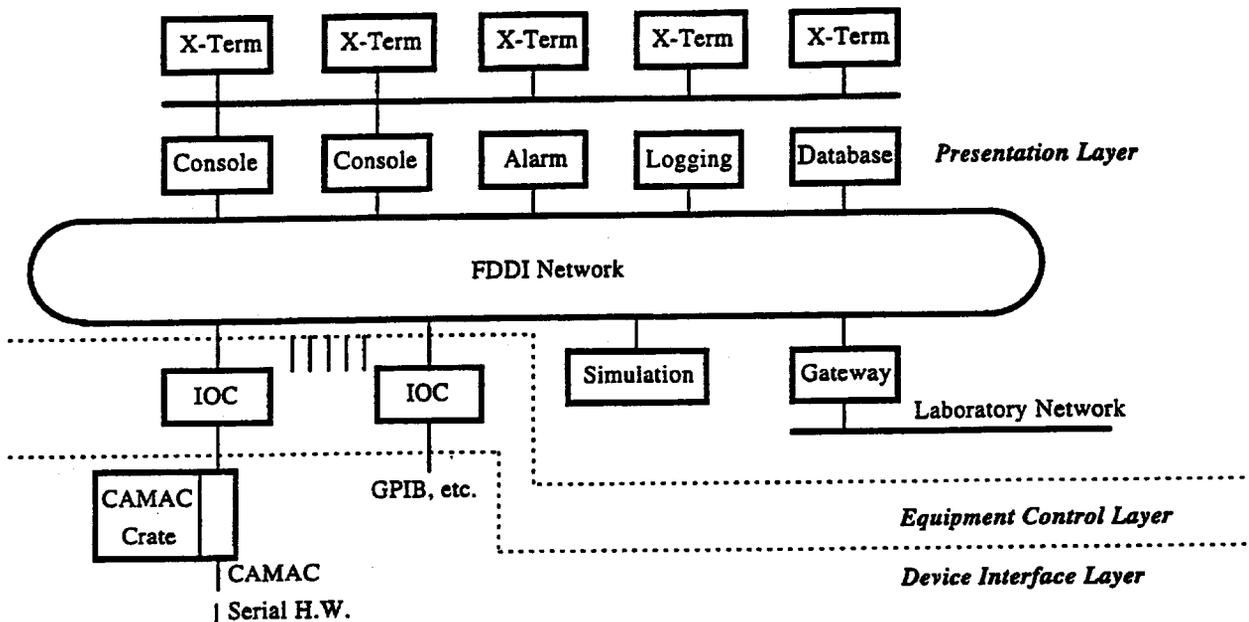
## SYSTEM ARCHITECTURE

The control computer system is divided into three layers - presentation layer, equipment control layer and device interface layer, as shown in Figure 1. The first two layers are divided functionally but connected with each other through a high-speed network such as FDDI. This is because the network traffic between the presentation layer and equipment control layer computers is expected to be dominant. There is also a possibility of adopting a distributed shared-memory network to obtain fast data transfer and event transmission among the computers. The presentation layer which is called an Operator Interface (OPI) in the EPICS world is composed of several workstations and about 6 to 10 X-terminals used as operator consoles. The functions which the workstations run can be executed by a UNIX server which has multiple processors. The main network is an FDDI ring that has a node at each local control building; there will also be an Ethernet segment in each building. The equipment control layer consists of about 60 to 80 VME Input/Output Controllers (IOC) distributed around the KEKB rings. Each IOC will be allocated to a hardware group to avoid conflicts among groups or users. The final layer, the device interface layer, has standard modules such as CAMAC, GPIB, VXI, etc. There will be a special interface for controlling the magnet power supplies.

### *Presentation Layer*

OPI, the presentation layer, will include operators consoles, database manager, simulation computer, alarm generation/recording, data logging, display and a gateway to the KEK in-site network. There will be workstations running UNIX. X Windows, MS Windows and Windows NT are candidates for the operators console. The database manager computer will keep all the information concerning KEKB including machine parameters, equipment specifications, location, and so on. The simulation computer will be used for accelerator physics simulation calculations for such purposes as orbit correction. Faults in the equipment will be monitored by each IOC and reported to the alarm computer for broadcasting and recording purposes. The data logging computer will collect data from various equipment control computers for later analyses. The display computer will display data and transmit information over the KEKB site through the CATV network and other media. There will also be a gateway computer to connect the KEKB control computer network with the KEK laboratory network to make it possible for the accelerator department staff to reach KEKB collider equipment from their offices.

The software environment required for this layer is very important for the whole system. The fundamental functions: to display status and measured values graphically, to control each piece of equipment etc., should be supported from the start of installation of the machine equipment. The programmer-friendly interfaces must be provided for the accelerator physicists to make application programs easily. And it is also important to have a very efficient data retrieval and analysis system via the database.



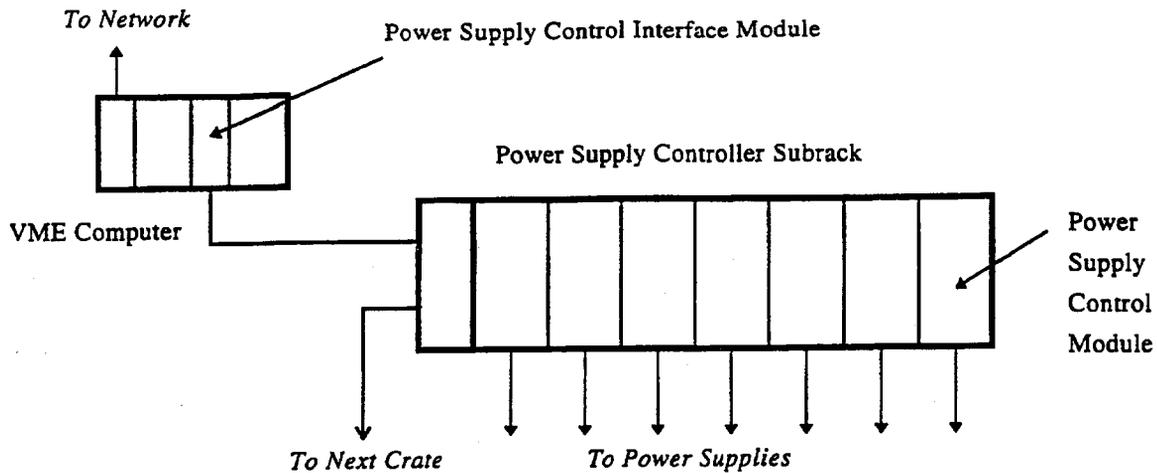
**Figure 1. System Architecture of KEKB Control System.**

### ***Equipment Control Layer***

The equipment control layer consists of IOCs that functionally control the equipment of each hardware group. Each IOC is a VME computer equipped with CAMAC serial highway drivers and other standard field-bus driver modules. The operating system for the equipment control computer is VxWorks. The type of processor on the VME processor board is not yet fixed and can be changed according to the requirements. The minimum requirement is that there must be a cross-development system which runs on a workstation.

### ***Device Interface Layer***

There will be several field-buses for the lowest device interface layer. There are CAMAC crates and CAMAC modules which can be re-used from TRISTAN. The CAMAC crates will be connected by CAMAC serial highways or CAMAC branch highways according to requirements. There will also be other standard field bus equipment such as GPIB. VXI modules will be used for the beam position monitors and fast signal measuring systems. There will be about 900 BPMs, and the electronics for them will be in the 20 sub-control rooms around the ring. Where the VXI system is used, it is planned to put a VME processor board into the VXI slot so that it can be used as an IOC. There are about 1900 beam steering magnet power supplies installed in the four power supply rooms around the ring. These power supplies will be controlled through a dedicated bus especially designed for them. A power supply control VME module will transmit parallel control signals and the bus adapters will drive power supply controller modules as shown in Fig. 2.



**Figure 2. Control Scheme of the Steering Magnet Power Supplies.**

## CONCLUSION

The KEKB control system is now in the final phase of design and the specifications are being defined. The first system for development will be installed early in 1997; the construction will start at that time and should be finished by the second quarter of 1998. It seems essential to use commercially available products based upon standards and to make use of software sharing to reduce the manpower and effort required. The system should be friendly to the accelerator physicists who create application programs. The database will become the central core of the KEKB control system and will make it possible to combine all the subsystems into one large system.

## REFERENCES

- [1] Shin-ichi Kurokawa, Status of TRISTAN-II Project, Proceedings of the 1993 Particle Accelerator Conf., pp. 2004-2006
- [2] Shin-ichi Kurokawa, et al., The TRISTAN Control System, Nucl. Instr. and Meth., A247, (1986) pp. 29-36.
- [3] L.R. Dalesio, et al., EPICS Architecture, ICALEPCS 91, KEK Proceedings 92-15, (1992) pp. 278-282.
- [4] T. Katoh, et al., "Control System Design for KEKB Accelerators", Proc. 1995 Particle Accelerator Conference and International Conference on High-Energy Accelerators, May 1 - 5, 1995, Dallas, Texas, U.S.A.

# Current Status of the Control System for the PLS 2 GeV Linac\*

I. S. Ko, J. H. Kim, S. C. Kim, J. M. Kim, G. S. Lee, and W. Namkung  
Pohang Accelerator Laboratory, POSTECH  
Pohang 790-784, Korea

## ABSTRACT

A distributed control system has been developed and successfully operated for the PLS 2-GeV linac. The system has three layers of hierarchy: operator interface computer, supervisory control computer (SCC) for data processing and device interface computer (DIC) for distributed data acquisition. The operator interface is based on a UNIX system with graphic-based development system, RTworks. The SCC consists of three subsystems: modulators, magnet power supplies and beam diagnostics. DICs attached to individual devices are placed around the 200 m long linac building. There are altogether 22 VME CPUs and various I/Os at the SCC and DIC levels. The realtime operating system is OS-9. During the preventive maintenance period in 1995, several upgrades have taken place. Beam loss monitors and beam current monitors have been added in the beam diagnostic station. We use fast sample and hold circuits to read 2 ns long pulse signals from the beam current monitors. We also report other upgrades and improvements that have been made.

## I. INTRODUCTION

The Pohang Accelerator Laboratory (PAL) completed the 2 GeV synchrotron radiation source named Pohang Light Source (PLS) at the end of 1994 [1]. The design and construction work had taken seven years. The PLS consists of a 2-GeV electron linear accelerator and a 2 GeV storage ring. Currently, there are two beamlines: one for vacuum ultra-violet (VUV) applications and one for X-ray applications. From September 1995, the PLS started its service as a low-emittance light source for various research projects in material science, surface physics, biology and semiconductor applications using X-ray lithography.

The 2 GeV linac has 11 klystrons and modulators on the ground floor and 42 accelerator sections in the tunnel. The overall length is 150 m and the average accelerating gradient is 15 ~ 20 MV/m. In order to achieve such a high accelerating gradient, 80-MW klystrons and SLAC-type RF pulse compressor systems (SLED) are used. There are also many magnet power supplies, vacuum monitors and various beam diagnostics devices. The circumference of the storage ring is 280.56 m. The storage ring lattice is based on the triplet-bend-achromat (TBA) with a 12-fold symmetry. There are 36 bending magnets, 144 quadrupoles and 48 sextupoles. There are three re-entrant type RF cavities with a 60 kW klystron for each. In order to correct the beam orbit, there are 108 beam position monitors and corrector magnets around the storage ring. There is a 96-m long beam transport line (BTL) to connect the linac and the storage ring.

Even though the control systems of the linac and the storage ring were developed by different teams, there are three common factors. Each system has a three-level hierarchy structure and a VME-based system with OS-9 operating system, and all custom-made codes are written in C.

The linac control system includes the main linac and the whole BTL except the Lambertson DC septum magnet which is located at the injection straight section of the storage ring. There are also two beam analyzing stations (BAS) to analyze the properties of electron beams up to 100 MeV and 2 GeV, respectively. The structure of control system shown in Fig. 1 was finalized by January 1993. At this stage, it was decided that we would use commercial products for all hardware such as CPU boards and I/O boards and concentrate our effort on developing necessary software with the help of a commercial development toolkit RTworks. Before starting the major work, we made the signal list and the design manual for the linac control system [2,3]. The actual software development started in May 1993 and the initial phase of the control system was completed by the end of June 1994. It played an important role during the commissioning of the 2 GeV linac, which started at the beginning of January 1994.

At present, the linac control system provides reliable service for daily operations. However it is continuously being upgraded, based on operational experiences and diagnostic equipment added.

---

\* Work supported by Pohang Iron & Steel Co. and Ministry of Science and Technology, Korea.

## II. HIERARCHY OF LINAC CONTROL SYSTEM

Our aim for the linac control system is to provide a reliable, fast-acting, distributed real-time system. As shown in Fig. 1, there are three layers in the control hierarchy; operator interface layer, data process layer and data acquisition layer. The last two layers are based on the VME realtime system. There are also three subsystems with different functional characteristics: modulators and the microwave system (MK), magnet power supplies (MG) and beam diagnostics (BM). Several special systems are connected directly to the data process layer. These various elements are linked by four independent ethernets.

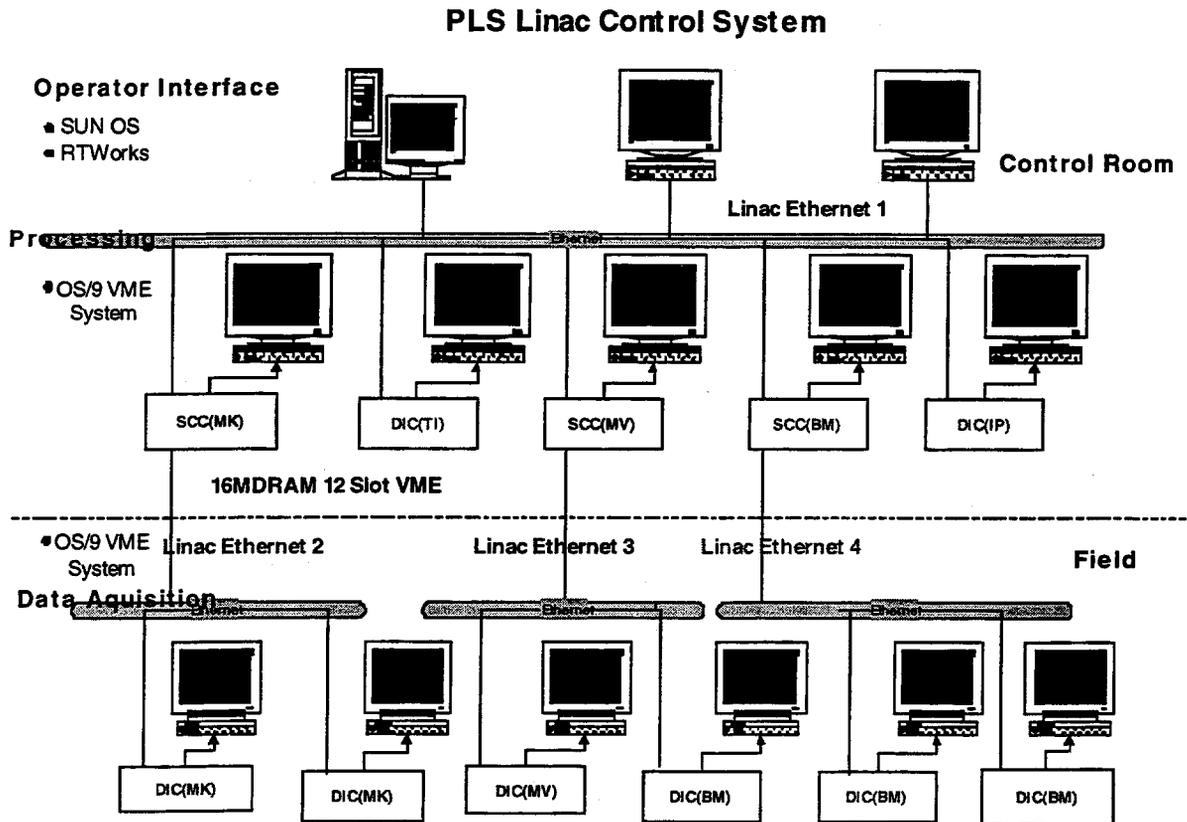


Fig. 1: Hardware structure of the PLS linac control system

### A. Device Interface Computer (DIC)

This layer is directly connected to the individual devices to be connected or monitored. The local computer connected to the individual devices is called the device interface computer. Each DIC is consisted of an ELTEC E-16 CPU board, a 14" EGA graphic monitor, a draw-type keyboard, and several I/O boards mounted on the standard 19" rack. The E-16 board includes a Motorola 68030 CPU, 4 MB of DRAM (MDRAM), an EGA compatible video port and an ethernet port. There is an extra memory board with 4 MDRAM in each DIC. The operating system is OS-9 with the MGR graphic development tool. On-demand local computer control is available to all DICs. This feature is extremely useful for the local commissioning and testing of an individual device, especially the 200 MW modulators.

There are eleven units for the modulator and microwave control systems (MK) and three units for magnet power supplies (MG). There are also two units for beam profile monitors and other diagnostic equipment in the linac and the BTL. All DICs are located in the klystron gallery.

All of the I/O boards used in the VME system are commercially available. Table 1 shows a summary of these I/Os currently used in the linac control system.

Table 1: Summary of I/O boards

Name	Functions	No *	Station
TSVME400	48 digital input (optically isolated)	13	MK, BM
TSVME401	32 relay-output	13	MK, BM
TSVME410	Multifunction	11	MK
TSVME500	4 x RS422	11	MG, Buncher
TSVME208	4 MDRAM (memory)	15	MK, MG, BM
AVME9470	80 I/O channels	10	MK (IPA)
AVME335	Image Processing	1	IP
E-16 GPIB	GPIB port	11	MK (oscilloscope)

(\* Number of used I/Os as of September 1, 1995)

### B. Supervisory Control Computer (SCC)

In order to avoid a heavy workload on the main computer, we divided the linac control system into the three functional subsystems noted above. The role of the SCCs is to act as an intermediate data manager for the assigned subsystem.

Each SCC unit consists of an ELTEC E-7 CPU board, a 19" monitor located on the sub-control console, a 3.5" floppy disk and a 900 MB hard disk. The E-7 board has a Motorola 68040 CPU, 16 MDRAM and a workstation graphic board. There are two ethernet ports in each SCC; one for the data acquisition layer and one for the operator interface layer. Some units are equipped with a streaming tape backup system. All SCCs are installed in one standard 19" rack in the sub-control room which is situated next to the main control room, only separated by large glass windows.

Three more such units are installed in another standard rack in the same room. Their major role is to allow the development of system software without disturbing the main control system; they can also provide backup functions for the main SCCs in case of troubles.

### C. Operator Interface Computer (OIC)

This is the main computer for the PLS linac control system. The OICs are actually a SUN-4 sparcstation and two X-terminals connected to this SUN workstation. They are located on the main console. There is one more system in the sub-control room, namely the backup system. The OS is UNIX, and RTworks is intensively used to optimize graphics and data handling between the UNIX system and the OS-9 system.

There is one more SUN-4 workstation in the main control room for physics-related work. The accelerator physicists can develop various simulation codes and apply the results to operations after development.

## III. SOFTWARE

In parallel with the hardware structure, we use three layers in the software structure. Subsystems such as MK, MG, and BM are monitored and controlled by individual tasks running on the appropriate layers as shown in Fig. 2. There are two important features in the linac control software; the client/server routines to exchange data and commands between layers and the separation of monitor/control tasks. When the operator selects a command from an appropriate window, it sends it down to the designated device via control client/server routines in the realtime system. The data collected by a given command returns to the operator's window through a separate route. The communication between layers is made using the TCP/IP protocol.

## PLS Linac Control System S/W Structure

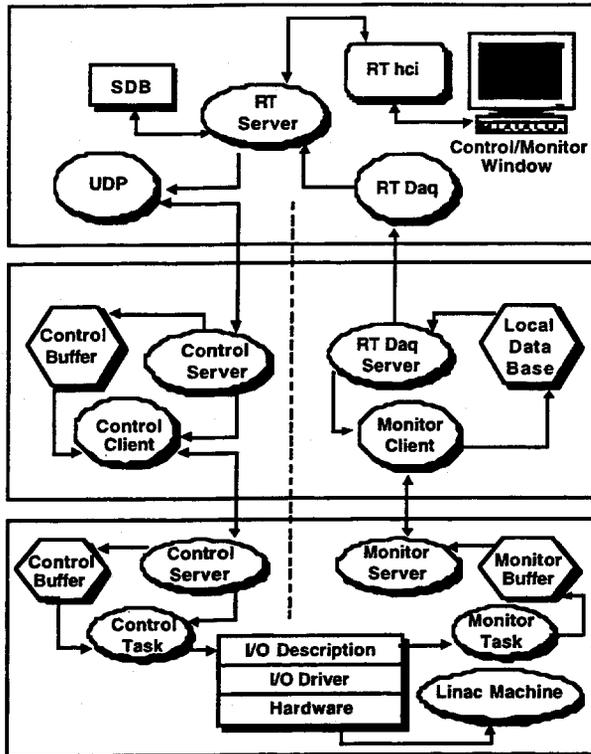


Fig. 2: Software structure of the PLS linac control system

### A. Data Acquisition Layer

The major roles of the data acquisition task in the DICs are as follows; to store monitored data from individual devices to the designated buffer (m\_buffer) regularly and to send updated data to the SCC upon request. On receiving a control command from the SCC, the corresponding control task sends data or messages to control individual devices in realtime through the appropriate I/O ports. Other important roles of the data acquisition task are to monitor any malfunction of the connected devices and to provide emergency cures if possible as well as to report the status to the SCC.

### B. Data Processing Layer

The major role of the data process task is to collect the updated data from the attached DICs and to store them in the realtime database. These data are sent to the RTdaq running on the OIC upon requests made by the operator. Also a corresponding task sends control commands to the appropriate DIC in real time. All the required application software is running in real time in this layer including client/server routines for the ethernet communications. Every piece of data in the linac operation is stored in the RAM area temporarily and transferred to the disk for permanent record.

### C. Operator Interface Layer

A commercial software package named RTworks is being used to develop graphic-based operator interfaces. RTworks is actually an integrated development toolkit for data handling between client/server systems. Using this toolkit, it was possible for one software engineer to develop all the operator interfaces required for linac control in a year.

The data received in the RTdaq from realtime CPUs are sent to the RTserver by inter-processor communications and they are displayed in the monitor windows by the RThci routine. A command selected by a mouse action is sent to the SCC through user-defined processes.

There are several windows for operators to control and monitor individual system such as microwave, bunchers, IPAs (isolator, phase shifter and attenuator systems to provide the best microwave conditions for the klystrons) and modulators. Each window has a value display area and a control sub-window. All the control actions can be made by selecting areas with a mouse or selecting items from pull-down menus. Figs. 3 shows the main window for the linac control system. This window includes the status of 11 modulators and phase angles for IPAs. Subsystems such as magnet control and monitoring windows can be started by selecting appropriate commands from the pull-down menu.

## IV. UPGRADE PLAN

As of the end of June 1994, most of the linac control system was completed. It was used for the commissioning to obtain 2 GeV beams and for the routine operations to provide beams to the storage ring [4]. Since then, only minor debugging work has been necessary. However, there are several things to be added to the present control system. Signals from beam current monitors (BCM) and beam loss monitors (BLM) will be displayed in the main control window. This work involves fast signal processing due to the 1 ns pulse duration of the electron beam. At present the necessary electronics, including fast sample/hold circuits for the BCM are completed. Full tests of the BCM and BLM electronics and data acquisition routines will be made in the next preventive maintenance period which is scheduled for summer 1996.

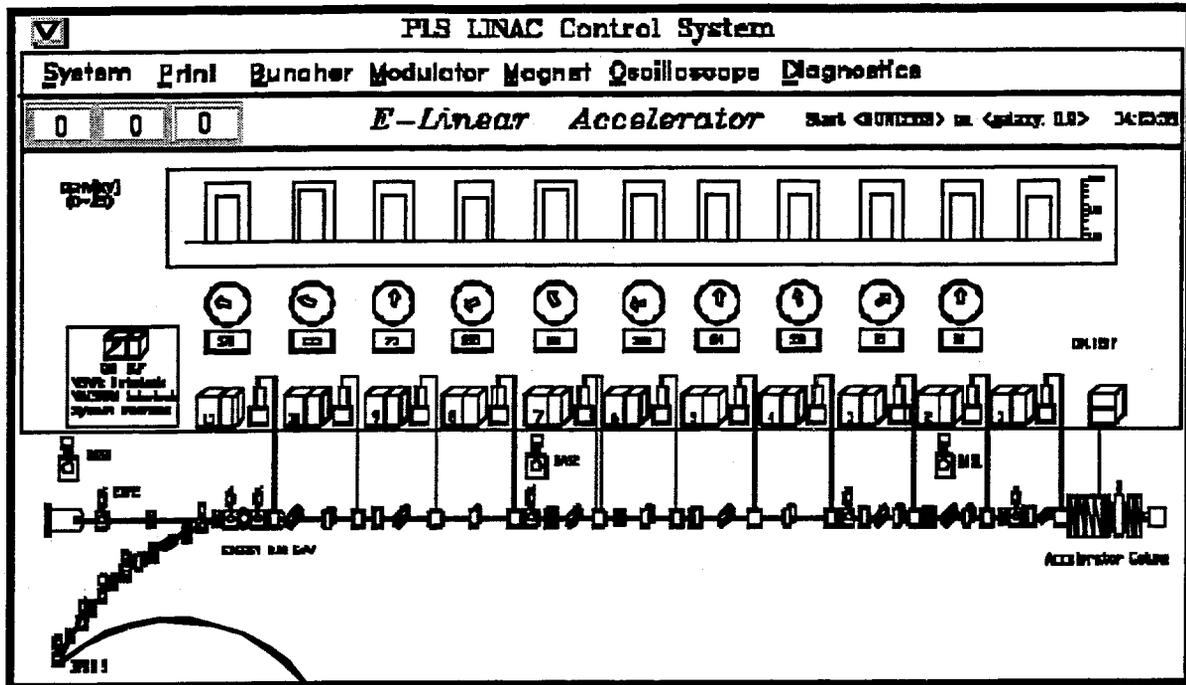


Fig. 3: Main window for the PLS linac control system

The timing system will be incorporated in the main control system after completing optical communication circuits between the individual time delay units located in the sub-control room and the corresponding modulators in the klystron gallery. There will also be a new electron gun control system.

## V. ACKNOWLEDGMENTS

Our grateful thanks to the team of linac operators, especially to M. K. Kim, K. W. Kim, N. S. Sung, H. S. Kang, and S. W. Kang for their valuable help and suggestions in developing the linac control system.

## VI. REFERENCES

1. "Design Report of Pohang Light Source (revised ed.)," Pohang Accelerator Laboratory, 1992.
2. I. S. Ko and W. Namkung, "Signal List for 2-GeV Linac," MA/LN-93001, Pohang Accelerator Laboratory, 1993.
3. I. S. Ko et. al., "Design Manual for PLS Linac Control System," MA/LN-93002, Pohang Accelerator Laboratory, 1993.
4. W. Namkung et. al., "Operation of PLS 2-GeV Linac," in Proc. of 1995 Particle Accelerator Conference, Dallas, Texas, USA, May 1995 (to be published).

# Control system of laser for plasma diagnostics.

I.M. Koltsov, A.A. Komarov, V.M. Rybin, A.A. Jelezin

The Moscow State Engineering Physics Institute

( Technical University ), Russia

Modern methods of diagnostics and research require laser radiation with many different parameters. One of the tasks that arose during creation of a complex for plasma diagnostics was the development of a control system for the radiation parameters of a tunable laser.

The control system for lasers was developed at our institute using a gauge to measure the radiation wavelength, a module for data acquisition and control and a module for data processing.

The module for data processing is a microprocessor module. Despite its simplicity it completely fulfills the technical requirements and can execute complicated operations of processing, converting and transferring data. The functional chart of this module is shown in figure 1.

The Z80 CPU provides the required mathematical calculations for the radiation parameters and generates commands for the control module. The 8751 micro controller is used for display, keyboard and serial input/output control, and contains a FIFO that receives and transmits data. The serial port can be connected to an IBM PC to be used as an operator workstation.

The distinct feature of the control module is the standard parallel interface, similar to the IBM PC printer interface. It allows one to connect one or several such modules directly to the IBM PC when necessary. The data exchange with the module is made by writing standard commands to the module registers and reading data from the registers. The exchange is done by means of 8-bit words.

This approach allows one to change the configuration of the system and to develop it step by step. The functions of the system are as follows:

- Analysis of the state of the laser plant
- Conversion of signals from the gauge and their analysis
- Controlling of operation modes of executive devices

The source of information is the wavelength gauge. Its optical system is shown figure 2. The radiation from a source 1 passes through a system of lenses 2 and through the Fabry-Perot interferometer 3. At the plane 4 an interference pattern consisting of light and dark circles appears. The coordinate sensitive gauge 7 mounted on transport platform 5 scans the interference pattern and gets information about circles. The platform 5 is moved by step driver 6.

Four identical zones are put on a uniform crystal substrate for the light sensitive element of gauge 7. Every zone generates an electrical signal proportional to the radiation exposure. The gauge forms a difference signal from two zones. It allows one to reduce the undesired influence of external light on the measurements.

For the definition of the length of the wave it's necessary to know two parameters; geometrical sizes of the optical system and diameters of the interference circles. Before the measurements with a tunable laser a calibration of the optical system needs to be conducted. For this the radiation of a stabilized laser is put into the optical system. Knowing the wavelength of the "reference" laser and the diameters of interference circles of the "reference" radiation and of the radiation of the tunable laser, one can calculate the wavelength.

The control software of the system was written for the IBM PC and does all functions mentioned below:

- provides automatic setup of system with "reference" and required wavelength
- supports switching of scanning modes
- makes periodic correction of tunable laser
- has a user interface

Experiments with the system have shown stable work with lasers with a range of tuning from 0.64 to 1.1 microns. The minimum period of scanning is from 50 seconds to 2 minutes in continuous scanning mode.

## Literature.

1. A.A.Jelezin, I.M.Koltsov, A.A.Komarov, V.M.Rybin, S.S.Sobolev, Information measuring system for LIDAR, magazine "Measuring technique", N 11, 1995, p.12-23.

2. I.M.Koltsov, L.A.Markova, U.P.Sidorenko, S.S.Sobolev, S.S.Timofeev, magazine "Applied Spectroscopy", N5, 1991, p860-865.

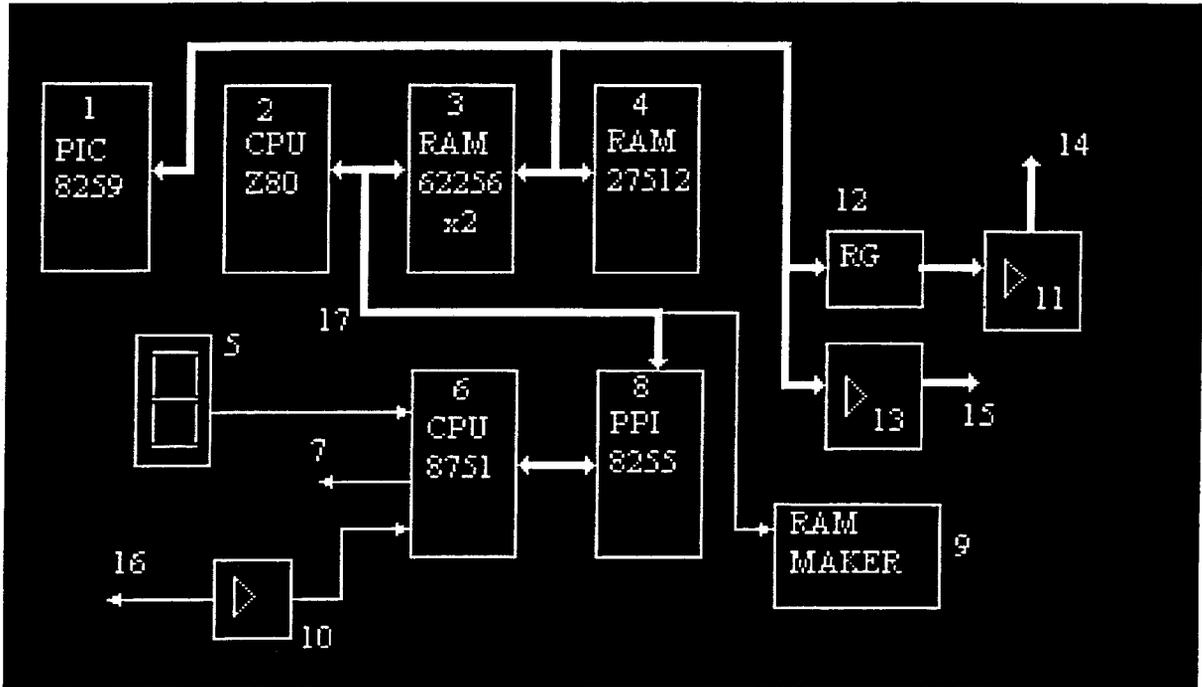


Fig. 1 The functional chart of data processing module:

1 - interrupt controller, 2 - central microprocessor, 3 - RAM, 4 - ROM, 5 - display, 6 - peripheral microcontroller, 7 - keyboard connector, 8 - peripheral adapter, 9 - memory controller, 10 - buffers of serial port, 11 - parallel port interface, 12 - parallel port registers, 13 - external bus interface, 14 - parallel port output, 16 - serial port output, 17 - internal bus.

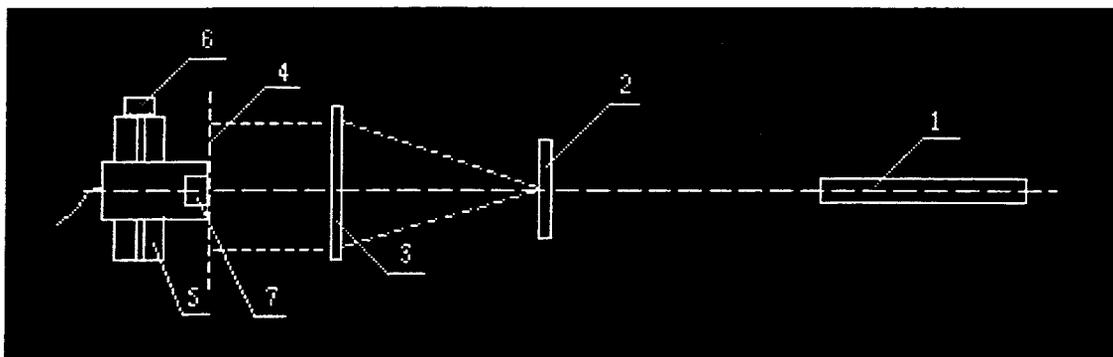


Fig. 2 The function chart of the wave length gauge:

1 - source of radiation, 2 - system of lenses, 3 - interferometer Fabri-Pero, 4 - plain of interference picture, 5 - transport platform, 6 - step driver, 7 - light-sensitive element.

# Control System of SR Source SIBERIA Commissioning and Future Plans

V. Fransev, V. Goluzov, A. Kadnikov, Y. Krylov, S. Kuznetsov, A. Valentinov, Y. Ypinov

Kurchatov Institute, 123182 Moscow, Russia

## *Abstract*

The SIBERIA SR complex, which has been under construction for the past few years, has become operational. The facility has been successfully commissioned using a control system. The paper describes the architecture, the techniques used and the commissioning experience. An implementation of UNIX/MS-Windows compatible software tools, such as LabView, for accelerator control is presented. This paper also reports the hardware-software solutions for the automatic process sequence of injection, accumulation and energy ramping. The hardware approach for the temperature monitoring, timing system, feedback control and turn-by-turn beam diagnostics is presented.

## 1. INTRODUCTION

The SIBERIA accelerator complex includes an 80 MeV electron linac, a 450 MeV booster storage ring, a 2.5 GeV main storage ring and two transport lines for electron beams. It that was developed by the Budker Institute of Nuclear Physics (Novosibirsk) for the Kurchatov Institute (Moscow)[1]. During the first months of 1995 the main ring was commissioned. The ramping of energy in the main ring is the goal for the next stage.

## 2. COMMISSIONING RESULTS.

### *2.1 Hardware*

The control system is separated into three layers. A PC network provides the upper level for use in the control room. Four IBM PC 486 computers with 19" color monitors and 8 MB of RAM form the consoles, running MS-Windows for WorkGroups 3.11. Tasks may be interchanged between computers using Network Dynamic Data Exchange (NetDDE). A single PC with a 50 MB Hard Disk forms the interconnection between the consoles and the second real-time layer.

The second layer is composed of eight 24-bit CAMAC-oriented diskless computers, connected to the PC network by 1 Mb/s serial links. They run the real-time multitasking ODOS operating system and are equipped with RS-232 interfaces for local terminals, RAM, network interface, graphic adapter and peripheral crate driver. Each computer controls a specific part of the accelerator (injection, main ring) or a specific system (vacuum, beam monitoring, temperature measurement, etc.)

The third layer consists of 36 CAMAC crates fitted with a total of 385 I/O modules of various kinds. These peripheral crates are placed near the power supplies and the RF and vacuum equipment. They are connected to the control computers by 1 Mb/s serial links on coaxial cables. The distance between crates is 50 to 100 m.

An important feature is the synchronization between the main ring RF, the booster RF and the fast deflectors for injection into the main ring. Multi-channel programmable CAMAC timing generators with a 0.4 ns resolution and a maximum delay of 5.2  $\mu$ s are used for frequency matching and selection of the number of bunches. The operator can select the bunches, control the phase of injection and measure the time delay with an accuracy of 1 ns. The RF control includes 16 DAC channels for cavity voltage control, feeder current setting and revolution frequency control.

The control computer supplies the control and feedback functions for all except the temperature monitoring system. These are based on an intelligent ADC module, which contains a CPU, RAM and program ROM. It has four relay outputs for the interlock system of the power supplies. The main features of the system are:

- all channels are sampled within a 1 to 5s cycle time
- acquired data are compared with stored warning and alarm levels
- alarm devices and interlocks are activated as required
- a graphic presentation is made of the status and history of selected channels.

A set of 12 movable scintillation probes is used for the first turn diagnostics, the position of each probe being measured to an accuracy of 0.2 mm, using an ADC. The main ring is equipped with 24 horizontal and 24 vertical BPMs, connected to a CAMAC set which includes timer, integrated ADC and amplifier/filter modules. Four pick-ups with special fast 8-bit ADCs can take and store one sample per turn (400 ns) for the first 1000 turns, allowing optimization of the injection into the main ring.

## 2.2 Software

The most important concept in the software design is that of a "regime" of the storage ring. A regime represents a collection of settings for the control channels; each may be saved in a file and edited by a special program. The ramping process consists of loading a set of regimes for different energies sequentially. During transition from one regime to another the settings in control channels change linearly. If any channel is not present in given regime the set for this channel is not updated during regime loading.

The program PRCS controls the real-time process for changing the facility mode (for example beam acceleration). This program includes following functions: setting the time program, external trap excitation and checking and setting the facility mode in the required number of steps. The PRCS task gives a command to the BANK program for execution. At present there are the following working cycles of the accelerator facility: 1) injection in SIBERIA-1 and beam storage, 2) acceleration to 140 MeV in 50 steps, 3) acceleration to 240 MeV in 100 steps, 4) acceleration to 450 MeV in 100 steps, 5) beam extraction from the SIBERIA-1 and injection to SIBERIA-2, 6) return to injection mode in 300 steps.

The automatic procedure for injection into the main ring SIBERIA-2 was proposed and realized. This approach was based on using the beam-current measurement for feedback. The PRCS program implements injection into SIBERIA-1 until achievement of the setting value of beam current. During this procedure the program compares the set point for the beam current and the measured value from a DCCT. Then it transfers to acceleration in SIBERIA-1 and ramping the energy up to 450 MeV over 30 sec. After that the PRCS automatically generates a sequence of commands for extraction from SIBERIA-1 to the main ring.

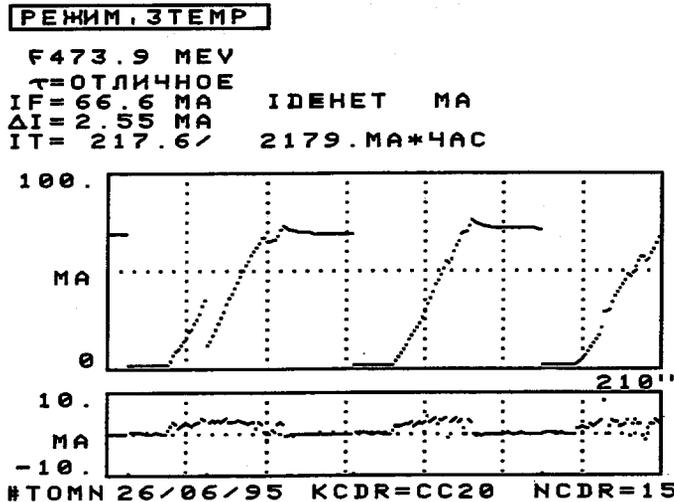


Figure 1. Injection complex automatic regime illustration.

The time of the injection cycle in this case is approximately 60 sec with a SIBERIA-1 beam current of about 80 - 100 mA. In this case the operators and physicist team can control and study the main ring injection without control commands to the injection complex. The operator can change the level of demanded beam current or interrupt the process and come back to manual mode. Figure 1 illustrates this automatic regime of the injection complex. The first graph shows beam current versus time, the full scale of time being 210 sec. The second graph displays addition of beam current each injection cycle into SIBERIA-1. The beam current falls to zero after extraction to SIBERIA-2.

### 3. NEW PROJECTS

#### 3.1 Local-global digital feedback system for orbit stabilization

The typical beam stability tolerances for modern light sources is about 10% of the beam size and divergence. For SIBERIA-2 this criterion means that we have to position the beam orbit accurate to the 10-15  $\mu\text{m}$  level. Active feedback systems, which detect and counteract undesired beam fluctuations by correcting the closed orbit, are essential to achieve the necessary beam stability. Potential sources driving the electron beam fluctuations are ground vibrations, magnet power supply ripple and thermal drift. A specific feature of the low-emittance light source magnetic lattice is the significant amplification factor for any mechanical vibrations of the magnetic elements due to the strong focusing. In the case of SIBERIA-2 any quadrupole lens motion will transfer to beam motion with the amplification factor (rms)  $M_{x,z} \approx 30$ . To stabilize the orbit oscillations a fast digital feedback system has been proposed [2]. System realization will include a local digital feedback system for single-point orbit stabilization at the first stage. Then we plan to implement several local systems for beam stabilization at insertion device points. The global digital feedback system will provide orbit stabilization at the 10-15  $\mu\text{m}$  level in the 0.5-100 Hz frequency range.

#### 3.2 LabVIEW to OS-9 software interface

The LabVIEW system is very popular for instrumentation applications. We tried to connect a powerful LabVIEW graphical interface to real time applications. The hardware is based on a standard Ethernet network. The structure of the complex is shown in figure 2.

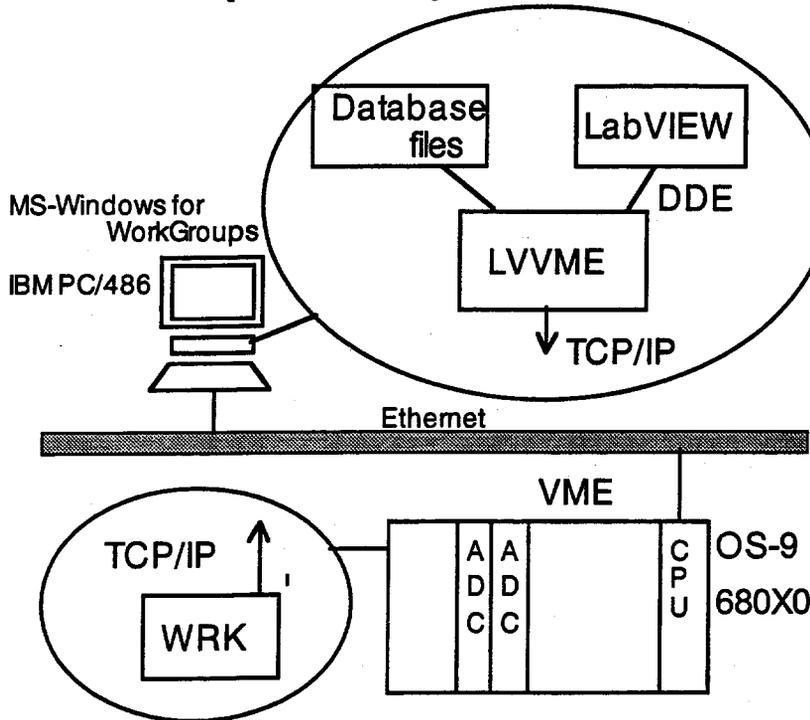


Figure 2. LabVIEW to OS-9 task connection.

The high level software runs on a PC and includes the LabVIEW application and dedicated software server named "LVVME". This server is a MS-Windows application and provides an interface between LabVIEW applications, data base and data-acquisition programs. Intertask communication uses standard MS-Windows Dynamic Data Exchange (DDE) protocol. Connection to the real-time module is based on our "ELib" library. The PC-level software is shown in figure 3.

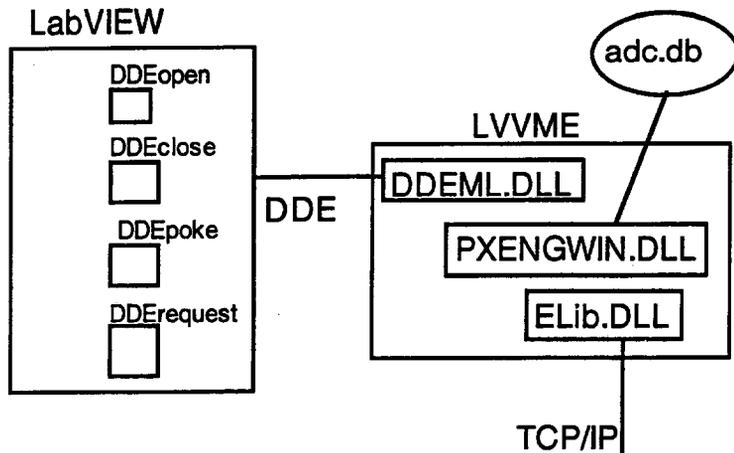


Figure 3. Structure of the LVVME server

For the network interface the special "ELib" library was developed. The library is based on Windows Sockets under MS-Windows and on Internet Support Package under OS-9. The main functions and advantages of the library are:

- asynchronous intertask connection;
- asynchronous data exchange;
- the same program interface under MS-Windows and under OS-9;
- compact (11 functions);
- intertask synchronization;
- TCP/IP implementation;
- one can connect socket and asynchronous event functions.

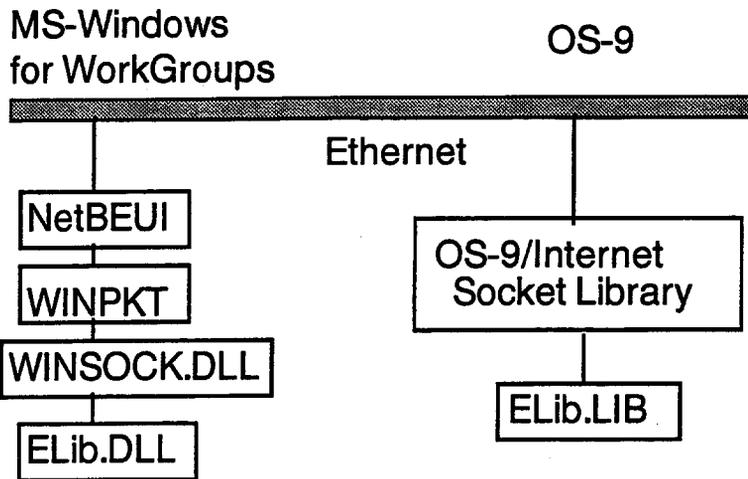


Figure 4. "ELib" up-socket library.

### 3.3 C++ Libraries

We have developed an object-oriented library for database access and connection between high-level applications and real-time tasks. The database describes the control system of the accelerator complex and consists of a main file and channel families' description files. Each record in the main file includes the name of the channel family, the name of the description file and comments. Hence the number of records is equal to the number of channel families. Family description files describe channels of the same type. For instance, the family of control channels for power supplies, the family of monitoring channels for temperature sensors, the family of measuring channels for digital oscilloscopes, etc. The channel description represents a record in the corresponding file, which contains static information -- names, coefficients, hardware addresses, comments, etc. Channels of different types can have records of different structures. The number of records in a family description file is equal to the number of

channels of this type. Channels have not just to be linked to hardware channels of ADC or DAC, but can also be designed to transmit instructions for a real-time operating system or to produce some mathematical calculations etc. For example, the BeamEnergy channel can be a result of processing the currents in bending magnets. Database access and the mapping of an object oriented library in computer memory has been developed.

Class TDataBase represents the concept of a database as a container of channel families in the form of an array. It also encapsulates methods for database reading from persistent memory and the construction of data members. Similarly, TFamily implements the concept of family as an array of channels. TChannel contains an array of dynamic data (TData) and static data (TStaticData) from the database. TData and TStatData are abstract classes, which include all the necessary methods for information treatment. They implement methods suitable for data of any type. Methods that are different for different data types are presented as pure virtual functions. Classes derived from TData or TStaticData define the implementation of these methods. These are methods of interpretation of database information, treatment of information from the low level control system, execution of requests from console level programs and methods of graphical representation of data.

The library for network and internet exchanges contains classes for communication to control the computer network. TExchange implements the hardware level of information transmission and decoding. TZone executes the link between particular real-time control programs and channels (TChannel). TSocket connects a console level application with the database server program.

Another example of successful implementation of an object-oriented approach is the library for on-line modeling [3]. This library is placed between primitive control channels and complicated 'machine' or 'beam' channels. A class library for on-line modeling is portable and uses the standard MAD input file interface for machine description. The object-oriented conception enables us to realize suitable parent-child class trees for control in "beam" terms.

#### 4. REFERENCES

- [1] V.N. Korchuganov et al., NIM 208 (1983), pp.11-18.
- [2] A.B. atrakov et al., "Digital Feedback System for Orbit Stabilization at the SIBERIA-2 Light Source", these Proceedings
- [3] S. Kuznetsov "C++ Library for Accelerator Control and On-Line Modeling", these Proceedings.

# DISTRIBUTED MEMORY IN A HETEROGENEOUS NETWORK, AS USED IN THE CERN. PS-COMPLEX TIMING SYSTEM

V. Kovaltsov<sup>1</sup>, J. Lewis

PS Division, CERN, CH-1211 Geneva 23, Switzerland

## ABSTRACT

The Distributed Table Manager (DTM) is a fast and efficient utility for distributing named binary data structures called *Tables*, of arbitrary size and structure, around a heterogeneous network of computers to a set of registered clients. The Tables are transmitted over a UDP network between DTM servers in *network format*, where the servers perform the conversions to and from *host format* for local clients. The servers provide clients with synchronization mechanisms, a choice of network data flows, and table options such as *keeping table disc copies*, *shared memory or heap memory table allocation*, *table read/write permissions*, and *table subnet broadcasting*. DTM has been designed to be easily maintainable and to recover automatically from the type of errors typically encountered in a large control system network.

The DTM system is based on a three-level server daemon hierarchy, in which an inter daemon protocol handles network failures and incorporates recovery procedures which will guarantee table consistency when communications are re-established. These protocols are implemented over a communications layer which performs the data conversions, packet splitting, error-correction with retry and timeout handling. The same communications layer is used to implement the application program interface which calls on the server daemon for client services. DTM is a registration based system, in which communications are established dynamically as needed and tables are distributed only to the clients who have registered their interest in them. The registration protocols include mechanisms to recover from daemon re-launches and to clean up after aborted clients.

## 1. INTRODUCTION

The PS Complex contains nine accelerators which interact in a time-sliced manner to produce particle beams varying in particle type, energy and time structure. The timing and sequencing of the accelerators in the network is controlled through a Master Timing Generator (MTG) which distributes events over a specialized timing network to the nine accelerator control subsystems [1]; some of these events are also distributed over the computer network. DTM complements the standard RPC-based PS equipment access, by providing *notify-on-change* events and *data-transfer* mechanisms, which are needed in any large control network and in particular by timing and sequencing mechanisms.

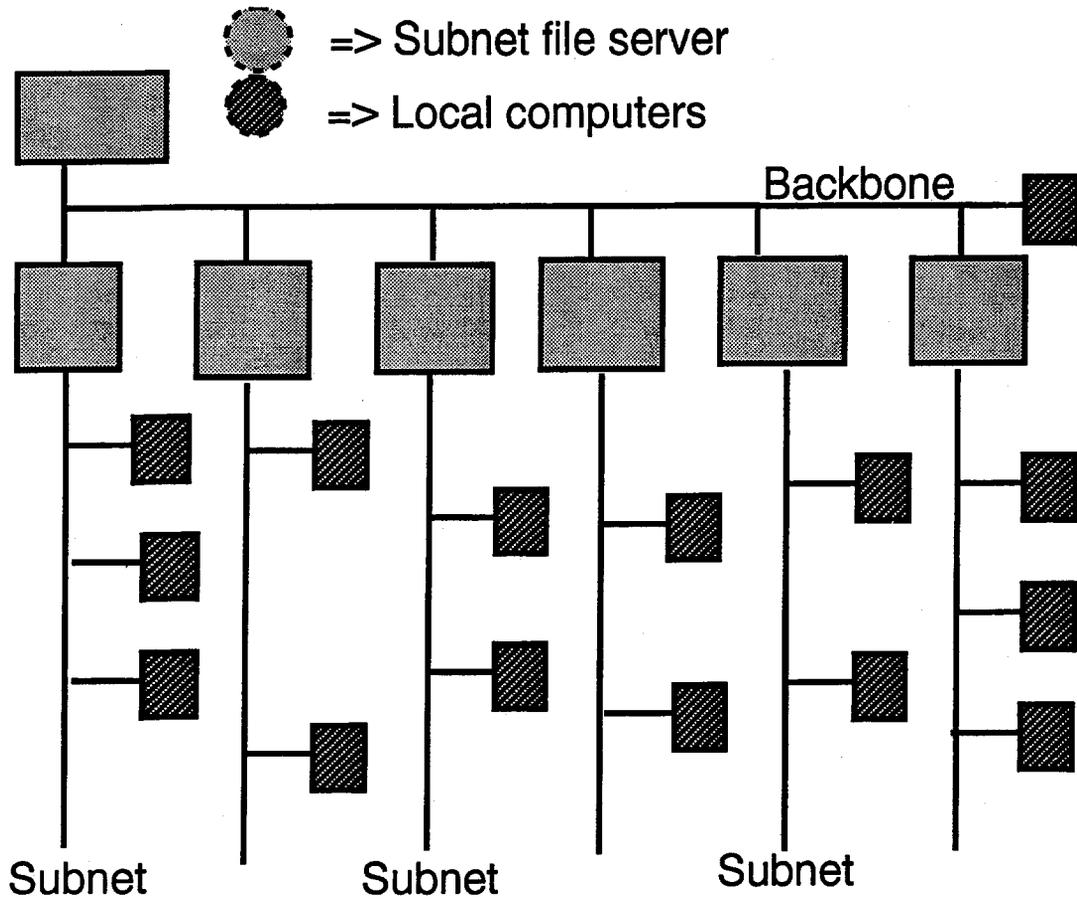
The *PS-Controls* general purpose computer network [Fig. 1], contains an isolated subnet for each accelerator and one main control system backbone which inter-connects them [2]. Each subnet contains its own file server, workstations and diskless VME-based front end processors and each must be able to function independently with its link to the main backbone cut. This network topology and subnet isolation requirement implies multiple copies of the common data needed to run each subnet independently and hence a mechanism capable of tolerating network failures, which correctly distributes pending data modifications when the fault disappears.

The implementation chosen for the timing system and its application program interface, implies almost instantaneous access to a number of common volatile data structures which are distributed over the network to each client every time they are modified. This access may be synchronous, meaning that a table update provides not only the new contents of the table, but also the event which signals the time it happened. In the PS timing system, such events can be used to transmit accelerator timings to application programs, so events are often more important than the new table contents.

---

<sup>1</sup> On leave from IHEP Protvino, Russia

Figure-01 Network layout

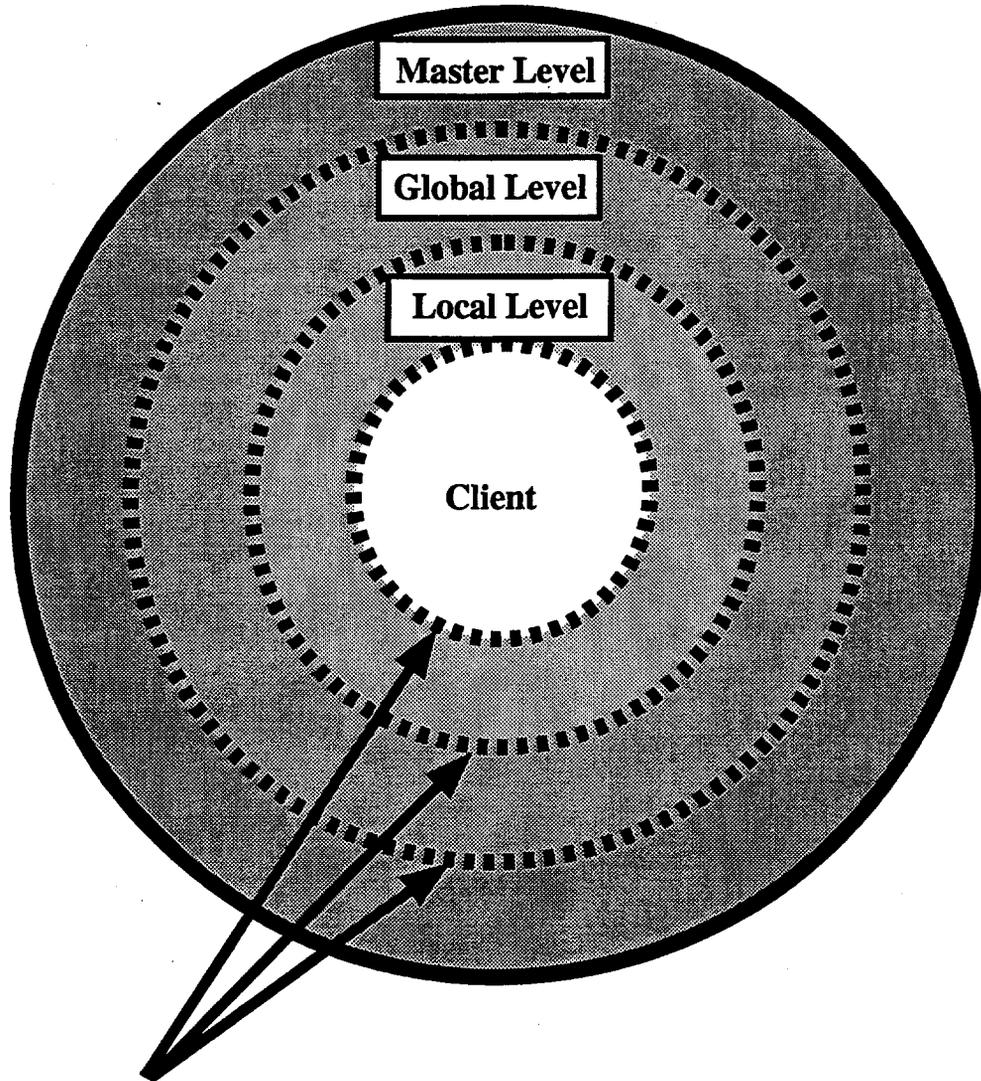


The concept of distributed memory satisfies these requirements, in which all copies of a memory segment residing on a set of *registered* hosts contain the same data and hence all writes into this memory space must be distributed across all copies. In DTM, this is a *One-To-Many* data flow, where a single client non-explicitly provokes a table change on all other registered clients in a heterogeneous network. This is not the only data flow; in *Many-To-Many* the consuming clients sequentially processes a queue of incoming new table instances. In a third data flow, one client explicitly reads another client's copy of a non-distributed table, this *One-To-One* flow has some similarities with a classic remote-procedure-call mechanism. All three of these mechanisms are used by the PS timing system and DTM has greatly simplified the task of implementing them, by providing applications with a fast platform-independent method for obtaining the data they need. Application programs are relieved of the burden of initializing and maintaining up-to-date copies of volatile data and of knowing who else is using it. No explicit inter-client actions are required, as all data is distributed between them by the DTM system. System maintenance is easier because of the ease with which data-driven programs can be built and the ease with which they can be distributed over a heterogeneous network. Lastly, a faster and more efficient use of limited network resources can be achieved by avoiding unnecessary transactions such as polling.

## 2. THE IMPLEMENTATION

The DTM system is built on a hierarchy of three classes of server daemon [Fig. 2]; a unique master server runs on the backbone subnet, one global server runs on each control subnet and one local server runs on all other hosts. Except when a network broadcast is used, all DTM data tables flowing between clients pass via the master daemon along the path shown in Fig. 3.

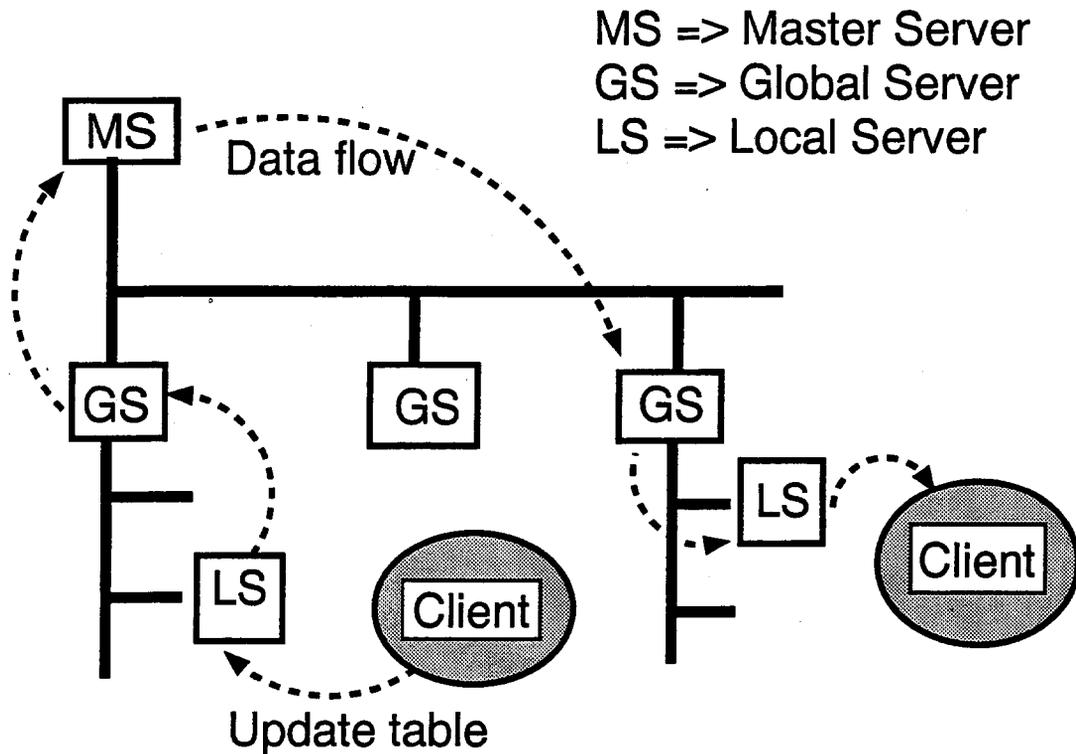
Figure-02 Daemon and client communications hierarchy



**These communication layers export the services provided by the next level up to the network, making all internals available for diagnostic tools. Local function calls are used between the levels in a single server daemon.**

The class hierarchy starts with the local server, which provides the applications program interface, and whose functions are available on all hosts. Next is the global server which inherits the local server functions and provides specific global functions. The unique master server inherits all local and global functions and provides specific master functions. Thus only one server runs on any given host. All communications between class levels and between the client and server are available across a communications layer which renders all such transactions independent of their location on the network. This approach has greatly eased the task of building diagnostic tools, which are able to communicate with any level of any daemon anywhere in the network. The current implementation makes use of UDP and UNIX domain sockets for inter host and inter task transactions. All daemon launches are triggered automatically in the standard way by *inetd*. Packet splitting, sequencing and error correction with retry are handled and connections are dropped and re-established between daemons using time-outs and an *I-am-alive* protocol. When a control subnet is isolated from the backbone, all DTM table updates to or from that subnet will be completed later automatically, when the inter server communications are re-established via the *I-am-alive* protocol.

Figure-03 DTM server daemon data flow



For ease of maintenance, the only system knowledge given to a server on start up is the directory path name: */usr/dtm*, where it finds all the information it needs in two configuration files. This directory is usually an NFS mount point for local server hosts on their global server and, besides the configuration files, it also contains disk copies of tables in network format and client registration tables. The two ASCII configuration files describe the attributes of each table and the host names of the subnet global server and master server respectively. The attributes of a table are as follows:-

- The tables name: E.g. CPS\_LINE\_NAMES.
- The tables format specification: E.g. 1L32{1L24{9C51C}}.
- The table producer host list: E.g. PsStation1, PsServer19.
- The table properties: E.g. PROD+NONC

The table name is any string by which the table is identified, and provides the binding between DTM servers for its transfer in the network. The table format string describes the table structure. The above example means: "1 Long, 32 Structures of { 1 Long 24 Structures of { 9 Characters, 51 Characters } }". The producer list contains the list of host names permitted to update the table. The table properties are any coherent subset of the following:-

- NONC Not kept on disk, so the table exists only in memory. This property concerns tables which are updated frequently. Disk-kept tables are only useful when recovering from a general power failure, or when bringing up a subnet in isolation, as in all other cases the latest table contents are transferred between servers when communications are re-established via the *I-am-alive* protocol..
- LOCL The table exists in local heap memory and not in shared memory. In this case the table memory is allocated to the client via the application program interface library.

- **PROD** The table may only be updated by a host in the producer list. Any write attempts by clients not on the producer list result in an error return from the library.
- **MCOP** The table has multiple different values. Used in *Many-To-Many* and *One-To-One*.
- **FIFO, FILO, LIFO ...** Various table queue disciplines for *Many-To-Many*.
- **BRDC** The new table contents is broadcast, rather than using inter-server point-to-point transfers. A UDP type-C subnet address is currently used, but for hosts outside the type-C subnet, the table is transferred in the usual way.

### 3. THE APPLICATION PROGRAM INTERFACE

The entire DTM application program interface is described by 10 simple function calls. All other daemon inter-level function calls are also available, but they are used exclusively for system internals and specialized diagnostic programs and are not listed here.

#### 3.1 Basic One-To-Many non-synchronous access

For the more common *One-To-Many* data flow the following five very simple functions provide an elegant way for an application to use non synchronized access to distributed shared memory.

- **int DtmrtRegisterTable(table\_name)**

A client calls this function when it wants to register its interest in a table. Two registrations are actually made by the system, a global registration saying that this host is using the specified table and a local registration saying that this task PID uses the table. Hence for any given table, there is one global registration per host using it and one local registration for each PID on that host using it. When the last local registration is removed, the global registration is also removed and all system resources are de-allocated.

- **int DtmrtUnregisterTable(table\_name)**

A client may call this function if it is no longer interested in a table. The local DTM server in any case checks that the PID exists each second and will clean up its registrations within this time if the task stops running for any reason. The only use of this function is thus in limiting the total resources consumed by a client at any one time, in particular the shared memory attach count.

- **table \* DtmrtShareTable(table\_name)**

This function returns a pointer to the registered table either located in shared memory, or on the local heap allocated within the function. Typically a client will cast the returned pointer into a defined data structure; for reading it can be directly accessed and for writing, if it is located in shared memory, a copy should be made because DTM can update it at any time.

- **int DtmrtUpdateTable(table\_name, new\_table)**

If this host has the correct write permission, calling this function pushes the new table contents out to all registered clients on the network and overwrites the local copy.

- **int DtmrtCheckTable(table\_name, callback)**

This routine can be called periodically from an application program main loop. If the table has been updated, the callback is invoked with the name of the table as its parameter. In this way an application is able to take specific action each time the table is changed; this is the non-synchronized access method. If the table name is NULL the callback is called once for each updated table.

### 3.2 Other data flows *Many-To-Many* and *One-To-One*

The next two function calls provide support for the *Many-To-One* and *One-To-One* types of data flows:

- `int DtmrtRegisterTableCopies(table_name, queue_size)`

This function is similar to the normal registration, but in addition, the DTM server allocates enough memory to contain *queue\_size* copies of the table, which it will pass back one at a time to the client in the event of multiple updates. This type of registration thus provides the support required for *Many-To-One* data flows.

- `int DtmrtReadRemoteTable(remote_host_name, table_name, table_buffer)`

This function explicitly reads the specified table from the remote host into a supplied table buffer. No host, in this case, would call `DtmrtUpdateTable`, but instead clients write directly into the table in shared memory. When many remote hosts do this, then the table instances all contain different data which can be read via this function. This function thus provides the support required for *One-To-One* data flows

### 3.3 Synchronized access and event handling

Synchronized table access is based on subscribing to table update events, which are interesting in themselves as a means of sending real timing events over the network, in this case, the time of arrival of an update event is often more important than the table contents.

- `file_descriptor DtmrtSubscribeTable(table_name)`

Subscribe to real time table update events, the function returns an integer file handle suitable for use within a select system call. The table must already be registered.

- `int DtmrtUnsubscribeTable(table_name)`

Un-subscribe to the table update events, this will happen anyway if the subscribed task stops running for any reason.

- `int DtmrtSelect(select_mask, event_handler, time_out)`

In addition to calling the standard UNIX file select system call, the table data is transferred between the server and the client's local memory. A subsequent call to `Dtmrt-CheckTable` will provide the table name.

## 4. CONCLUSION

The latest DTM package has been operational in the PS control system since March 1995, and is dealing with 20 different tables distributed over 6 control subnets, and about 1000 global registrations is a typical figure. It is currently running on:

- DEC MIPS *Ultrix 4.4.*
- DEC ALPHA *OSF/1 2.1.*
- IBM RS/6000 *Aix 4.1.*
- MOTOROLA MVME147/167 *LynxOS 2.2.*
- CETIA PowerPC 601 *LynxOS 2.2.*
- PC486 *LynxOS 2.2.*
- SUN Solaris 2.3 *SunOS 4.1.3.*
- HP *HP/UX 9.1*

During this time it has recovered from network failures, CPU crashes, general sector power-failures, task crashes and new users developing application programs. The simple and elegant application program interface has permitted the *port* of complex timing-system libraries and the applications using them, to all of the above platforms with minimal effort. DTM also allows us to modify control system tables on all platforms while applications are still running and avoids the need to build global data into these programs, with the maintenance problems and inflexibility this causes.

## REFERENCES

- [1] J. Lewis, V. Sikolenko: "The new CERN PS timing system", ICALEPCS, Berlin, Germany Oct 18-23, 1993, Nucl. Instr. And Meth. A352 (1994), 91.
- [2] F. Perriollat, C. Serre: "The new CERN PS control system overview and status", ICALEPCS, Berlin, Germany Oct 18-23, 1993, Nucl. Instr. And Meth. A352 (1994), 86.

# STAR Controls System

J. Meier, R. Burke, M. Cherney, P. Colarco, J. Chrin, J. Gross, T. McShane, P. Teeter  
Creighton University, Department of Physics, Omaha, Nebraska 68178, USA

P. Barale, F. Bieser, J. Hunter, S. Jacobson, S. Klein, C. McParland  
Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA

## *Abstract*

The Slow Controls System for the Solenoidal Tracker At RHIC (STAR) is presented. The application of the Experimental Physics and Industrial Control System (EPICS) to this system is discussed. A prototype Hardware Controls system incorporating the High-level Data Link Control (HDLC) protocol has been developed and tested. This Controls system is used to provide a number of on-chamber control functions as well as an alternative path for accessing digitized detector data. Results from the preliminary system tests are presented.

## I. The STAR Experiment

The STAR detector is located on the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory. The STAR experiment will investigate collisions between gold ions accelerated to 100 GeV per nucleon, searching for evidence of the transition of nuclear matter to a quark-gluon plasma state [1]. The use of a Time Projection Chamber (TPC) will enable the STAR experiment to track thousands of particles per event as well as examine and correlate individual particle properties and global event variables. The STAR experiment will be on-line in early 1999.

## II. STAR Controls System

The main purpose of the STAR Controls System is to ensure the validity and consistency of the recorded data so that physics data may be extracted. To accomplish this the STAR Controls System:

- Sets the system parameters according to a pre-defined sequence
- Saves and restores the detector configuration
- Provides an alternate data acquisition path
- Verifies correct functioning of the detector by monitoring the subsystem parameters
- Warns about possible subsystem malfunctions
- Handles alarms
- Controls normal operations
- Provides fault diagnostics
- Allows for simultaneous testing of different subsystems
- Archives information about the subsystem parameters in an easily accessible way
- Provides logging capability

EPICS was selected as the foundation for the STAR Controls software environment because it incorporates all of the system features outlined above. EPICS was designed by Los Alamos National Laboratory and the Advanced Photon Source at Argonne National Laboratory. At STAR, EPICS is run on a UNIX-based workstation connected to VME crates operating under VxWorks, a real-time operating system [2].

EPICS was designed as a development toolkit and common run-time environment that allows users to build and execute real-time control and data acquisition systems for experimental facilities [3]. A principal component of EPICS is its database. The database is an ASCII file that is configured off-line using a Database Configuration Tool and is loaded into the memory of an Input Output Controller (IOC). The database consists of specific records written for alarm checking and setting and for monitoring and controlling parameters within each subsystem.

The current STAR operator interfaces (OPI) are SUN IPX workstations. A display editor is used to create and modify the OPI displays. This allows the user to design the specific control needed for the application [3].

The system hardware consists of a VME crate for each subsystem using commercially available industrial interfaces and programmable controllers wherever possible. The monitoring tasks and system control are distributed on SUN IPX workstations. The workstations and VME crates are networked with ethernet. If a single input-output-controller (IOC) becomes saturated, the processing can be shared with other IOCs. If an operator interface were to become saturated, then the processing could be distributed to other OPIs [2]. The organization of EPICS is shown in figure 1.

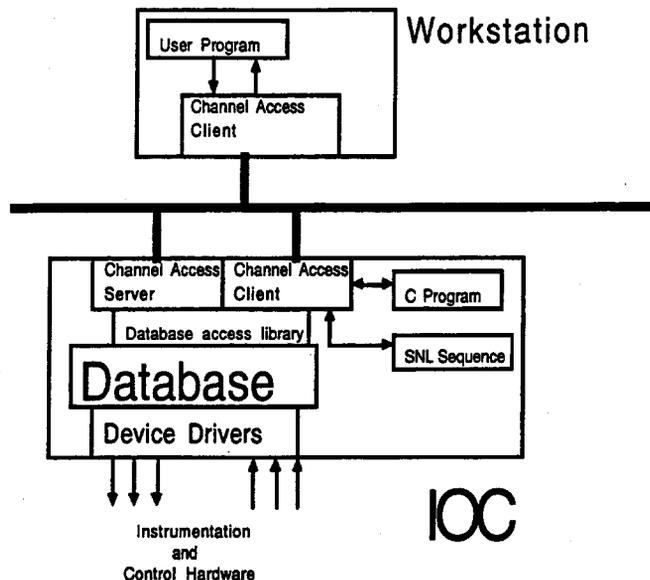


Fig. 1. Configuration of EPICS.

Each subsystem in STAR has an EPICS-based hardware control system which is used by Experiment Control as a tool to control and monitor the performance of the detector hardware. Experiment Control does not execute detailed control of subsystem hardware directly; it issues commands to the appropriate subsystem control processes. This method of implementation serves two purposes. First, it preserves the "chain of command" so that it is clear who or what is controlling a piece of hardware. Second, it provides an interface between the experimentalists working on subsystem control and those involved in Experiment Control.

### III. System test of the Slow Controls Interface with the TPC

The STAR TPC readout boards require a link to the Slow Controls system for a number of testing, control and monitoring tasks as well as to provide STAR with an alternate data acquisition path. Since each readout board stores over one megabyte of data per event, the Slow Controls link speed is required to be in excess of 10 kB/s to be of use as an alternate data acquisition path. This link must operate well from the electronics platform to the TPC endcaps, a distance of 30 meters, as well as in the 5-kG magnetic field present in STAR. Due to space considerations, the link must also accommodate a multi-drop topology to reduce cabling [4],[5]. To accomplish this, a bi-directional data transmission and control system that interfaces easily with the EPICS software environment, is required.

HDLC was selected as the protocol for the TPC/Slow Controls link because it incorporates all the requirements outlined above. The link throughput is about 70 kB/s. In the STAR implementation, the signals are sent using RS-485 drivers and receivers. The differential transmission is important to avoid ground loops. The major disadvantage of the generic HDLC is that it requires some software development and more on-board hardware than some of the more highly integrated choices [4],[5]. The HDLC interface accepts function-specific parameters passed from an EPICS database to execute a desired data transmission.

The Slow Controls TPC System test was designed to test the TPC readout boards functionality as well as the Slow Controls and Data Acquisition links that interface with each readout board. The TPC Front End Electronics (FEE) boards receive event signals from the chamber, then perform amplification and digitization of the signals. The TPC readout boards receive input from the FEE boards and store the data in event memory or send the information to the Data Acquisition (DAQ) subsystem through a fiber optic link. By pulsing the FEE

boards with known signals the Slow Controls link is used to verify that the signals were properly amplified, digitized and stored in memory. The Slow Controls link is also used to write random data to the readout board memory. This data can be read from the readout board by DAQ and also by Slow Controls. Comparisons can be made between the original data and the data sets that are read by DAQ and Slow Controls. This method is used to test data transmission on the fiber optic link as well as on the Slow Controls link.

The TPC has two endcaps each of which is divided up into 12 super sectors. Each super sector network is made up of 6 readout board slave nodes and one commercially-built master node in a VME crate. The slave nodes are based on a 68302 microcontroller running at 20 MHz and are built on a 73-mm by 84-mm printed circuit "daughter" board which is designed to plug on to a readout board. The board requires +5 V at 200 mA, and communicates with the readout board via memory mapping [4]. The commercially-built master node is a Radstone SBCC-1 which is based on a 68360 processor and supports 4 HDLC channels. For the system test, one TPC sector was fitted with readout boards and slave nodes which were then connected to one HDLC channel on a Radstone master node. Timing signals were sent from the Radstone board and reflected back by the readout boards to generate a proper phase read clock. This was done because the 68302 serial ports cannot self clock. The hardware configuration for the system test is shown in figure 2.

The HDLC-EPICS interface consists of subroutines which are loaded into the IOC at boot time and perform read and write transactions to and from a TPC readout board memory in a transparent manner via the Radstone board. The HDLC command format allows specification of starting address, network number, slave node number, number of 16-bit words to read or write and an offset value which is the number of 16-bit words to skip between each read or write. Commands sent via the HDLC link may read or write up to 1012 16-bit words [4],[5].

A new EPICS software record was created that allows EPICS-based control programs to pass the appropriate parameters to the HDLC interface on the Slow Controls side. This record was based on a similar record for VME data transfers developed at the University of Chicago Synchrotron Radiation Community. The record allows the user to select which network and node to access and to input a starting address, the number of words to read or write and an offset value.

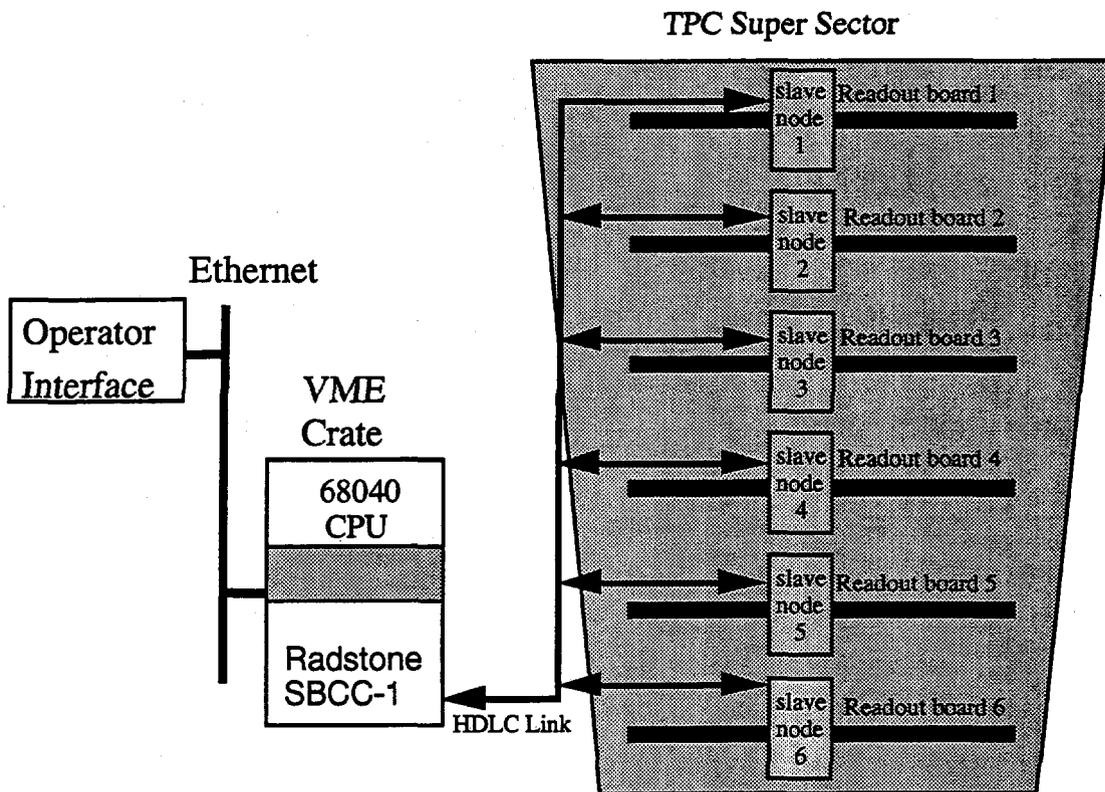


Fig. 2 Hardware configuration for system test. There are six readout boards per super sector, each of which is slave-node daisy-chained by an HDLC cable to a Radstone board. (Each Radstone board can service up to 4 HDLC links, only one is shown.)

#### IV. Summary

A solid foundation has been laid for interfacing the EPICS-based STAR Slow Controls system with the STAR TPC subsystem using the HDLC protocol. Preliminary results show that the HDLC-EPICS interface is capable of reading and writing event data to and from the TPC readout boards reliably. Applications have been built that control and monitor functions on the readout board. The transfer of randomly generated event data from a ASCII file on a workstation to the memory of a 68302 daughter board and back for comparison has verified the proper functioning of the HDLC link.

#### VI. References

- [1] J.W. Harris and the STAR Collaboration, " The STAR Experiment at the Relativistic Heavy Ion Collider ", Nucl. Phys. A566, 277c (1994).
- [2] J. Gross, M. Cherney, T.S. McShane, " A Unified Control System for the STAR Experiment ", IEEE Transactions on Nuclear Science, (February 1994) 184-87.
- [3] R. Cole, M. Kraimer, B. Wilson, F. Lenkszus, J. Anderson, J. Hill, B. Cha, F. Vong, A. Kozubal, L. Burczyk, B. Ziemann, B. Gunther, N. Karonis, B. Daly, M. Stettler, "Experimental Physics and Industrial Control System (EPICS) ", (January 1992).
- [4] P. Barale, F. Bieser, J. Hunter, S. Jacobson, S. Klein, C. McParland, " The STAR TPC FEE HDLC Link " (STAR internal note), (22 June 1995).
- [5] S.R. Klein et al., " Front End Electronics for the STAR TPC", submitted to IEEE Transactions on Nuclear Science, (October 1995).

# A New Approach to Control Systems for Medium-Scale Accelerators

Masakatsu MUTOH and Yoshinobu SHIBASAKI

Laboratory of Nuclear Science  
Tohoku University 1-2-1 Mikamine,  
Taihaku-ku, Sendai 982, Japan

Isamu ABE

National Laboratory for High Energy Physics (KEK)

1-1 Oho, Tsukuba-shi, Ibaraki 305, Japan

An accelerator control system which uses an object oriented database has been designed. This control system will be installed in a new accelerator, the STretcher-Booster ring (STB), being constructed at Tohoku University. Furthermore, we plan to change the old control system of the Tohoku University electron linear accelerator to this control system in the next stage. It should become the most suitable control system for medium-scale accelerators on the grounds of both minimized construction cost and good performance.

## 1. INTRODUCTION

The STB [1] has three operation modes: the first is the stretcher-ring mode, which converts a pulsed beam from a 300 MeV electron linac to a continuous beam; the second is the booster-ring mode, which will accelerate the beam up to 1.2 GeV and inject it into a new storage ring which will be built in a few years time and the third is the storage-ring mode, which will store the beam at 1.2 GeV. The control system has been designed so as to be suitable for these complex operation of the STB. The design goals are to reduce the construction and development costs, to enhance the flexibility and extendibility and to avoid becoming old-fashioned because of the fast innovation of computer technology. In order to achieve these goals, we have decided to employ the latest computer technologies: personal computers and Programmable Logic Controllers (PLC), which are rapidly advancing in performance and object-oriented programming technology.

A simple database system, which saves only the operation parameters, has been used in the linac control system[2]. Since the accelerator will become a larger part of a more complex system, the establishment of a more sophisticated control system is required. A powerful database therefore becomes of great importance. In the future, if this control system is to be adapted to the linac and other accelerators, it should be done without any major modifications. Therefore we cannot afford to develop software that is suitable only for specific accelerators. Using an object-oriented database to enhance the flexibility and extensibility of the control system will provide a good solution to this problem.

## 2. OBJECT-ORIENTED DATABASE

In general, a database called "object-oriented" should have not only object-oriented programming functions, such as the encapsulation of data and services (methods), a definition of class layers and an inheritance from basic class, but should also have relational database functions, such as a guarantee of data integrity and process checking (rules, triggers and stored procedures). In order to construct an object

oriented database having the functions mentioned above together with the ability to respond quickly to query messages, a combination of a Microsoft SQL server and interface programs using Open Database Services (ODS) has been used. The structure of the database is shown in figure 1. The data contained in the database are preset values for normal operation, actual operating values, scanning rates to collect data, PID coefficients for loop controls, alarm messages, bit data from the beam monitors and all other data needed to control the accelerator. These data must be treated by methods constructed from procedures stored in a Structured Query Language (SQL). The stored procedures set the operational parameters for each accelerator device through the PLCs, collect the operational data from the PLCs and store the data to disk. The data and the stored procedures are put together into data tables in the database. One data table is formed together with the data and the stored for each type of device. Common items (voltage, current, coefficients, etc.) in the tables for each category are collected together in tables which are placed in an upper layer. The stored procedures in the lower layers can use the data and stored procedures in the upper layer of the same category corresponding to the inheritance in the object-oriented programming. If a new device is added to the database, the data and stored procedures which should be inserted into the database are only different from that for other devices that have been entered in a few respects, and so it is a form of differential programming.

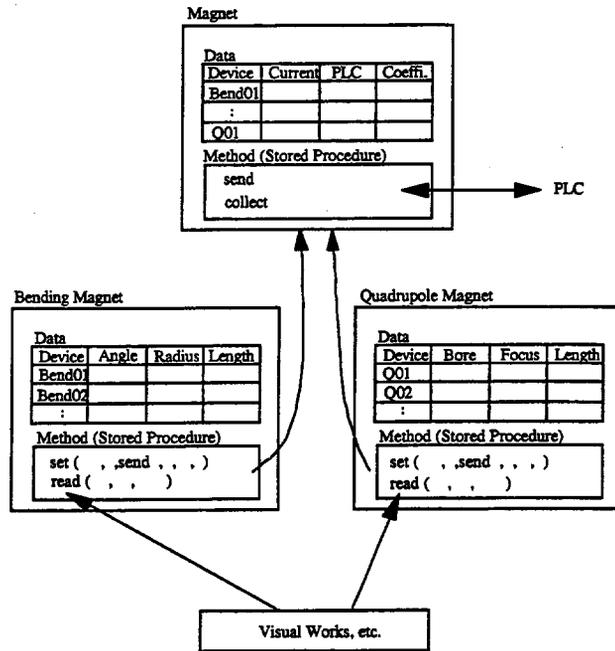


Figure 1: The structure of the database

The distributed-database functions provided by the SQL server will become available further if the accelerator is expanded.

We have completed the basic tests on the database, and have made sure that it will be possible to communicate between the client and the server within several tens of milliseconds.

### 3. SOFTWARE SYSTEM

Figure 2 shows the software configuration. The control program consists of the object oriented database, based on the SQL server and application programs (Visual Works, Excel, Visual Basic, WWW sever, FINS, etc.) which are commercially available. Visual Works is used as a Graphical User Interface (GUI) on the operator's consoles [3] and provides the SmallTalk language, convenient tools and a number of useful class libraries for program development. Excel is used for the statistical display of logged records. Visual Basic is used as a simple language for sequencing operations. The WWW server provides not only real-time information concerning accelerator operation, but also provides help messages and on-line manuals written in Japanese for the accelerator operators as well as to researchers through the campus network. The Factory Interface Network Service (FINS, supplied by OMRON Co.) is a communication protocol between a personal computer and the PLC. The application programs are connected to the database by means of the Open DataBase Connectivity (ODBC), ODS, or the Dynamic Link Library (DLL). The use of their application programs is of benefit to this control system: there are fewer program bugs than are found in laboratory-designed ones, comprehensive manuals, the expectation of upgrading to new versions in the future, and reductions of the development and maintenance loads. We have adopted Microsoft WindowsNT for the personal computer operating system, because it has various important

features: a multi-tasking system taking advantage of 32 bit performance, good network connection and an architecture following the client/server model.

Due to the speed of technical innovations, if much effort is not taken to continuously upgrade the control system, it will become technically obsolete within a few years. To cope with this problem, the personal computers in the system will be replaced by new ones having a higher performance every few years and the application programs will be updated with new versions provided by the software suppliers, or replaced by other similar applications having better functions. Therefore, the control system will be constantly maintained at the optimum condition.

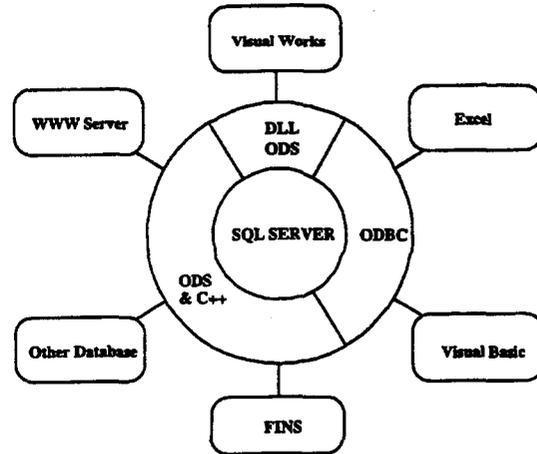


Figure 2: The software configuration

#### 4. PLC

An outline of the control system is shown in figure 3. A PLC has been adopted as a device driver and as an interface between the computer and the accelerator, because the price is lower than that of the CAMAC and VME systems. It is capable of complex control of external devices with simple programs and various modules are available commercially. Recently, because the programming, process speed and network environment of the PLC have improved remarkably, it has become possible to use it for accelerator control [4], except for applications which handle a large quantity of data during a short period

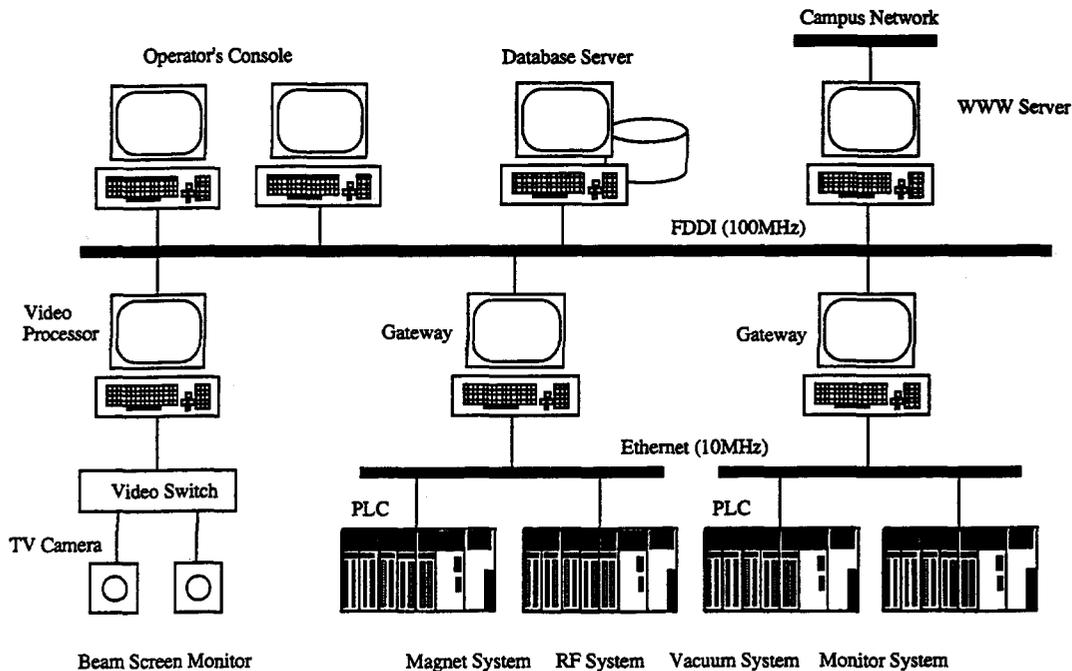


Figure 3: The configuration of the control system

of time.

In this control system, the PLCs will control various magnet systems, an RF system, a vacuum system, a beam-monitoring system, etc. The PLCs function as device drivers that set the operation parameters, monitor the operating condition, check the interlocks (main interlocks are hardwired) and act as a

feedback loop controller. The I/O modules used in this system include the following types of module: ADC (12 bit, 8 channel), DAC (12 bit, 4 channel), digital input (16 bit, 4 channel), digital output (16 bit, 4 channel) and GP-IB interface. The program executed in the PLC is developed in a Sequential Function Chart (SFC) running on personal computers connected to the FDDI backbone network and it is stored in the database server computer. When the system starts, the program is loaded into the PLCs with the characteristic data on the controlled devices. The PLCs are connected with a 10 MHz Ethernet using a UDP/IP protocol, and communicate with the database server through gateways (the personal computers) for converting the transfer speeds and protocols.

## 5. PERSONAL COMPUTER

The performance of the personal computer has improved rapidly during the last few years. Especially in the GUI environment. The appearance of new operating systems available for multi-tasking (OS/2, WindowsNT, etc.), network connection, programming languages and tools for application development, memory and hard-disk capacity and low initial and running cost, makes the personal computer a better choice than the workstation. In our system, personal computers will be used in the following roles: database servers, operator consoles, image processors for digitizing and displaying video signals from beam screen monitors, gateways and WWW servers. All of the personal computers will be IBM PC/AT compatible.

## 6. CONCLUSIONS

Various basic tests on the database have been carried out and the control system is being constructed at the test facility. Since the STB will start operation for nuclear experiments next autumn, this control system must be completed by next summer at the latest.

## REFERENCES

1. Oyamada et al., "The Tohoku University Stretcher-Booster Ring", Proceedings of the 10th Symposium on Accelerator Science and Technology, Hitachinaka, Japan 1995
2. Mutoh et al., "Improvement of the Control System for Linac and Pulse Beam Stretcher at Tohoku University", Proceedings of the 7th Symposium on Accelerator Science and Technology, Osaka, Japan 1989 p.240
3. Mejuev I., Abe I. and Nakahara K., "Application of an Object-Oriented Analysis for the PF Linac Control System Development", Proceedings of the 20th Linear Accelerator Conference Osaka, Japan, 1995 p.212
4. Shirakawa, I. Abe and K. Nakahara, "Feasibility Study of PLC in the PF Linac Control", *ibid.* p.218

# Upgrading of the U-70 complex controls

V. Komarov, A. Sytin, E. Trojanov, V. Voevodin, V. Yurpalov  
*IHEP, Protvino, Moscow region, Russia*

The 70 GeV accelerator complex in IHEP has been in operation since 1967. Its control system was originally oriented on remote manual control. Some computer-based control subsystems created subsequently have different structures and use obsolete computers. To provide better availability and reliability and to reduce maintenance costs a system upgrade is unavoidable. The paper describes the upgrade plan and the basic solutions adopted. As an example of the approach to the new control system structure the ejection system is described.

## 1. INTRODUCTION

The 70 GeV accelerator complex consists of a 30 MeV linac, a 1.5 GeV booster, the main proton synchrotron ring (U-70), slow and fast ejection systems and proton and secondary beam channels (fig. 1). The booster (and hence the linac) is a fast cycling machine which works in a batch mode (up to 32 pulses per batch with a 60 ms period). The accelerating time is approximately 30 ms. Pulse to pulse modulation (PPM) is used to provide beams for two users - U-70 and an "internal" booster experimental facility. The U-70 magnetic field pulse has 2 flattops, the first for beam injection and second for beam extraction. The cycle duration is less than 10s, including a 2.6s rise time. The fast ejection system can be triggered 3 times during a cycle, extracting from 1 to 29 bunches per shot. The ejection energy range is 30 to 70 GeV. The slow ejection system can extract a debunched beam up to one second in duration. There are special modes of operation such as using internal targets and slow non-resonant ejection. During a run all these modes of beam extraction may be used in any preprogrammed combination.

The complex has been used for about 30 years for partial physics. Except for the booster, which was added in 1985 with computerized control, the other installations of the complex had originally only remote manual control. Their computer-based control systems were developed independently from each other and at different times. To use the complex as an injector for UNK, the Beam Transfer Line (BTL) was constructed in 1993 with its own control system [1,2]. The U-70 complex has no general control system and its individual technical systems are controlled from the Local Control Rooms (LCR) situated in different buildings. The LCRs exchange a limited number of operational data in a non-modern way (fig 2). This leads to difficulties in system maintenance and development.

## 2. PRESENT STATUS OF CONTROLS

The current status of the controls of the U-70 complex may be summarized as follows:

- the computerized control of the U-70 technical systems is based on obsolete computing means (EC1010, E-60, SM1420, SM1810.XX, etc.)
- the hardware of the U-70 complex individual control systems consists of a great variety of methods of construction and standards, and the software is written in different languages and works under different operating environments
- a significant part of the U-70 technical systems equipment is still controlled manually.

## 3. UPGRADE PLAN

The upgrade of the U-70 complex control system will take about 3-4 years and comprise the following list of tasks:

- the replacement of the obsolete computers, providing new MCR consoles and local access from technical systems
- putting into operation a control network
- the replacement of old electronics and hence the development and production of electronic modules to modern standards
- adaptation of those technical components which now have only manual control into the computer system
- re-engineering of all control system software.

The U-70 complex is a running machine and the upgrade of its control system must be done without serious disruption of its high energy particle physics program.

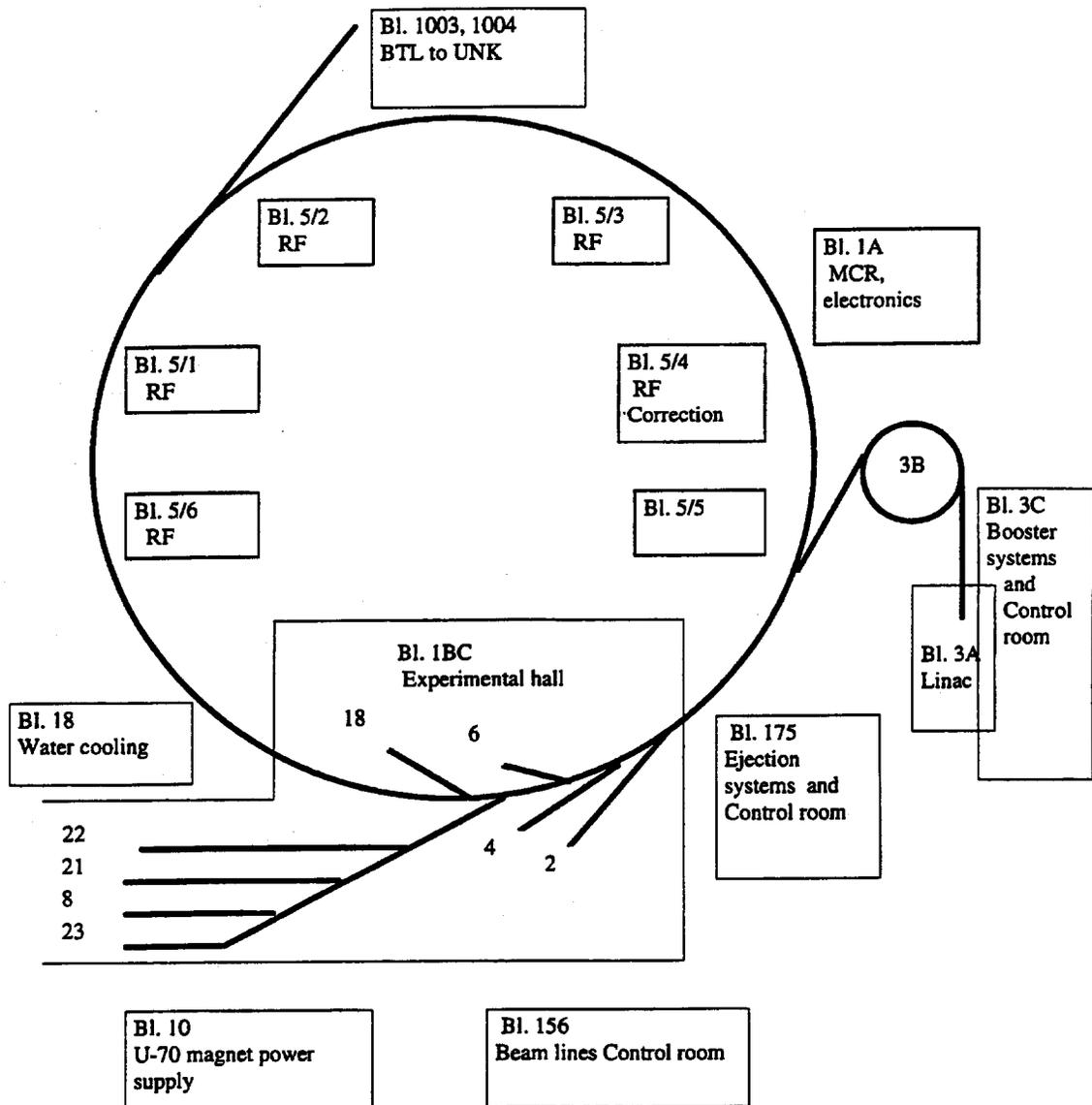


Fig.1. Technological buildings of the U-70 complex

### 3.1. Architecture

We intend to use the results of the UNK Control System upper-level design as much as possible. In particular the new Control System of the U-70 Complex is planned to have the same architecture ("standard" three levels) and to contain the same hardware/software components as were chosen or developed for the UNK Control System [3]. However the final decision concerning the components will be taken after their reliability and speed tests at the U-70 complex under actual working conditions have been completed. Together with Work Stations and X-terminals the 16 PCs with MS DOS will be used as operator consoles on the upper level of control system as a cost-effective measure. Some of these PCs will act as front end computers (FEC).

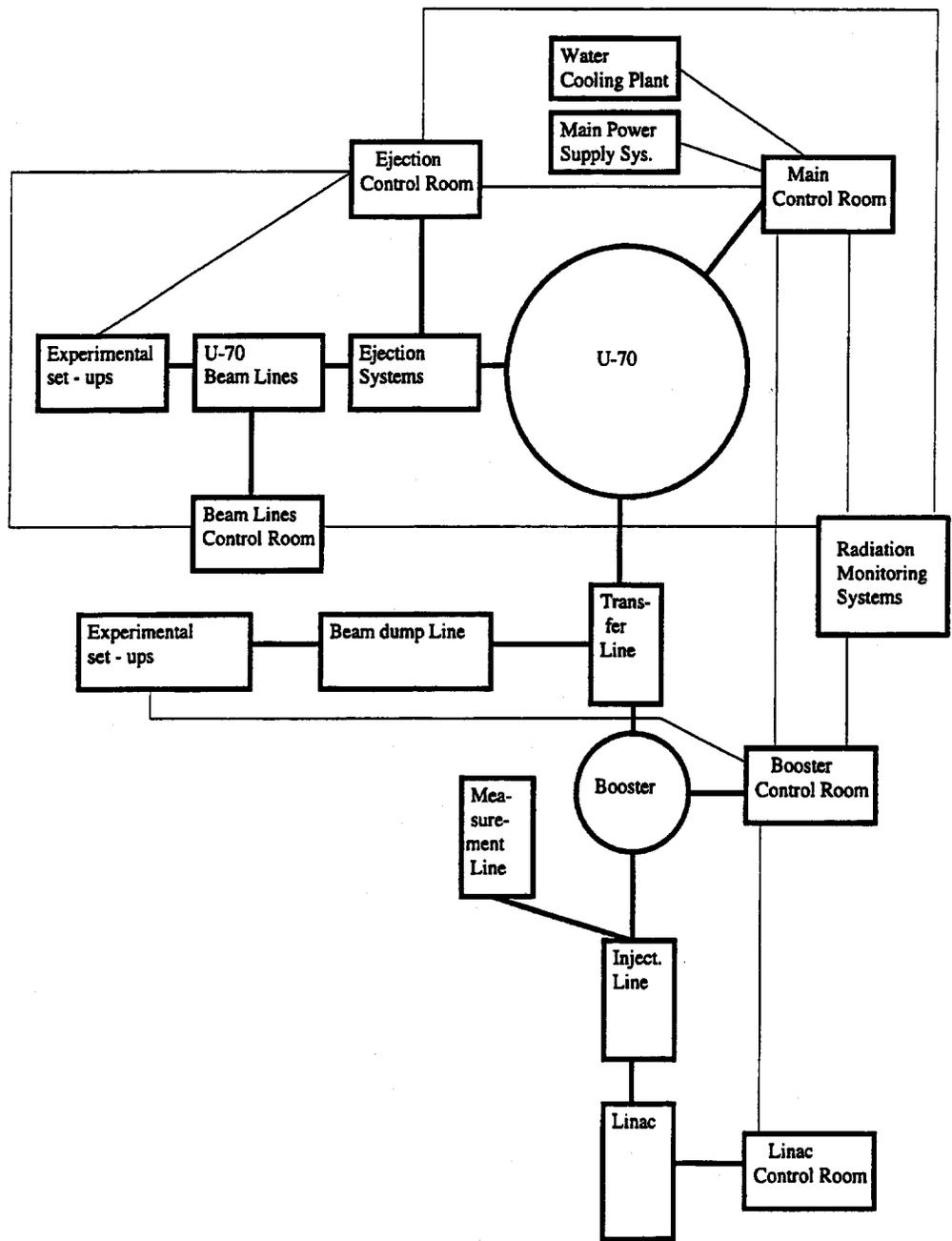


Fig.2. Cabling between Control Rooms

### 3.2. Controls network

The Ethernet-based controls network infrastructure (fig. 3) currently covers the most important sites: the U70 MCR, the Booster CR, the U70 beam extraction controls, the two local control rooms for U70 beam lines and the UNK BTL controls.

Each site has one or several "access points" (multiport repeater) to which computers are connected. Sites are separated by bridges. This control network is connected with the OEA development platform network, in order to provide computing resources and services, including access to the IHEP campus network.

The controls network infrastructure is not yet completed and must be extended to a few more sites. In addition a separation from the office network has to be made to make control more reliable.

### 3.3. Equipment controllers (EC)

A typical EC (total number is more than 70) is a Euromechanics crate with a Multibus I compatible backplane bus. An EC contains a single board computer (PC-16) based on a microprocessor similar to the Intel 8086 and a field bus interface (MIL 1553). As a first step most of old electronics made in SUMMA (CAMAC) will be controlled via a branch driver in the EC. Later, the SUMMA crates will be replaced with specific I/O cards in the same EC. EC's based on the Intel 8031 and 8051 microprocessors embedded in technical equipment will be used. for some subsystems (i.e. Power Supply).

### 3.4. Software

The software for the new control system will be based on the well known standards and protocols accepted for the UNK controls project:

- Unix and/or Unix-like operating system
- C programming language
- TCP/IP communication packages
- X-Windows and OSF Motif as the basis of the MMI
- Oracle DBMS and related tools

In addition, we plan to use the equipment access library which was developed for the UNK project [4,5], a home-made real time DBMS - SSUDA which provides a 3-D table structure for dynamic data storage, and also several home-made tools developed to simplify operator interface creation in applications [6].

## 4. CURRENT ACTIVITY EXAMPLES

### 4.1. Ejection system

Parts of the control electronics for the ejection systems were designed at various times by various organizations (CERN, IHEP, MRTI). Part of the equipment is still operated manually, the rest has control based on obsolete computers "Electronika-60" (like LSI-11) and SM1420 (like PDP-11). As a result we have now a conglomeration of various styles, configurations and standards. The following table illustrates the volume and diversity of the existing control equipment:

Mechanics and standards

Type	NIM	"Vishnia"	SUMMA	non-standard
Number of Crates	81	31	6	22
Number of Module Types	105	51	20	6

The new control system for the ejection complex will present a homogeneous structure of 8 functional subsystems integrated by fieldbus and FEC (fig.4). It will provide computerized control for all the extraction equipment. The EC I/O modules are connected to the controlled equipment either directly or through adaptation electronics. Some of the control and measurement procedures have to be synchronized with the beam structure. For this purpose a special RF generator (RFG) forms a so-called RF-train of pulses linked to the bunches. The RFG uses 2 input signals which

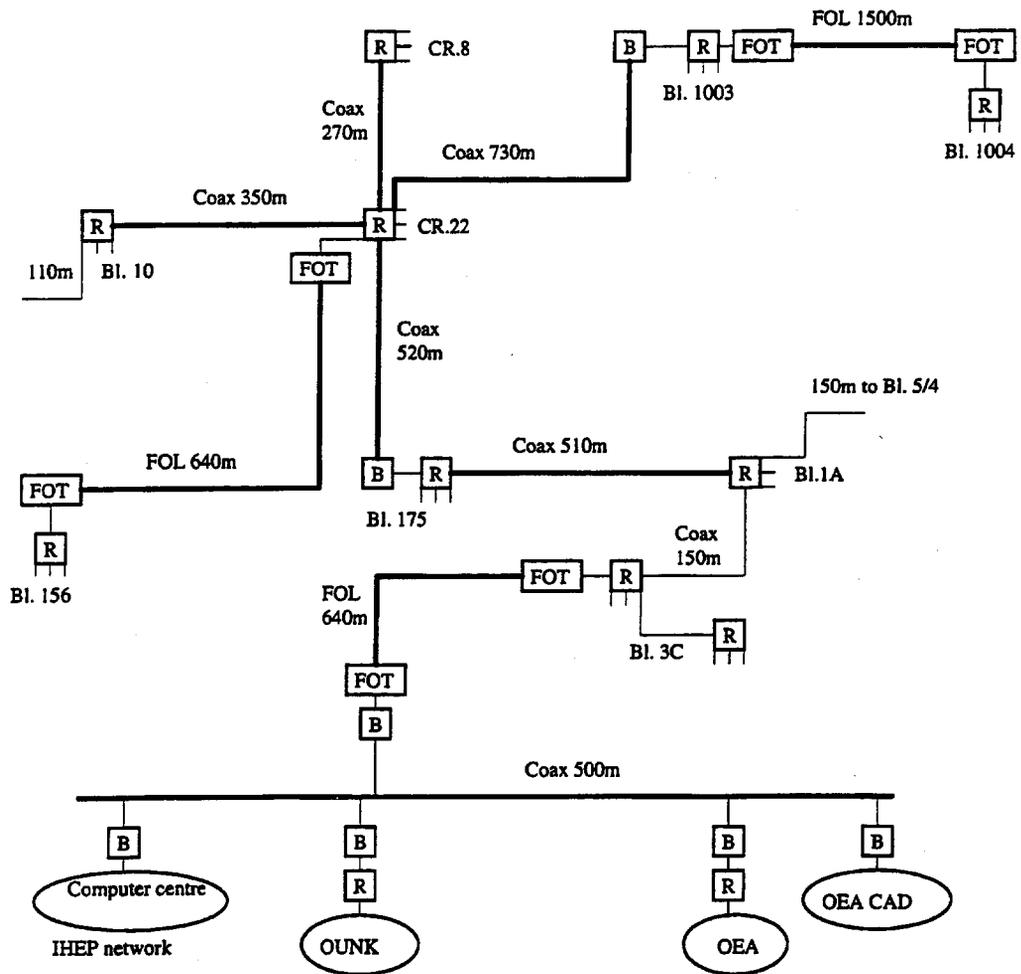


Fig.3. Control network infrastructure

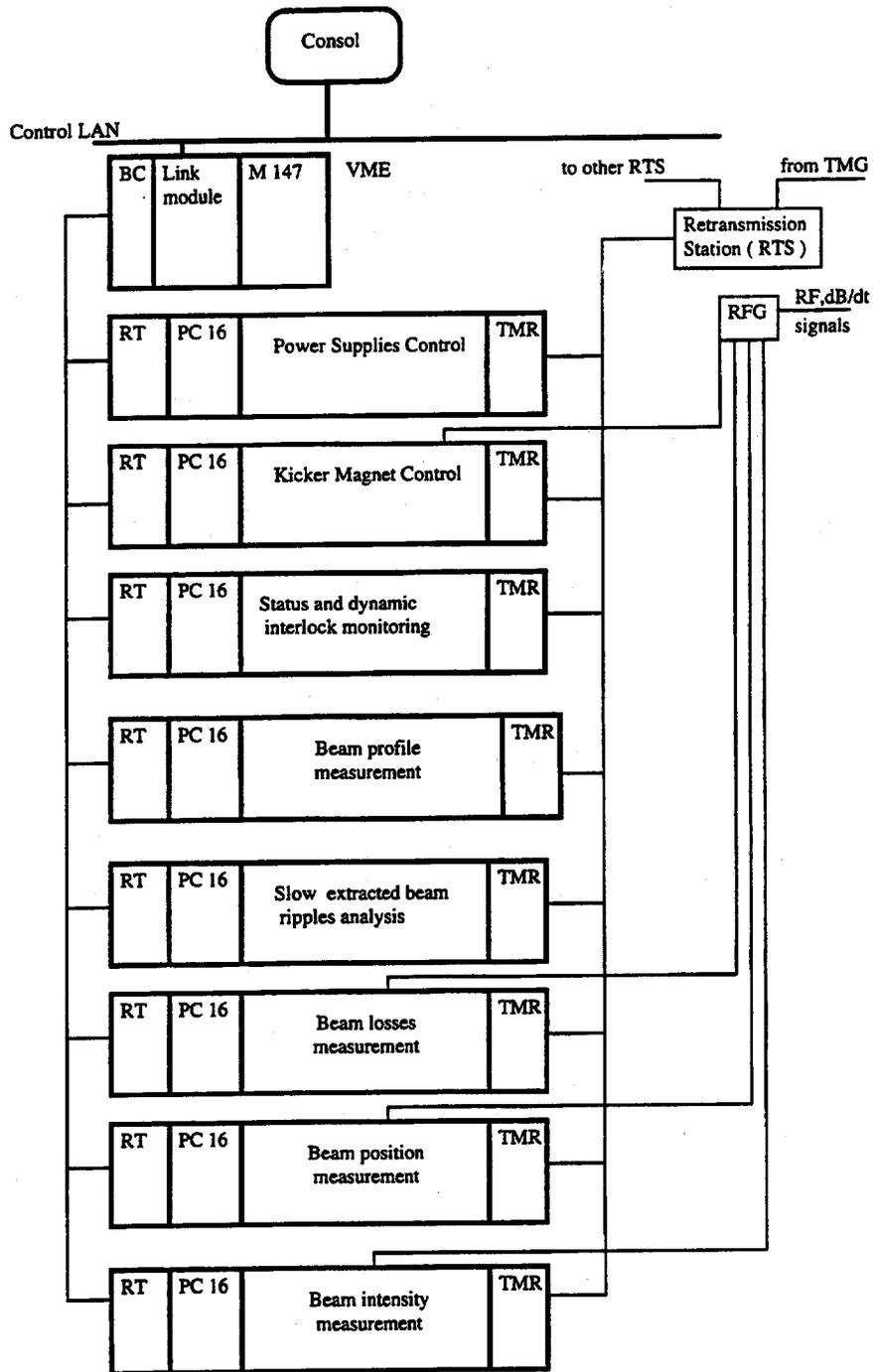


Fig.4. Ejection upgraded control system

are proportional to the accelerating RF voltage and the main magnetic field derivative (dB/dt). The different types of control hardware needed have been estimated to be the following:

- EC (Euromechanics crates).....8
- EC module types.....11
- adaptive electronics (small size crates).....17
- adaptive electronics module types.....8

Such a great decrease of the volume and diversity of the control electronics will simplify the maintenance and reduce its cost.

At present about 80% of the module types are developed and ready for serial manufacture. This year two ECs for power supply control and beam profile measurement are being assembled and will be installed in the ejection system LCR. Next year we plan to assemble three ECs, leaving the rest for 1997. The commissioning of the ECs and the control system as a whole will be carried out during the shut-down of 1996-1997. The completion of the upgrade design is planned to be in the middle of 1998.

#### 4.2. General timing system

The existing U-70 general timing system was built in 1967. It generates and distributes a few pulses called the T-train and the B-train for common use ("cycle start", "reset", etc.). Each of these pulses and trains is transmitted from the MCR to users by means of individual distribution channels. To produce all necessary pulses locally, a large number of scalars is used. The new timing system is based on the technical solution designed for UNK and presented at this conference in detail [7]. The basic principle of the system is time-division multiplexing, which provides efficient utilization of the transmission cabling and the electronics. The main components of the system are the: timing message generator (TMG), timing message receiver (TMR) and multimode timer (MMT). The TMG forms the stream of encoded input information and sends it into the common transmission channel with the rate of up to 10 timing messages/ms. There are four sources of timing information: a 1kHz clock, preprogrammed events written in buffer memory, possible external pulses and 16-bit input data (number of cycles, astronomical time, current values of main magnetic field and beam intensity, etc.). The TMR decodes and processes the input information: it extracts the 1kHz clock, converts the event codes to output pulses and to interrupt request signals, writes down the new 16-bit data in dedicated registers. The multimode timer (MMT) can be programmed to produce both up to 6 pulses delayed with respect to the trigger and up to 3 pulse batches with programmed parameters. At present all the main components of the system have been designed. A segment of the new timing system was tested under actual working conditions and good results were obtained. This year two sets of the timing equipment will be installed in the booster and ejection systems. The installation of the new timing system design will be carried out gradually, depending on the readiness of the control subsystems, and completed in 1997.

#### REFERENCES

- [1] A. Sytin. Present status of the UNK control system at IHEP. In Accelerator and Large Experimental Physics Control System, p. 56-60, North Holland, 1994. Special issue of NIMS Vol. A347.
- [2] V. Gotsev et al. Controlling the UNK transfer line beam diagnostics. In Accelerator and Large Experimental Physics Control System, p. 199-200, North Holland, 1994. Special issue of NIMS Vol. A347.
- [3] V. Alferov et al. The UNK control system. In Proceedings of The International Conference on Accelerator and Large Experimental Physics Control System, p. 134-139, KEK, Tsukuba, Japan, 1992. National Laboratory for High Energy Physics, KEK.
- [4] A. Elin et al. Control protocols for the UNK control system. In Accelerator and Large Experimental Physics Control System, p. 271-273, North Holland, 1994. Special issue of NIMS Vol. A347.
- [5] N. Trofimov et al. The equipment access software for a distributed UNIX-based accelerator control system. In Accelerator and Large Experimental Physics Control System, p. 274-276, North Holland, 1994. Special issue of NIMS Vol. A347.
- [6] L. Kopylov et al. Tools for man-machine interface development in accelerator control applications. In Accelerator and Large Experimental Physics Control System, p. 421-423, North Holland, 1994. Special issue of NIMS Vol. A347.
- [7] V. Komarov et al. Timing system of IHEP accelerator complex. Report on this conference.

# Managing an Evolving Control Environment

G. Neu, R. Cole\*, K. Lüddecke\*, G. Raupp, W. Treutterer, D. Zasche, T. Zehetbauer  
Max Planck Institut für Plasmaphysik, EURATOM Association, Garching, Germany  
\*Unlimited Computer Systems GmbH, München, Germany

## ABSTRACT

The tokamak ASDEX Upgrade is equipped with a fully digital machine and discharge control system. Long term operation and continuous upgrading of such a system must allow for changes of physical discharge definition, real-time control procedures and I/O system peripherals. A system administration platform is under construction to keep track of the real-time system's relevant process and signal inventory along with its evolution and which allows one to check its consistency with the discharge definition selected for the execution of actual experiments.

## 1 INTRODUCTION

During their lifetime the control systems of large experimental devices undergo frequent modifications. Improvements and optimization measures are carried out continuously to achieve higher performance, functionality, operability and reliability. Constant evolution is not an indicator of bad design but a measure of the important role the control system plays in the experimental set-up.

System evolution of an experimental control environment may be characterised by three separate but interdependent modification cycles, Figure 1. Device operation is based on the variation of discharge parameters collected in the discharge definition and observation of the resulting changes in physical key parameters from cycle to cycle. The set of discharge parameters is modified by the physicist in charge, typically on a hourly timescale. Knowledge gained during operation periods leads to the improvement of existing, and development of new, control algorithms and methods. Hence the real-time software will have to undergo modification by the software engineer. Change of the control software or availability of new or improved sensor signals or actuator can require modification of the peripheral I/O hardware. Component failures require instant replacement of modules during the operation of an experiment, and as a result, peripheral I/O hardware can and will be reorganized and altered by the technical staff.

To guarantee correct operation of the control system all these modifications have to be made consistently. Due to the system size, with a large number of hardware and software components and its complexity and interdependencies, it is extremely difficult if not prohibitive to keep track manually of the consequence of any intervention. This is especially difficult as many scientists, engineers and technicians collaborate in the execution of these tasks, all with their own scope and knowledge.

What is therefore needed is a management concept which, based on information about the real-time system's relevant process and signal inventory and their evolution, ensures consistency with the discharge definition selected for execution.

In this paper we will describe the ASDEX Upgrade discharge control

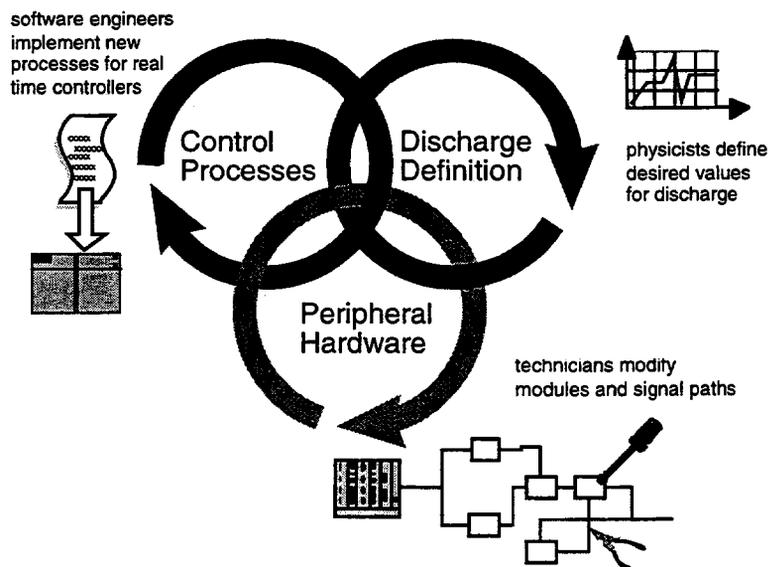


Fig. 1 : Modification Cycles in System Evolution

environment and its parametrization, and derive requirements for a system administration platform, Chapters 2 and 3. In Chapters 4 and 5 adequate information representation mechanisms are introduced which help to describe the control processes and signals and retrieve appropriate information. These are used to validate the physical discharge definition and to create an executable discharge programme.

## 2 ASDEX UPGRADE DISCHARGE CONTROL SYSTEM

ASDEX-Upgrade is a mid-sized divertor tokamak designed for investigation of plasma boundary physics and wall interaction in various geometries and under conditions similar to those in a fusion reactor. Physical control tasks range from the active stabilization of the plasma's position in the vessel and the shaping of its outer countour, to control of refueling and heating systems by feedforward access or feedback of plasma quantities and monitoring of technical subsystems and overall technical and physical discharge status. To perform these real-time control tasks and to operate the machine in the most flexible way ASDEX-Upgrade has been equipped with a fully digital control environment as indicated in Figure 2.

The discharge control system (DCS) is a hierarchical cluster of transputer-based computers running a multitude of real-time monitoring, feed-forward, and feed-back control processes. Processes require information on hundreds of actual physical and technical quantities and access to tens of technical actuators to act back onto the plasma. I/O is performed via a widespread system of peripheral hardware modules. Each controller transfers its output data to and receives input from specified interface points which relate to the diagnostic sensors and technical actuators previously configured by the machine control system (MCS).

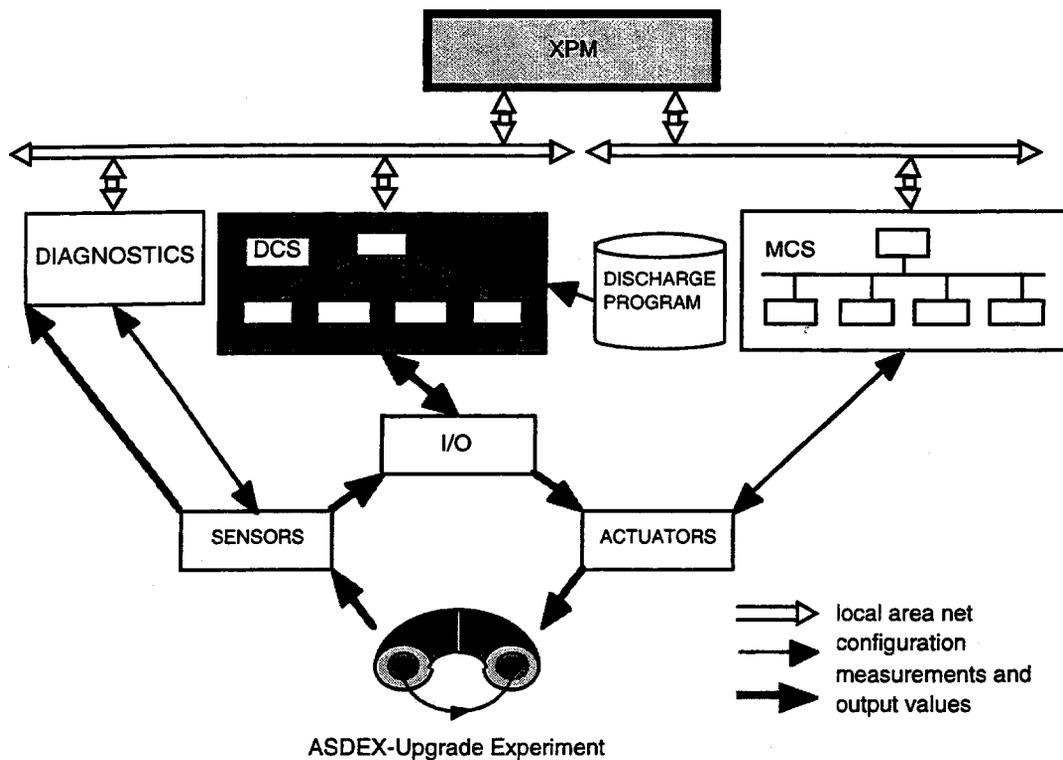


Fig. 2 : ASDEX Upgrade Control System Environment

An experiment management (XPM) software platform integrates MCS and DCS for automated experiment operation. It serves as a user interface for the technical operator and processes his commands, hiding the associated complex time ordered sequences of actions and information exchange between the distributed systems [1]. The core of the DCS is the real time control processes. These are specified by experimentalists and control system designers and implemented as code modules by the software engineer. Each process performs operations on time-varying signals. For each process an entry in a header file exists which provides identification of all its signals and defines their usage.

Signals can be subdivided into I/O ones characterized by their transfer characteristics, addresses, and format and reference parameter signals whose values must be provided in the discharge program. A manually maintained global quantity descriptor database (QTD) contains the defining characteristics of all signals in the DCS.

It is via the peripheral hardware that processes access actuators and sensors to interact with the experiment. When a signal coming from a diagnostic sensor's interface point reaches the computer port, it will have passed through a series of modules such as decoupling and pre-amplifiers, ADC or digital input, multiplexors, parallel to serial converters and optical and electrical transmission lines. This also holds for an output signal on its way to an actuator's interface point. Signal paths are specified by the control system designer and set up by the technical staff to meet the format and transmission characteristics required by the control computer to correctly access, identify and interpret a signal. Beyond the interface points, sensor and actuator characteristics have to be known to correctly interpret an I/O signal. Whereas the latter are set in the MCS, the former are input both into the QTD entry of the corresponding input signal and into a diagnostics database.

To prepare specific experiments the physicist designs a discharge definition consisting of the full set of desired signals required to parametrize a given control software version. A signal oriented editor is used to input values for configuration switches for process activation, desired values for monitoring trajectories, parameter switches and associated timebases. The executable discharge programme is created by linking the discharge definition, the MCS operation parameters defining actuator characteristics, and the peripheral I/O system's transfer characteristics and sensor characteristics as given in the QTD. The dependencies of these various inputs are shown in Figure 3.

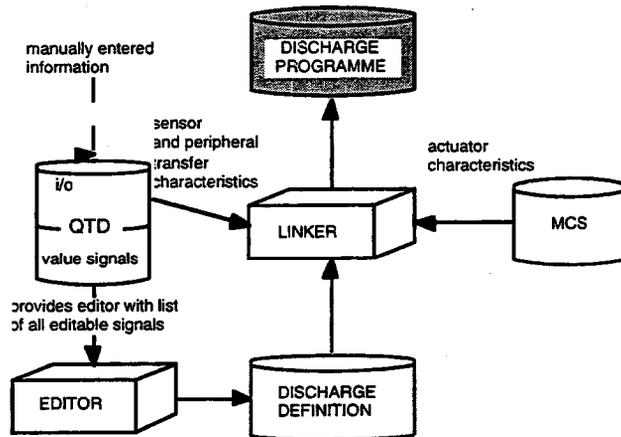


Fig. 3 : Assembling a Discharge Programme

### 3 REQUIREMENTS FOR CONTROL SYSTEM ADMINISTRATION

There are a number of ways in which inconsistencies can be introduced into this system. Major changes in functionality begin with the specification of a new process description which leads to the implementation of a new code version. If new signals are required these are specified in the QTD. Necessary modifications of the I/O configuration (e. g. additional I/O signals, changes in I/O format, rearrangement of already existing signals) have to be implemented in the peripheral hardware, including diagnostics sensors, and the corresponding updates made in the QTD. One should note that inserting or replacing a single mux module may mean having to recompute the transfer characteristics of several signals. Finally, the I/O routines of the control computers have to be adapted. Discharge definitions which were not designed for the new software must be enhanced and/or modified to account for new signals or signals whose usage has changed.

Clearly consistency is not enforced by the system described above and information required to validate a scheme is not provided automatically. The description of the process structure and lists of signals of a given software version - needed to define a consistent discharge description - must be manually assembled by inspecting the configuration files in the source code. There is also no true description of the I/O hardware structure at the module level - needed by the technician to assess the impact of changes on the I/O signal entries of the QTD. Modifications in the sensor characteristics are usually protocolled in the diagnostics database from which they have to be manually transferred into the QTD.

From the above considerations it is clear that maintaining consistency within the control system is far from trivial. It relies on the correct interplay between control processes and peripheral hardware and between control processes and discharge definition. All of these are independently configured and modified by different people and on different timescales.

An ideal solution would be a complete model description of the entire system software and hardware, which could be used to allocate and parametrize real-time processes and activate and configure real-time I/O hardware. With the given system complexity and implementation resources, however, such an approach is currently far from being realised.

To provide the flexibility required by experiment goals or operation procedures, the ability to change control system characteristics is fundamental. Hence, what has to be provided are supporting measures for the prevention and the detection of inconsistencies.

A first step towards prevention is an organization of system information avoiding multiple definition, and an adequate and correct representation of this information. Whenever changes are made it should be clear where the corresponding information is maintained, and mechanisms should exist which update related information automatically. Any intervention should be based on reliable and precise knowledge of the set-up of the components to be handled. For the physicist designing a discharge definition, the information of interest would be an inventory list organized by process of all signals required in a downloaded software version; a software engineer can draw valuable information from a visual representation of a controller's process structure; and a technician will appreciate an I/O hardware representation which will allow him/her to modify module descriptions instead of having to go through endless lists of I/O signals

Detection of inconsistencies relies on the specification of rules which define the relations between information from the various components of the system. Instances must be created which implement these rules and extract the relevant information.

#### 4 REPRESENTATION OF THE PERIPHERAL HARDWARE

In the previous chapters we have seen that a useful description of the peripheral hardware must be module oriented. It must mirror the installed I/O system's structure to facilitate maintainability, and it must allow the retrieval of the actual signal I/O characteristics to be processed by the controller at a given port.

The general strategy in modelling the peripheral hardware is straightforward - the real-world hardware modules are represented as module descriptions, including board and transfer characteristics. The real-world hardware connections between module I/O ports are represented as connection descriptions. The end points of this peripheral I/O system to the other system components are a sensor/actuator description on the peripheral side to sum up the characteristics of the signal preprocessing hardware outside the control system, and a controller port description as the end point of a fiber optic transmission line to define the interface to the controller internal process description.

Any relational database can serve as an implementation platform. For each type of module templates must be provided which define its static and variable properties and characteristics. Examples of static properties are the type description, front panel set-up, number and kind of I/O connections, and connector type. Even before being actually used a specific module will allow personalization of its representation. This is done by assigning values to some of the module's variables: a serial number for future identification is defined, values for specific parameters are set (e.g. the addresses of multiplex channels or amplification factors), status (never used, repaired, modified ...). Finally, when the module is installed, a location description and interconnections with other modules can be put into the description. Special 'modules' describe the two ends of the peripheral hardware: on the one side the controller I/O boards with ports, and on the other side virtual modules which describe the interface points with the sensors and actuators of the experiment. A sensor module will typically contain the name and characteristics of a sensor and one entry describing its connection to some module of the peripheral hardware.

Such a model representation describes the entire I/O hardware system topology and transfer characteristics, using all required information and avoiding redundancy. The transfer scalings of a peripheral signal on its way to a controller can simply be inferred by stepping through the modules and extracting the local transfer characteristics along the signal path. To check whether a signal path is consistent with given controller software all that is needed is the

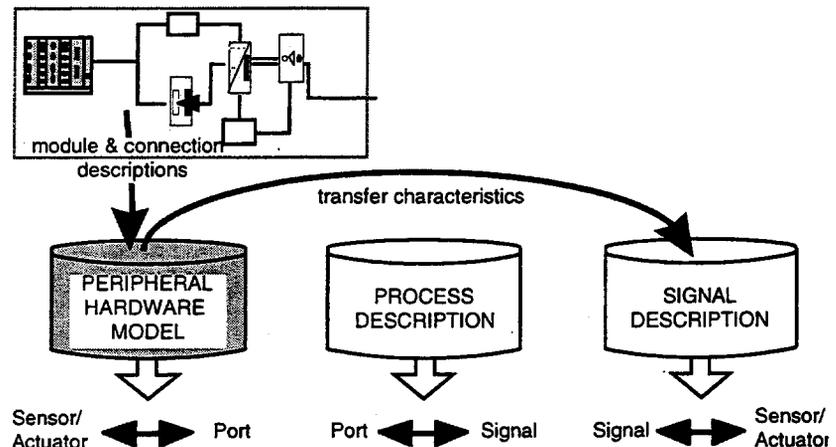


Fig. 4 : Application of the Peripheral Hardware Representation

relation connecting the signal name to an actuator or sensor and the information concerning at which port that signal is expected by the controller, Figure 4.

If the loop indicated in the figure can be closed the signal path exists and the signal can be accessed with the actually derived I/O characteristics. The latter are then automatically updated in the signal description of the QTD. Otherwise, the signal path hardware is not installed correctly or the physicist's specification to use a particular sensor/actuator as a control signal is wrong or the control process expects the signal at another port or not at all. In such a case, the model description can assist in debugging by giving information on where the loop is broken, where the signal path ends or which set of signals is actually sent to a given port.

## 5 ADMINISTRATION OF THE PROCESS STRUCTURE

Enhancing the software of an evolving real-time system to provide the functionalities required for administration is both difficult and inefficient. This is especially true for a distributed system running on a variety of platforms and programmed in different languages. Checking the compatibility of a given peripheral hardware set-up or the completeness and correctness of a discharge description would mean having to activate all controllers and related subsystems.

Therefore, to deal with this problem at ASDEX Upgrade a generic model was devised, which can be configured for various software versions and parametrized by arbitrary discharge programs. To keep it as simple as possible, the model only contains those data structures and rules or functions strictly necessary to perform the desired administrative tasks.

The hierarchical architecture of the model reflects both the existing software structure and the tasks to be performed. Rules can be classified according to their complexity and generality. Whereas some checks can be performed locally and by every process (e.g. determining whether all signals it requires are included in a given discharge program), others will decompose into sequences of specific actions. As an example for the latter, consider the real time task of controlling the antenna coupling of the ICRH. Its implementation in ASDEX Upgrade's control system software requires the cooperation of two control processes running on different controllers [2]. When activated, the first one feedback-controls the antenna coupling by computing a desired gap between antenna and plasma boundary. Provided the second process is configured to control the outer radius of the plasma and to receive this desired value as input, it will compute currents for fast control. The correct execution hence relies on particular settings of various software control switches and on the definition of desired values and gain factors for both processes.

The model has been specified using an Object Oriented (OO) paradigm, and a prototype is currently being implemented in C++. The process class library contains classes ranging from basic template processes, elementary processes (e.g. single-variable PID controller) to derived classes defining the complete process structure of a real-time controller. Inherent properties of OO methodologies such as encapsulation of methods and structures into a class, inheritance in derived classes and polymorphism of methods were helpful in designing a library of reusable and easy-to-enhance modules.

The generation of a particular model instance is performed using customized methods which address configuration tables declared in header files of the original source code of the control software. This guarantees that the model structure corresponds to that of the target system, see Figure 5.

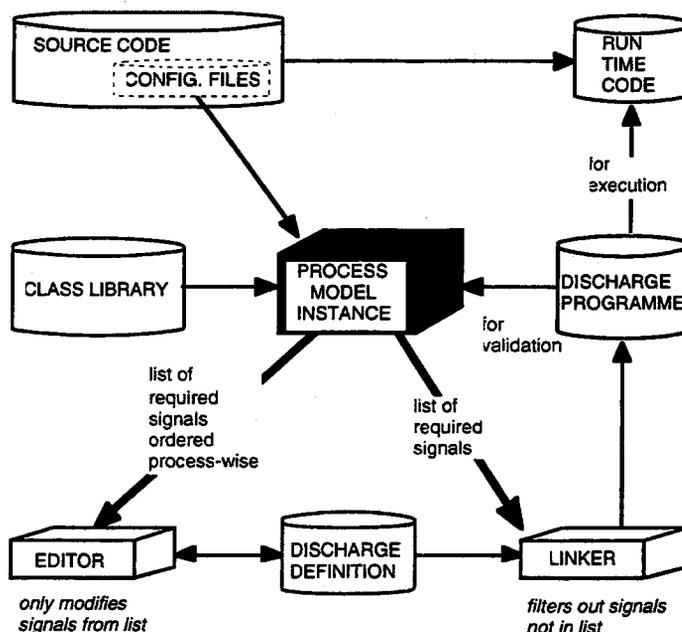


Fig.5 : Process Model: Creation and Application

Once a model has been activated, it can provide the discharge definition editor with the lists of processes and those signals which have to be defined for a complete discharge definition. A similar list is passed on to the linking editor, which will add to the discharge programme only those I/O-signal descriptions which are actually needed. Finally, the discharge programme can be validated according to the methods defined in the classes of the process class library.

## 6 CONCLUSION

In an evolving control system for a large experiment such as ASDEX Upgrade, control processes, discharge definitions and peripheral hardware are modified continuously. In the past, a system description was obtained by entering critical information manually into a set of unrelated databases. This carries the risk of introducing inconsistencies. With the new approach presented here a system description is extracted from the installed hardware and software modules, and rules are added. Within this system model, consistency checks can be performed and discharge programmes can be generated and validated. We expect this approach to considerably improve the management of change within ASDEX Upgrade's control environment

## REFERENCES

- [1] Richter, et al.: "Overview of the ASDEX Upgrade Experiment Management Software" ; Proc. 17th Symposium on Fusion Technology, Roma (I), 1992, p. 1077
- [2] T. Zehetbauer, et al. : "Management of RT Processes for Plasma Parameter Optimization at ASDEX Upgrade" ; Proc. 9th Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics, Chicago, (USA), 1995

# Generalized Control And Data Access At The LANSCE Accelerator Complex -- Gateways and Migrators\*

S. C. SCHALLER and M. A. OOTHOUDT  
Los Alamos National Laboratory  
Los Alamos, New Mexico 87545, USA

All large accelerator control systems eventually outlast the technologies with which they were built. This has happened several times during the lifetime of the accelerators at Los Alamos in the LAMPF/PSR beam delivery complex. Most recently, the EPICS control system has been integrated with the existing LAMPF and PSR control systems. In this paper, we discuss the provisions that were made to provide uniform and nearly transparent sharing of data among the three control systems. The data sharing mechanisms have now been in use during a very successful beam production period. We comment on the successes and failures of the project and indicate the control system properties that make such sharing possible.

## 1. INTRODUCTION

The Los Alamos Neutron Science Center (LANSCE) at Los Alamos National Laboratory is composed of an 800-MeV proton linac, a Proton Storage Ring (PSR) to compress the beam pulses in time and beam lines that transport the proton beams to experimental areas including some with targets devoted to neutron production. The original LAMPF Control System (LCS) controlled the linac and several external beam lines [1]. About 10 years ago a separate control system was constructed to handle the storage ring [2]. In the past few years, server software has been added to allow LCS access to PSR data and commands [3]. This past year we have embarked on a project to upgrade the PSR controls to an EPICS-based system [4]. The combined new control system is called the LANSCE Control System (also LCS). This paper discusses the data and command access infrastructure needed to keep a heterogeneous control system functioning smoothly and with high reliability in the face of the major upheavals caused by such an upgrade. A complete description of the status of the PSR controls upgrade can be found in a companion paper at this conference[5].

## 2. THE ENVIRONMENT OF THE UPGRADE

The basic goal of the PSR controls upgrade was to introduce more reliable and flexible hardware and software while continuing to provide a highly reliable and available controls environment for accelerator operations and development. Budgetary constraints forced us to use an incremental approach to the upgrade. This condition meant that many parts of the old system would have to work with the new system for periods of unknown duration. The need for high reliability implied that we needed fallback capabilities in case some parts of the upgrade encountered problems that severely interfered with operations. A constraint that aided us in making control system decisions was the long term goal of using EPICS-based controls for the entire LANSCE control system.

Fig. 1 provides an overall view of the data access environment with the three (LCS, PSR, and EPICS) controls systems in place. The upgrade plan had essentially two parts: 1) replace the PSR front-end computers (called Instrumentation SubSystems or ISSes) with EPICS Input/Output Controllers (IOCs) and 2) move the PSR application programs onto the EPICS Operator Interface computers (OPIs). We expected that not all the PSR front-end computers could be replaced at once and also that if problems should develop with the EPICS IOCs, we should be able to go back to one or more of the original PSR front-end computers. We did, however, expect that once the PSR application programs had been moved to the OPIs we would not have to fall back to the original PSR operator interface.

---

\* Work supported by the U.S. Department of Energy.

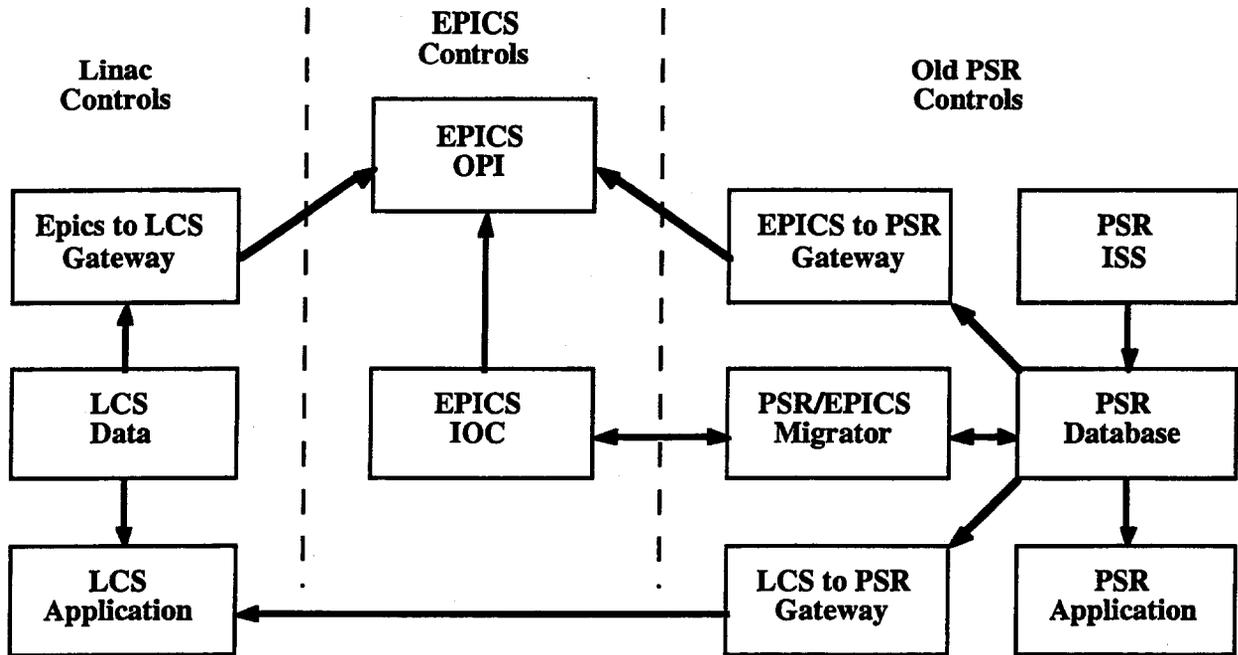


Fig. 1. Data flow in the LANSCE control system. Commands flow in reverse.

### 2.1 Data availability

The data access environment in the various control systems differs as follows:

- o LCS data is acquired on demand through VAX/VMS and VAXELN systems. The data itself comes from the hardware through CAMAC, RS-232 and locally designed hardware interfaces. All references to LCS data make use of the static LCS database.
- o PSR data in the old PSR control system was obtained from the dynamic central PSR control system database. The data was continuously acquired through CAMAC on PDP-11-based ISSes. The central PSR database was updated at regular intervals from information contained in the ISSes.
- o EPICS data is extracted on-demand or on-change from dynamic databases residing in the EPICS IOCs. The IOCs obtain data by polling through CAMAC interfaces, which are the same CAMAC interfaces that were on the ISSes. Access to the databases in the IOCs is via EPICS Channel Access, a TCP/IP-based protocol.

### 2.2 Application program execution

The environments for application programs and generic tools also vary among the three control systems.

- o LCS applications/tools run on VAX/VMS systems. Communication with the data access systems is via a locally designed RPC interface based on DECnet.
- o Old PSR applications run on a single VAX/VMS system. They acquire data from the local central PSR database. Some PSR applications, however, access the CAMAC directly without going through the database.
- o EPICS applications/tools run on EPICS OPIs which in our situation are Sun workstations. The applications communicate with the IOCs via EPICS Channel Access.

### 3. COMMUNICATION PATHWAYS

The data and control pathways needed to support the upgrade were determined by the following requirements.

- (i) We wished to continue using a number of generic LCS tools to communicate both with PSR and EPICS channels. In particular, we wanted to make use of the LCS data archiver and LCS knobs. This meant that we needed a path between LCS applications and both PSR and EPICS channels.
- (ii) Since we didn't know how many of the old PSR ISS systems would be converted to IOCs and because we needed a mechanism to fall back from IOCs to ISSes, a path was required from the EPICS OPIs to the central PSR database. Note that this requirement restricted our freedom in naming channels in the IOC databases.
- (iii) The long range desire to have the entire LANSCE facility controlled through EPICS and the fact that some PSR data physically resides on LCS systems meant that we needed to provide LCS data to EPICS OPIs.
- (iv) Finally, we knew that not all PSR applications would be converted to EPICS OPI programs. The unconverted PSR applications would need to continue to acquire their data through the central PSR database, including data that had been moved to the EPICS IOCs.

#### 3.1 Gateways

The requirement in (i) above for giving the LCS system access to PSR data and commands was already in place before the current upgrade began. As part of an earlier project to combine the linac and storage ring controls in a single control room and to move toward a single operator interface for the two machines we had implemented a server process on the PSR control system. This server responds to generic data and command requests from LCS application programs and tools. This server is labeled "LCS to PSR Gateway" on Fig. 1. It implements the LCS standard pseudo-device access protocol and allows tools such as the LCS archiver and knobs to have access to PSR data and commands.

We defer the second part of requirement (i), the LCS access to EPICS, until after we have discussed migrators below.

Requirements (ii) and (iii) essentially required us to provide the EPICS channel access protocol on both the LCS and PSR systems. Fortunately, a similar requirement had arisen at DESY and, through the EPICS collaboration, we became aware of preliminary work on an EPICS to DESY gateway that ran on VAX/VMS systems. Mathias Claussen and Gabor Csuka kindly shared their code with us and we used it to develop the software that forms the "EPICS to LCS Gateway" and the "EPICS to PSR Gateway" on Fig. 1. As part of this development, we added channel access puts and monitors to the DESY gateway and modified the LCS database to more closely conform to the EPICS database functionality.

#### 3.2 Migrators

The need to run unconverted PSR applications in requirement (iv) meant that the PSR database had to contain current data and allow commands to channels without regard to whether these channels were on an EPICS IOC or on an original PSR ISS system. The original PSR database update scheme used a program called the "migrator" to read the ISS databases and put the new data in the central PSR database and to send changed command parameters from the PSR database to the ISSes where the commands would take effect. This migrator effectively kept the PSR and ISS databases synchronized.

Requirement (iv) meant that we had to find a way to keep the PSR and IOC databases synchronized. To do this we implemented another migrator, the "PSR/EPICS Migrator" in Fig. 1. This migrator runs in the context of the old PSR control system and uses the notification on change mechanisms that are present in both the old PSR system and in the EPICS IOCs.

Unlike a gateway, a migrator has a fixed list of channels and channel fields to handle and it provides data bidirectionally. The PSR/EPICS migrator handled three types of data. Alarm and warning limit changes made by LCS or PSR tools in the PSR database were polled and sent to the EPICS IOCs when needed. Commands given by LCS or PSR tools were migrated from the PSR database to the EPICS IOCs. The PSR database provided notification on change for these channels. Commands, readbacks and alarm/warning limit changes made by EPICS

software were migrated from the IOC databases to the PSR database. For these channels, EPICS channel access monitors rather than polling could be used.

Note that with the migrator and LCS to PSR gateways in place, we have effectively satisfied the second part of requirement (i), namely providing LCS access to EPICS data. LCS tools and application programs can request EPICS IOC data by going through the path as follows: LCS application -> LCS to PSR gateway -> PSR database -> PSR/EPICS Migrator -> IOC. The data is returned along the same path. This route is admittedly roundabout, but its existence allowed us to concentrate on other aspects of the upgrade and helped us to meet our schedules.

#### 4. LESSONS LEARNED

The process of creating gateways and migrators has given us a wider perspective on the issues of control system design and the underlying philosophies. Compromises in performance and functionality are inevitable in making such disparate systems communicate. We hope that some of our experiences may be of use especially in light of the continuing discussions on software sharing [7] that have been and are part of the ICALEPCS conferences.

One of the difficulties in integrating these three control systems was differences in their basic data acquisition philosophies. LCS being the older system was designed with a static database and entirely demand-based data acquisition. Both PSR and EPICS have live databases. The EPICS databases are located in the IOCs; the PSR databases are located in the ISSes and also replicated in the central PSR database. To supply the EPICS notification-on-change functionality, we had to implement a polling mechanism in the LCS system. The polling is done at a fairly high level and hence has less than optimal performance. Parts of the LCS system already support polling at a low level; we expect to use these in the next version of the EPICS to LCS gateway.

Especially during the EPICS control system commissioning and also during the initial storage ring turn-on this year we had a very fluid controls environment. Sometimes devices were available on IOCs and sometimes on ISSes; frequently we ran with a combination of IOCs and ISSes. We were forced to adopt a mechanism for global configuration control which kept lists of channel names that were used by the gateways and migrators to determine which channels to handle (or not to handle). Through all of this OPI screens were being built which referenced channels without regard to their location, i.e., whether they were on an IOC or ISS. In addition, generic LCS tools were in constant use to observe the same channels.

Error handling differences among the three systems also created some problems. Detailed error messages when data acquisition problems occurred were the norm in both the LCS and PSR systems. The VAX/VMS condition code mechanism was used to report error conditions. This mechanism allowed detailed errors to propagate to higher levels of code rather readily. EPICS errors are reported by a small set of status and severity codes and usually cause data displayed on the OPIs to change color. This usually meant that operators knew that the data was good or bad but not why it was bad. We added a number of diagnostic screens to the EPICS OPIs to allow access to more detailed error information.

Another feature that caused problems for the gateway and migrator software was the differences in database contents and functionality among the three control systems. Generic control system tools acquiring data and issuing commands through a gateway need complete information on the channels they are handling. In several instances, our gateway functionality was compromised either because the databases did not contain sufficient information or the interface was not sufficiently robust. In evaluating the functionality offered by each of the three systems, the following observations have been made:

- o Multiple-state channels (i.e., mux positions, magnet states) should be straightforward generalizations of simple binary channels. In fact, binary channels should be a specialization of multiple-state channels. All possible channel states should be spelled out in the database -- either explicitly listed or algorithmically generated. Problems arose in the EPICS to PSR gateway and in the LCS to PSR gateway because some states needed by generic EPICS and LCS tools were not in the PSR database.
- o It is important to distinguish between two types of analog channels, namely, setpoint channels (e.g., DACs) and incremental channels (e.g., stepper motors). It may be possible to treat an incremental channel as a setpoint channel by referencing an associated data channel, but, especially for knob-based control the knowledge of the incremental property is very useful. In an environment that has both setpoint and incremental analog channels both setpoint and incremental commands should be supported.

o In a similar vein, multistate channels should support a "next state" command -- essentially a generalization of a binary toggle command. Needless to say a "go to this state" command, i.e., a setpoint command, is also needed. For a robust implementation, these commands require information about the possible states of the channel from the database.

o Access to complete channel information is vital to building robust generic diagnostics. In particular, it should be possible to determine hardware addresses through references to the database. The EPICS channel access protocol explicitly disallows such access.

o The channel naming standards must be general enough so that the name space can be common for many subsystems. Although the mapping had irregularities that had to be ironed out, the LCS/PSR integration was greatly aided by the fact that their channels shared common naming conventions. These conventions were maintained for the most part when the EPICS IOC databases were defined.

o One would not expect analog data formats to be common among such varied systems. EPICS channel access, however, defined a "network" standard format. The LCS and the old PSR systems use VAX floating point but the EPICS system uses IEEE floating point. Provisions were made in the server (i.e., gateway and migrator) software to do the necessary format conversions.

o Some database functions, including alarm and warning limits were found to be provided in different ways. Some systems handled only one kind of limit, some gave upper and lower limits and some gave target values plus tolerances. A generic system should include algorithms to translate sensibly among these options. The EPICS/PSR migrators in particular were faced with making some difficult and sometimes arbitrary decisions.

o Data representation also entered in the assumptions some systems made about accessibility to "raw" data. LCS data originally came from a limited set of I/O channels which all had a well-defined raw data representation. The raw data was used extensively in the LCS system in particular to provide diagnostic information. The lack of well-defined raw data in the PSR and EPICS systems required some approximations to be made when raw data requests were made by LCS tools.

The compromises that had to be made in each of these areas have a significant effect on the performance, functionality and maintainability of the combined system. The fallback requirement also limited the freedom we would liked to have had in EPICS database structure and naming. But perhaps the real frustration came from having to solve once more problems that one has already solved before.

## 5. FOR THE FUTURE

The combination of gateways and migrators described here served to glue together the LCS, PSR, and EPICS systems for a successful production run. Our primary goal for this year was to eliminate the PSR ISS computers. We have successfully done that -- and thus the the PSR ISS box in Figure 1 could be grayed out. In the near future, we hope to eliminate the entire right hand side of Fig. 1 by replacing the PSR functionality with EPICS-based systems. One thing this change will require is direct access to EPICS IOC data by LCS application programs. This project is already under way.

One of the recent advances in the EPICS collaboration has been the implementation of a server-level application program interface [6]. This interface supplies a standard mechanism to connect the EPICS channel access protocol to non-EPICS control systems. This software will allow us to do away with the rather ad hoc channel access interface used in our current EPICS gateways. Being part of the standard EPICS software release, this interface will allow us to take advantage of new EPICS developments.

## ACKNOWLEDGMENTS

Matthias Clausen and Gabor Csuka of DESY generously shared their initial developments of an EPICS gateway. Jeff Hill of the EPICS development team at LANL helped to guide us through the mysteries of the EPICS channel access server. Eric Bjorklund, John Faucett, and Gary Carr, our colleagues on the LANSCE controls team, provided many useful comments.

## REFERENCES

- [1] G.P. Carr et al., Proceedings of the 1987 Europhysics Conference on Control Systems for Experimental Physics, CERN Yellow Report 90-08 (1990) 107.
- [2] P.N. Clout et al., Proceedings of the Second International Workshop on Accelerator Control Systems, Nucl. Instrum. Methods. A247, (1986) 116.
- [3] S.C. Schaller et al., Proceedings of the 1991 International Conference on Accelerator and Large Experimental Physics Control System, KEK Proceedings 92-15 (1992) 7.
- [4] L.R. Dalesio, et al., Accelerator and Large Experimental Physics Control Systems,, Nucl. Instrum. Methods. A352, (1994) 179.
- [5] M.A. Oothoudt et al., these proceedings.
- [6] J.O. Hill, these proceedings.
- [7] B. Kuiper, Accelerator and Large Experimental Physics Control Systems,, Nucl. Instrum. Methods. A352, (1994) 501.

# Commissioning and Operation Applications of ELETTRA

M.Plesko, Institute Jozef Stefan, Ljubljana, Slovenia

C.J. Bocchetta, F. Iazzourene, E. Karantzoulis, R. Nagaoka and L. Tosi  
Sincrotrone Trieste, Trieste, Italy

## ABSTRACT

ELETTRA is a third generation light source with an electron beam energy of 1.5 to 2 GeV. For the commissioning and operation a large amount of high level software (HLS) was prepared and has been successfully used. We believe that this is the first time that the HLS had been developed well in advance and was ready for the operating personnel. The application programs cover the whole range of needs, have a consistent graphical user interface (GUI) and give a fast response to user actions. Almost all the commissioning tools have now been reused as operational ones, also maximizing the value of manpower

The state-of-the-art applications, which were designed systematically, were the key to the rapid commissioning and great success of ELETTRA. The applications use a common data structure based on machine physics and a set of routines which transparently access the control system. This environment greatly simplifies the development of applications, because the programmers can concentrate on the GUI, which gives immediate information on the status of the machine. This approach significantly reduced the development time and will simplify future maintenance. It is easy to implement new programs in very short time whenever the needs for additional actions arise.

The content of the data structure is not specific to ELETTRA, since it is generated dynamically at run-time from several text-only editable data files. Most of the applications can be easily transferred to other accelerators as long as their control system uses UNIX workstations. The use of standards (C and X11/Motif) and the transparency in the access to the actual control system implementation assure this.

## I. INTRODUCTION

The third generation light source ELETTRA [1] consists of a linear accelerator, a storage ring and a 100 m long transfer line between them. Both the transfer line and the storage ring are attached to a distributed control system, which has been developed in-house[2].

Most of the machine-physics related tasks that were previously done manually and analysed off-line can now be automated via software which has direct access to the equipment through the control system. The software which performs these tasks is generally called high level software (HLS). The topics of typical HLS applications include measurements and analyses of measurements, orbit correction, modelling and machine monitoring.

The HLS which was used during commissioning and later for the operation of ELETTRA had been defined well in advance [3]. It was ready and tested on "day one". This had been achieved by the use of a common data structure which was used by all programs and a simulation program with which the applications had been tested and debugged before they were used on the real accelerator.

A common data structure which provides all relevant machine physics parameters readily in the data structure or on a function call was used in order to have a standardised approach thus simplifying maintenance of the HLS. The call to control system functions is made through only one routine, which is hidden from the application programmer through a layer of utility routines for setting and reading one or several control parameters. The HLS data structure and the utility routines are described in detail in a separate contribution [4].

All application programs are written in C. The reason for this choice was that the graphical interface used by the programs is based on the X11/Motif tool kit, which is supported only in C. The programs run in a UNIX-like environment on Hewlett-Packard 9000/700 series workstations.

Great emphasis has been placed on the graphical user interface of each program. The interaction with the user is completely event driven. The power of X-Windows and the Motif window manager is exploited, using buttons, pull down menus, sub windows, etc. Graphical output is presented in a special plot-widget [5], which has been developed in-house. The applications are running on the normal control system workstations. However, due to their distinct functionality, they are invoked and run independently of the control system man-machine-interface.

The following chapters present all the main applications, sorted by their function and purpose. All applications have a graphical user interface unless noted otherwise.

## II. COMMISSIONING APPLICATIONS

**Display** is an interactive optics design tool. The user may visualise through graphs and numbers the optics associated with the actual power supply settings. By zooming into one section, the user sees a layout of the elements and may change interactively the quadrupole settings and the insertion device gaps and visualise the resulting optics in the model.

The programs **EnergyTL** and **EnergySR** scale the optics of the transfer line and storage ring, respectively, to another energy set by the user. Those programs were extensively used during linac and transfer line commissioning, and storage ring first turn steering and initial accumulation, respectively. **EnergyTL** is still used during linac optimisation, which is being performed frequently during the user mode, i.e., when the electrons are circulating in the storage ring.

**GAP** (Ghost Accelerator Program) [6] is a code which simulates an operating accelerator. It was used for the commissioning of the HLS and for testing of the application programs. The program mainly performs two tasks: *modelling* of a real accelerator lattice and *simulation* of the ELETTRA control system response to application programs. The code is made so that the application programs can run either with the real control system machine or with GAP without any change. GAP is based on the modelling and tracking code **RACETRACK** and is therefore written in FORTRAN. This poses no problems as GAP is called as a subroutine instead of as a call to the control system.

## III. MEASUREMENT, ANALYSIS AND MODELLING APPLICATIONS

Those applications were used intensively during the early days of the commissioning. Currently they are used during machine studies mainly for chromaticity correction and compensation and for dispersion measurements.

The emittance of the linac is measured in the transfer line with the program **Emittance**. It varies one or several quadrupoles and determines the emittance and Twiss functions at the linac exit from the change in the beam profile. The beam size variation with quadrupole current and the resulting phase space ellipses are displayed graphically.

The program **Tune** [7] has been written mainly for applying desired tune values to the storage ring. During the setting of the tunes, the user may also record the associated lifetime and beam current and perform a correlation between these parameters and the tunes currently being set by the program, thus giving a measurement of the stop bands of any resonances encountered.

**Optiks** [8] is a program that measures and corrects the optical functions (i.e. beta functions, phase advances, tunes, dispersion and chromaticity) in a circular accelerator. The program offers in a user friendly and compact manner almost all the information that the operator wishes to know about the machine optics as well as the means for correcting. It makes a wide use of the high-level software data structure including its built-in Twiss function calculation that permits the determination of the nominal machine optics in a fast and easy manner. The measured machine optics functions are obtained by analysing the closed orbit data. Alternatively the quadrupole strength variation technique is also employed to measure the beta function at the quadrupole locations. The user may also measure and correct the dispersion and chromaticity independently.

The integer part of the tunes is determined by a Fourier transform of the orbit. One can also get the Fourier transform of the corrector strengths. Since this routine does not need a closed orbit, it may be used for injection studies and was particularly useful at the initial stages of the commissioning. This routine called **Harma** is also a separate program.

## IV. OPERATION APPLICATIONS

The following applications were used and are still used on a daily basis during the operation of ELETTRA.

The *machine file* allows one to save and restore the status of the machine - the transfer line and the storage ring in the case of ELETTRA. The machine file is used to restore the equipment which is relevant for machine physics and machine operation. It therefore contains the currents of all magnets (bending, quadrupole, sextupole and correctors), the insertion device gaps and the injection element settings. Both the set values and the actual read values are recorded.

This machine file can be considered to be a snapshot of the machine. All HLS applications can access either the actual machine directly or alternatively a machine file. Conversion routines convert the currents from the machine file into machine physics variables and vice versa. The operator loads and saves a machine file with the application **MachineFile**.

Since a tune feedback system [9] has been implemented in Elettra, the program **TransferTune** has been written in order to transfer, once the system is no longer in use, the residual current settings of the quadrupoles used by the feedback to other quadrupole power supplies which are optically more suitable.

The **PStool** is a relatively low level application, which cycles all or a set of magnets and switches them on or off according to a requested procedure. This application is used at every injection, because the storage ring is running at 2 GeV,

while the injection energy is 1 GeV. Before each injection all storage ring magnets must be cycled. Eventually, this application will be replaced by the automatic start-up program.

With increasing time being dedicated to users of the machine, an automatic program which prepares the machine for injection has been written. The program *Automat*, currently in a draft version, performs all the necessary work which a normal operator should perform in order to be ready for the injection procedure. While most of the actions are performed sequentially, others such as opening the insertion devices and cycling of the transfer line are done in parallel as background processes. Special care has been taken both in error handling and in the graphical interface in order to give a feeling to the operator of the actual actions being taken on the machine. The main lack of the present version is the primitive communication with the background processes which is done via files. At the present, a new version is under progress in collaboration with the Elettra Control Group [10] in which all actions on the machine will be in parallel and managed intelligently via pipes. Furthermore, the final version will be extended in order to include the ramping and the positioning of the beam at the centre of the insertion devices, in order to achieve a full automatic preparation of the machine for the users.

## V. MACHINE MONITORING APPLICATIONS

A third generation light source is characterized by the need for an unprecedented beam stability. This requires complete short and long term monitoring of all machine components in order to detect, understand and remove any sources of instabilities. Although the monitoring applications might not be used by operators on a daily basis, some of them run constantly in the background and the results are examined by experts. The use of monitoring programs grows with the efforts to increase the beam quality.

The program *MonitorBPM* displays the readings of all 96 BPMs in the storage ring as a function of time, thus monitoring the drifts in the orbit. In four different display windows, the average and the r.m.s. values of each BPM are plotted for the respective planes. The program saves the entire data collected into a file for post processing.

The program *BPMexpert* has been written in order to understand the functionality of the 96 beam position monitors and their electronics. The program records all the relevant outputs of the monitors such as the readings and the electrode voltages together with the electronic settings as a function of beam current. The annexed graphical interface permits one to correlate and to detect anomalies both by visualizing the above quantities and by extracting those monitors which behave differently than the average.

The program *Mramp* displays the beam energy which is defined by the dipole current, the beam current, the betatron tunes and the average and r.m.s. values of the orbit as a function of time so that the operator can monitor the beam behaviour during the course of the energy ramping.

The programs *MonitorPS* and *Hist* track the behaviour of the power supplies. Both the set current from the control system and the actual current read from a current transformer are recorded for each power supply. The difference between the set and the reading is compared and displayed graphically with *MonitorPS*. *Hist* histograms the difference between the reading at start-up with the actual reading over longer periods and issues alarms if the differences exceed the design values.

While the monitoring applications described above specialize in quick monitoring of individual parameters, *Macmon* [11] not only monitors a wide range of machine parameters but also stores, analyses and compares them. Thus useful correlations can be found and interesting conclusions can be drawn leading towards a better understanding of the machine and consequently increasing its reliability.

*Macmon* is composed of two parts. A "low" level part that performs the data acquisition and storage and a "high" level part with the graphical user interface that deals with data reading, analysing, comparing and visualizing. The visualizing part includes plots showing the value of one parameter versus time, the distribution of values in a histogram and a correlation plot between two parameters. Additionally some high level software programs used to control and/or to measure automatically store data in the *Macmon* storage files.

*Macmon* is monitoring the following machine parameters: machine current, lifetime, injection rate, accumulated current, tunes, beam position in all the 2x96 beam position monitors, beam size, rf frequency, voltage and temperature for each individual rf cavity, injection element voltage and time delays, vacuum in each vacuum sector, insertion devices gap opening, power supplies current read and set values, scraper positions and radiation. Additional information stored in *Macmon* from application/control programs is: dispersion, chromaticity and actions such as the file name used to load the machine power supply currents, cycling and ramping.

## VI. ORBIT CORRECTION, OPTICS CORRECTION

There are two HLS programs developed for the correction of orbit in ELETTRA: *Orbit* and *Bump* [12]. The program *Orbit* integrates all necessary actions on the orbit of both the storage ring and the transfer line together with many options.

Its core is the orbit correction software package *COCU* developed at CERN [13], upgraded with other correction methods. All operations including those required for *COCU* are handled through a very sophisticated graphical user interface that gives complete control over any possible option.

A routine to correct the spurious dispersion in both planes using the steering magnets has been incorporated in *Orbit*. A special effort has been made to obtain sufficiently accurate response matrices for the dispersion, by taking into account the effects on the orbit of the feeddown from sextupoles and thick quadrupoles. By utilizing the existing six-dimensional tracking routine in an extended version of the tracking code *RACETRACK*, a special routine has been developed to introduce the notion of path length in *Orbit* in its simulation mode. With the new routine, the path length deformation due to a RF frequency shift or a sudden shift in the beam energy caused by changes in the corrector strengths modifying the path length can be simulated.

The program **Bump** on the other hand is a dedicated program, developed separately to create local bumps in an interactive manner. The main characteristics of the program are:

- the continuous display of the orbit, interpolating the consecutive BPM readings and also taking into account the corrector kicks by a numerical fit of the trajectory;
- the graphical scroll bars with which the user can select the bump location and set the amplitude and the slope of the orbit;
- the representation of magnetic elements in terms of graphic push-buttons with which the desired correctors can be selected.

**SlowFB** is an extended version of **Bump** intended for the routine machine operation, which performs, under a certain time interval, an automatic local correction of orbit in the insertion device straights using 4-corrector bumps. At a given time interval the program will measure the orbit and correct it if the measured value is outside the specified tolerance. As the suppression of bump leakage, that is to say the modification of the orbit outside the bump area, is an important issue in third generation light sources, a special routine has been developed to calibrate empirically the coefficients for the four corrector strengths which will result in the bumps being purely local. The routine is still under test but has been shown to work successfully in the vertical plane.

The program **IDcomp** [7] has been developed in order to compensate for the linear distortions due to insertion devices. The software presents several compensation techniques, which go from local compensations, where the distortions are forced to be localised, to global compensations, where the variation of beta around the ring is minimised. Tune re-adjustments are also possible. An important feature of the software is the possibility for the user to visualise the resulting optics, before any action is performed on the machine.

## VII. FUTURE DEVELOPMENTS

The high level software was used very successfully during the commissioning and first operation of ELETTRA [14]. In order to complete the set of HLS applications, the following new developments are planned:

- Automatic generation of intermediate machine files for the ramping, with automatic corrections of tunes, closed orbit, optics, chromaticities and optionally dispersion;
- Automatic orbit and vertical dispersion correction;
- Implementation of the single value decomposition (SVD) method for orbit correction;
- An application for changing the horizontal/vertical coupling;
- Automatic change of cavity temperatures as a function of beam current;
- Full automatic preparation of the machine for users;
- Cascade monitoring of all the machine systems coupled with intelligence to interpret error messages, failures and down times.

## VIII. CONCLUSIONS

All major HLS applications for ELETTRA exist. There is practically no off-line post-processing of measured data, since all the necessary applications run interactively on the workstations in the control room. New applications are being developed whenever the need for additional actions arise. Due to the well-defined environment of the data structure and the utility functions, it is relatively easy to implement new programs in very short time.

The great success of the HLS at ELETTRA is a proof that our concept of the HLS has been correct. Although the effort has been substantial, corresponding to about 10 person-years, it was well worthwhile. The majority of the applications have been in principle written in a general way, not specific to ELETTRA. Therefore other accelerator centres could make use of the human resources already invested at ELETTRA.

## IX. ACKNOWLEDGEMENTS

The authors thank the members of the ELETTRA control group for their great help on the X11/Motif tool kit and other computer related topics. Further thanks go to R.Richter and A.Wrulich for fruitful discussions. This work is supported in part by the Slovenian Ministry for Science and Technology.

## X. REFERENCES

- [1] ELETTRA - *Conceptual Design Report*, Sincrotrone Trieste, April 1989.
- [2] D.Bulfone, *Status and Prospects of the ELETTRA Control System*, Proc. ICALEPCS 93, Berlin 93.
- [3] M.Plesko, *The High Level Software Routines*, Sincrotrone Trieste. ST/M-TN-91/11. 1991.
- [4] M.Plesko et al., *An Approach to Portable Machine Physics Applications*, this conference.
- [5] F.Potepan, *Plot Widget 1.6*, Sincrotrone Trieste, 18.5.1994.
- [6] F.Jazzourene, *GAP: Ghost Accelerator Program*, Sincrotrone Trieste. ST/M-94/3 1994.
- [7] L.Tosi, *The High Level Programs: Tune and ID-COMP*, Proc. EPAC 94, London 1994.
- [8] E.Karantzoulis, *Optiks: An Optics Measurement and Correction Program for ELETTRA*, Proc. EPAC 94, London 1994.
- [9] A. Carniel et al, *Tune Feedback System at Elettra*, Proc. EPAC 94, London 1994.
- [10] D. Bulfone et al, *Controls in the Past Year of Elettra Operation*, this conference.
- [11] E.Karantzoulis and M. Plesko, *Macmon: A Monitoring Program for ELETTRA*, Proc. PAC 95, Dallas 1995.
- [12] R.Nagaoka et al., *Orbit Correction in ELETTRA*, Proc. EPAC 94, London 1994.
- [13] B. Brandt et al., *COCU User Guide (Version 7.0)*, CERN-SL/AP/Note91-19, 1991.
- [14] A.Wrulich, *Status of ELETTRA*, Proc. EPAC 94, London 1994.

# Electron Accelerator Control System Based on Radiation–Acoustic Effects

G.F. Popov, V.A. Deryuga, A.I. Kalinichenko,  
Yu.A. Kresnin, and N.G. Stervoedov  
Kharkiv State University, Kharkiv, Ukraine

## Abstract

The system uses ultrasonic radiation as a source of primary information about the current status of the accelerator. The sound is generated by the electron beam and by a high-frequency electromagnetic field in their interaction with the accelerator components and with the target. The system incorporates two CAMAC crates and a computer, and it fulfills the requirements for transporting, focusing and monitoring the beam and follow-up phase-locking electromagnetic waves in accelerating cavities.

The system has been designed to control a traveling-wave linear accelerator with electron energy of 5 to 30 *MeV*, an electron pulse width of 10 to 4000 *ns*, a pulse repetition frequency of 1 to 300 *Hz* and with an average beam power of up to 5 *kW*. A block diagram of the electron accelerator control system is shown in Fig. 1. It incorporates a set of acoustical sensors with electronic preamplifiers, two CAMAC crates and an IBM PC-compatible computer with corresponding software. Crate A deals with phase-locking a h.f. electromagnetic field in an accelerating cavity to an electron injector pulse and controls the transmission of a beam through a transfer line. Crate B carries out the focusing of the beam onto a target and measuring the beam parameters. The analysis of data and the generation of control commands are executed by the computer. The crates contain facilities for digitizing signals from sensors, buffer memory devices, units for controlling executive devices, and microprocessor crate controllers.

The sources of primary information for system operation are ultrasonic waves  $\sigma_T$  generated by the beam and the h.f. 1.8GHz electromagnetic  $\sigma_P$  field [1-7]. When a pulsed electron beam hits components of the accelerator rapid heating and expansion of their materials takes place. Thus sonic waves are generated and carry information about beam parameters and about the location of the interaction spot.

Sonic waves are also generated by ponderomotive action of a pulsed h.f. field on waveguide elements. The waves carry information about the form of the pulse of electromagnetic waves and this form depends on the conditions of energy exchange between the h.f. fields and electron bunches being accelerated [6-10].

The sonic waves  $\sigma_T$ ,  $\sigma_P$  detected by broadband ( $\Delta f \leq 10$  MHz) piezoceramic sensors AD-AD5 (see Figure 1) placed on accelerator elements and on a target T. This sort of sensor has high sensitivity and good radiation resistance and sensor construction provides protection against electromagnetic interference. The h.f. field phase-locking is executed based on signals from an acoustical sensor AD1 placed on an absorbing load AL of the accelerating waveguide AW. The maximum transfer of energy from the h.f. field to the electron bunches is achieved when the amplitude of acoustic stress is minimal. Pulses from the sensor AD1 are led to a gated amplitude-to-digital converter in crate A. A corresponding subroutine operates through the crate controller and a relay control unit for a klystron KL phase-shifter PS servomotor to make the amplitude of  $\sigma_P$  minimal. Thus one strives for the maximal energy transfer.

The automatic control of beam transport is carried by multiple-cavity accelerators based on signals from acoustical sensors AD2, AD3 placed on a diaphragm in an electron transfer line. Signals from the sensors are led to both gated analog-to-digital and time-to-digital converters in crate A. If the beam is off the transfer line axis some signal amplitudes go over a preset limit. In that case the determination of beam position is made on the basis of the delays of two acoustical pulses relative to the sync pulse of the accelerator. Then corrective changes of beam positioning magnets BPM are made. Corresponding digital-to-analog converters will be used for that purpose in crate A.

Some other system of beam transport were tested on the accelerator LUE 300 in Kharkov. The beam transmission through the system is monitored by using signals from acoustic detectors placed along the accelerating waveguide. Both pulse amplitudes and the delays relative to the sync pulses of the accelerator were digitized and inputted into a computer, where the beam position was determined and the control commands for changing the currents of deflecting magnets were generated.

The current correction continues until the signals from the detectors vanish and the signal from the acoustic probe AP showing beam position appears. This procedure worked in single-pulse mode in a short time due to the high information content of the acoustical signals.

On high-current accelerator Fakel in Moscow these detectors serve for emergency protection of the beam-transport pipes against damage from the high-current beam. When the amplitude of an acoustic signal goes over a preset limit the injector is disabled preventing damage to the accelerator elements.

Focusing the beam and measuring the beam parameters are implemented on the basis of signals from sensors AD4, AD5 placed on the target T (AD5) and on an acoustic probe AP (AD4) which is on a metallic wire crossing the beam. The signals are digitized by a gated analog-to-digit converter in crate B and are analyzed by a corresponding subroutine. The greater the amplitude of the signals from AD4 and AD5 the smaller the diameter of the beam on target. The system controls the current of a focusing lens FL by a digital-to-analog converter in crate B so as to keep a predetermined diameter of the spot. On the basis of the delay of a pulse from AD4 relative to the sync pulse, the current position of the spot is determined. It allows a follow-up control of the current shape of a magnetic scan deflector BCD when a scanning beam is used. This makes it possible to keep the required dose distribution on the target. The duration and frequency of the electron bunches, the energy of the beam and the size and location of the electron spot on the target may be determined. There is a supervision program that keeps data from the experiment and presents current information on the monitor screen. It makes it possible to view signals from the

sensors, to test the system, to change parameters of the beam in response to operator instructions and to calculate a dose profile on the target.

#### REFERENCES

1. R.M. White, J.Appl.Phys., 34, No.12, 3559 (1963).
2. B.B. Oswald Jr., D.R. Schallhorn, H.A. Eisen, and F.B. McLearn, Appl. Phys. Lett., 13, No.8, 279 (1968).
3. F.C. Perry, Appl. Phys. Lett., 17, 478 (1970).
4. I.A. Borshkovskii, V.D. Volovik, I.A. Grishaev, et al., Pis'ma v ZhETF, 13, 546 (1971).
5. I.I. Zalyubovskii, A.I. Kalinichenko, and V.T. Lazurik, Introduction to Radiation Acoustics [in Russian], Vishcha shkola, Kharkov (1986).
6. A.I. Kalinichenko and G.F. Popov, Akusticheskii Zhurnal, 36, No.5, 950 (1990).
7. Yu.V. Kapurin, V.T. Moiseev, V.V. Petrenko. Atomnaya Energiya, 63, N6, 400 (1987).
8. Yu.A. Kresnin, G.F. Popov, N.G. Stervoedov. Instruments and Experimental Techniques. 37, N.5, part. 1,p.592-595.(1994).
9. A.I. Kalinichenko, G.F. Popov, V.A. Deryaga, Yu.A. Kresnin, and N.G. Stervoedov. Proceedings of contributed paper of "4-th European Conference on Accelerators in Applied Research and Technology". Zurich, Switzerland. Aug. 29-Sept.2. 1995. p.B-67.
10. V.A. Deryaga, Yu.A. Kresnin, V.V. Petrenko, G.F. Popov, and N.G. Stervoedov. Proceedings of contributed papers of "8-th conference on applied accelerators".Sankt-Peterburg, Russia.sept 26-28.1995.p.145-146.

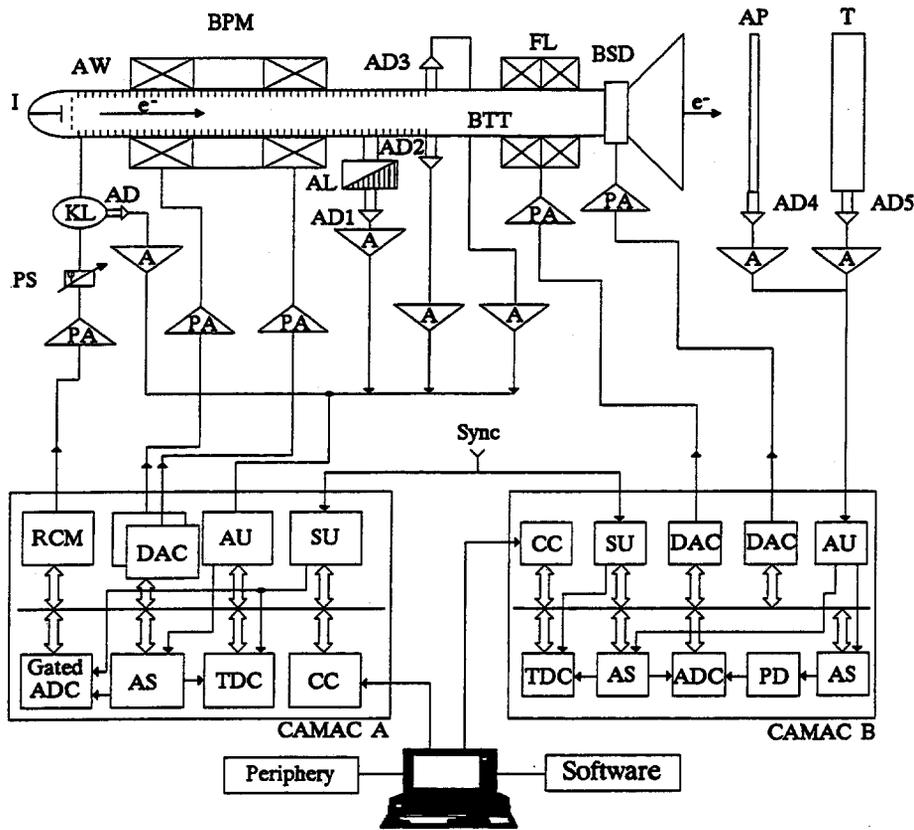


Figure 1: Block diagram of the electron accelerator control system. **AW** — accelerating waveguide; **BPM** — beam positioning magnet; **FL** — focusing lens; **BSD** — beam scan deflector; **BTT** — beam transport tube; **KL** — klystron; **AL** — absorbing load; **PS** — phase shifter; **AP** — acoustic probe; **T** — target;  $e^-$  — electron beam; **AD-ADC** — acoustic sensors; **PA** — power amplifier; **A** — pre-amplifier; **AU** — amplifier unit; **SU** — synchronization unit; **CC** — crate controller; **AS** — analog switch; **DAC** — digital-to-analog converter; **TDC** — time-to-digital converter; **Gated ADC** — gated analog-to-digital converter; **RCM** — relay control modulus; **PDM** — peak detector modulus.

# The PETRA Archive System

W. Schütte

Deutsches Elektronen-Synchrotron DESY  
Notkestr. 85, D 22603 Hamburg, Germany

## Abstract

Currently a new, PC-based control system is being introduced for the high energy particle accelerator PETRA [1]. All relevant data for its operation and control are sent with standardized IPX calls. Any subset of those data can be stored with an archive system at a rate of up to one Hertz.

Archivation\* can be done at certain time intervals, on a specified relative/absolute value change or for values within a specific range. Archivation can be made dependent on the state of the accelerator, such as injection, and on the accelerated particle type. An example would be the storage of values of the proton injection orbit which differed from the last stored injection value by at least half a millimeter. Another example would be the storage the vacuum values that changed by at least 50%. The definition of the archivation is completely given by entries in a database. Archiving can be done in quasi-infinite tables. Tables of a defined depth are treated on a first-in-first-out basis. Both the definition of archiving and the archiving itself are done with Microsoft Access 2.0 as a database. The database is quite hard pressed, but powerful enough to accomplish such a job.

## 1. INTRODUCTION

PETRA is a large lepton and proton accelerator at DESY in Hamburg, Germany. It was commissioned in 1978 with a NORD-computer-based control system. These computers are no longer built. After our staff succeeded in getting the large HERA project at DESY working, a new control system for PETRA was started. This system is supposed to be a prototype for those of our other nine accelerators. Any modern control system should be based on powerful and modular industry standard products with a fast design cycle. It was decided to use Windows 3.1 PCs and a Novell 3.1x file server connected by a LAN<sup>1</sup>. The only communication protocol is IPX<sup>2</sup>. All the applications are written in Visual Basic 3.0, and even the device servers are true Windows applications. For the IPX communication a visual basic extension (VBX) was written [2]. It allows the transmission of integer and single arrays, of strings, of two special formats for hardware access and of a cycle message format. This latter allows a synopsis of the most relevant machine data. Information that is considered to be useful for many users of PETRA is sent by broadcasts. This means that the machine status is available to anyone on the network and that no special request or registration is necessary on the sending PC. The cycle message is special; it contains the most important information of PETRA, as well as the time and a unique cycle number. It is sent at a rate of one to two Hertz. Most broadcasts are sent as a direct response to the cycle message. Changing the rate of the cycle message changes the network traffic and the reaction time of the entire system.

## 2. DEFINING THE ARCHIVATION PROCESS(ES)

An archiving system was built as part of this control system. Any defined subset of the broadcast data can be stored under defined conditions into the Access database by as many servers as needed. The definitions themselves are also kept in such a database. The following examples could be realized by the archivation system just by editing database tables: storage of the DC beam current every minute; storage of the last 1000 vacuum pressures that changed by at least 50%; storage of the values of the proton injection orbit which differed from the last stored corresponding value by at least half a millimeter. Each of those examples is called an **archivation**. The set of all the archivations is called the **PETRA archivation**. Each archivation is represented as a record in an archivation table that has specific relations with other tables to precisely define the details.<sup>3</sup> Archivations have the following information: a unique archivation name, a reference to the archivation server, the data definition, the particle type (optional), the machine status (optional), a history filter reference (optional), a data taking interval (optional), the maximum number of stored values of a particular type (optional), a reference to the data output location and auxiliary information (partly optional). We first look at how the data definition of the archivation is realized with the help of the detailing tables.

### 2.1 What information to store and how to get it?

Any information on the PETRA network segment is uniquely defined by an I.D., a format type and an index. The I.D. is an eight character string sent with all of our IPX messages. An example would be "PX\_All" for the positions from all horizontal beam position monitors. The format would be a string of floats, one for each measured position. The index would be the i<sup>th</sup>

---

\*EDITOR'S NOTE: The author uses the word archivation throughout this paper. This word does not exist in standard English. It has been suggested that it be changed to the gerund "archiving" or the slightly pompous "data storage." Instead I have decided to keep it as used by the author and feel that the reader will understand what is meant.

<sup>1</sup>The other accelerators get each at least one LAN and file server of their own. Net segmentation should reflect the division into accelerators.

<sup>2</sup>This means that there is no IP on the Net.

<sup>3</sup>See figure 1 in the Appendix.

element. For example the 26th element contains the position "SL 141"<sup>4</sup>. The information can be most naturally managed in a master table of IPX I.D.s and a detail table with the interpretation of all indices for each format.

Additions, removal and significant change in the equipment lead to a different way of using the IPX format. These will result in different table entries. These tables only reflect the format at the time they are used and not its history making it possible for the archiving system to be able to deal with past years' data. In other words one wants a backward compatible model for the archive data representation.

The I.D.s are not only related to data belonging together, but also to hardware implementation. For example there are 277 getter pumps in PETRA, which are treated by two different device servers, both having different I.D.s for the same types of values. One gets 93 vacuum values for the north half and 164 for the south half of the ring. The archiving system should be able to treat all those values simply as PETRA vacuum getter pump pressures. In other words one wants the ability to have a logical archive data representation.

Both arguments lead us to use a separate internal and external model for the data. All the getter pump pressures have the internal data name "GP". They are all of data type single and have the unit mBar. In a detail table there is a list of all components belonging to the data name "GP" and an external key into the list representing the data sent by the IPX. If a pump is removed one only has to remove the foreign key entry and the corresponding record in the IPX I.D. detail table<sup>5</sup>. The archiving process involves selection and reordering of data.

In the internal model there are two types of data, scalar and vector. The DC beam current is scalar; at any point in time there is only a single value representing it. The 277 getter vacuum pressures are vector data. For data marked as scalar in the master table the corresponding index of the component will not be stored.

## 2.2 When to Store the Information?

It is not desirable to archive or even look at all the specified information each time it is sent into the network. For example, the getter pump pressures are only interesting to look at once every ten seconds. This is accomplished by an entry in the master table of IPX I.D.s. Such a strategy saves data processing power at an early stage. Some information is only interesting if there is the right kind of particle in the machine. For example, a fine grained archiving of the lifetime is necessary only when there are leptons in the ring<sup>6</sup>. For the particle type one has the choice of: any, protons, leptons, electrons or positrons in the machine. Some information is only interesting when the machine is in the right state. For example a proton injection orbit requires the machine to be in injection mode with current. For the machine state one has the choice between any or injection, ejection, ramp up, ramp down. Each choice can be supplemented by "with current". It is simple to implement more choices, but it requires to code changes in the archiving server program.

Assume the particle in the machine is right and the machine is in the correct state. Still one does not want to store all values at one Hertz. The simplest way to reduce the data volume is to introduce fixed intervals for storing, e.g. storage of the machine current every sixty seconds. Allowed intervals can range from one to 32,767 seconds (a little more than nine hours); "no fixed interval" can also be used. An attempt is made to read the data at the same relative point for every fixed-length interval. Data with a one minute interval is always read at the exact minute. Data with a twenty minute interval is read at the exact hour, the exact hour plus twenty minutes and plus forty minutes. Standard target times make it easy and efficient to correlate different channels, such as current, lifetime and energy, since the data taking times should be very nearly equal for equal intervals.

Fixed interval data taking has two potential problems. Firstly, no data are taken between the intervals. Secondly, some values, such as particle type, remain constant for long times. An appropriate choice of interval involves a trade-off between redundant and stale data. Therefore an alternative means of deciding when to store a value is provided. This is the so called **immediate history filter**, whereby any datum can be specified for storage when its value changes. This is ideally suited to the particle type. Numeric data can be specified for storage when the value changes by a minimum amount (good for "linear channels") or by minimum percentage (good for "logarithmic channels")<sup>[3]</sup>. A typical linear channel is a beam position; a typical logarithmic one a vacuum pressure. Also, the range of tracked values can be restricted with optional minimum and maximum values. The first time a value exceeds the limits, it is stored,<sup>7</sup> and no more values will then be stored again until one is read which is within its limits. **Range restriction** allows the data of a DC current archiving to be ignored, with 10% resolution, when there are no particles in the machine. The noise of the transducer is usually not interesting. Range restriction is also a low level implementation of limiting data to their physical range. The filter is implemented in such a manner that one can define a default, and then add different values

---

<sup>4</sup>South Left at 141m

<sup>5</sup> It may be the case that one is interested only in the pressures of pumps which are exposed to much synchrotron radiation. In this case one would have to invent a new data name, such as "GPhighSy" and add in the detail table the list of corresponding devices. It is possible to combine any data, as long as they are available on the same segment of the network and it have the same atomic data type.

<sup>6</sup>PETRA is used as a storage ring for HASYLAB synchrotron radiation experiments.

<sup>7</sup>Though one is by definition not interested in the value itself, it is necessary to store it (or any other non-valid value) to flag that the previously stored one is no longer valid.

for some special channels. This allows, for example, filtering noisy vacuum pumps and also being able to refer to an "UHV50%"<sup>8</sup> filter as a single entity in the definition of the archivation.

Another problem to solve for history filtered data is signaling that a particular archivation stopped. This might be due to the breakdown of the corresponding device server or to a pause of the archivation server itself. Here the convention is to fill a Null<sup>9</sup> value right after the last known valid time. This is done in the following situations: firstly, an IPX message of with a particular I.D. does not occur within a specified time-out period, and, secondly, the archivation server is shut down in an orderly manner. At start-up of the archivation server, it is checked that all entries end with a Null. If one does not the Null is added, with a timestamp, immediately following the last valid datum. Important operations, such as shut-down of an archive server, and important incidents, such as a lack of IPX messages, are logged. Normally it is sufficient to look at the data to get a valid picture of what has occurred.

### 2.3 Where and how to store the information?

This decision was heavily influenced by the use of Access and not using either direct files or a full featured client server database. We use a table for each archivation, the table name by definition being the archivation name, and the only choice remaining being how to divide the tables over different databases. This allows control of the size of different database files and keeps some logic in their organization. The table format consists of timestamp, data index (only for vectors) and value. The timestamp is the time at which data was taken, as given by the cycle message. It is coded in seconds starting from a fixed time. As opposed to the UNIX timestamp, it does not take into account daylight savings times, resulting in loss of one hour of archivation data each year. The one table per archivation approach is viable because all data therein are of the same atomic type. Conversely, in Access very long tables do not perform well.

Any database which is corrupted<sup>10</sup> is automatically repaired on restart. Any nonexistent database files or tables will automatically be created, including indices. The counting of the archivation channel entries is done with a special table at archivation start-up. This table is computed using the data themselves. Any first reference of an element will be looked up in the table and following references use counter arrays in main memory. Since all archivations use exclusive write privileges on a table, this leads to efficiency and consistency. The system dynamically adapts to an increase in the specified table depth but not to a corresponding decrease. The archivation server appends and edits records but never deletes any.

## 3. PERFORMANCE, TOOLS AND OUTLOOK

During the last six months we collected roughly 75 MB of data in 35 archivations distributed over nine database files. Thus far one archivation server PC is sufficient. Ninety-two percent of all messages are handled within one second, 99% within five seconds and 99.9% within nine seconds. The archivation server generally runs reliably for weeks at a time. It is shut down for manual backup roughly once a week. As long as it is running, the database files are open and cannot be copied under Novell. In the future we will run update queries on a different server for that purpose.

A tool is also provided for viewing the data. The user chooses an archivation<sup>11</sup> and time scale and the tool tries to retrieve 300 corresponding data values. It is important to visualize gaps in data taking (for fixed time interval archivations) and invalid data (Null). Gaps lead to dashed lines on the display, times of no data have no line at all; the last and first valid points around a gap are emphasized by circles; values out of the plot range are treated as non-existent. Invalid values frequently appear due to the device server signaling invalid data. The archivation system does not translate private invalid values, in particular not to Nulls. Up to eight archivations can be viewed in one plot in parallel and up to ten plots are allowed. All values are correlated against time and not against each other.

Future tasks include a program interface to the archivation. It is hoped that with Visual Basic 4.0 a versatile OCX can be written. We are still looking into the possibility of using a full client server database. One advantage would be relatively easy access to the data from other computer platforms and the ability to handle large tables in a multi-user environment in a speedy way. We would consider using tables of atomic data type and adding a column with a unique archivation number.

## 4. REFERENCES

- [1] F. Peters et al. "The PETRA Control System", to be published
- [2] M. Peters, to be published
- [3] Kay Rehlich, private communication.

---

<sup>8</sup>An Ultra High Vacuum filter that requires a relative change of values of 50% (default) and allows a reasonable pressure range ( $10^{-15}$  to  $10^{-3}$  mBar).

<sup>9</sup>Special database value with a meaning of "undefined", which is different from any possible valid entry.

<sup>10</sup>Reasons are archive device server hard failures, such as loss of power or major Ethernet problems, in the middle of the transaction.

<sup>11</sup>For vector archivations one also has to specify the component (for example vacuum pump "NWL 127")

## 5. APPENDIX

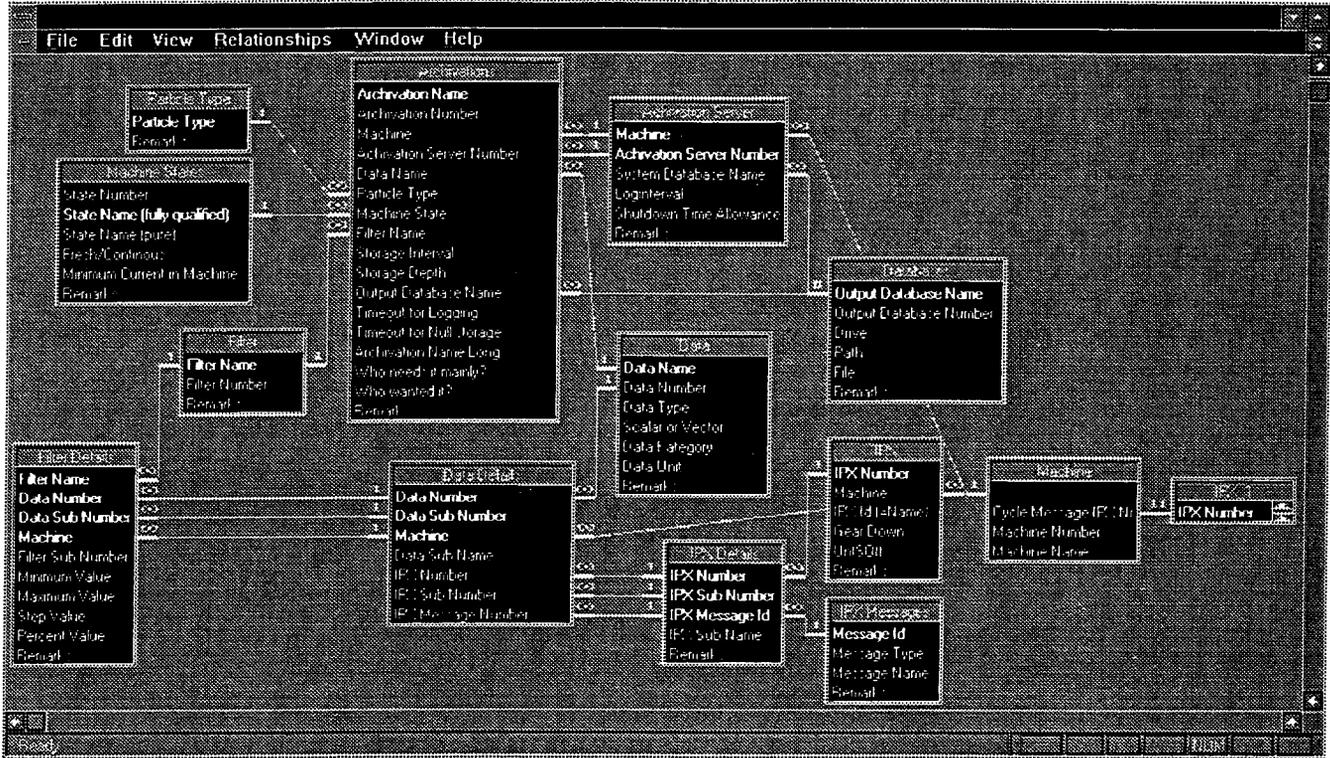


Figure 1: Definition database layout.

The central table is the archivations table. Data and data details refer to the internal data model. IPX and IPX details refer to the external data model. The filter table gives names to history filters, which are defined in the filter details table. After any change a preassembled table is created for the archivation servers (see figure 2).

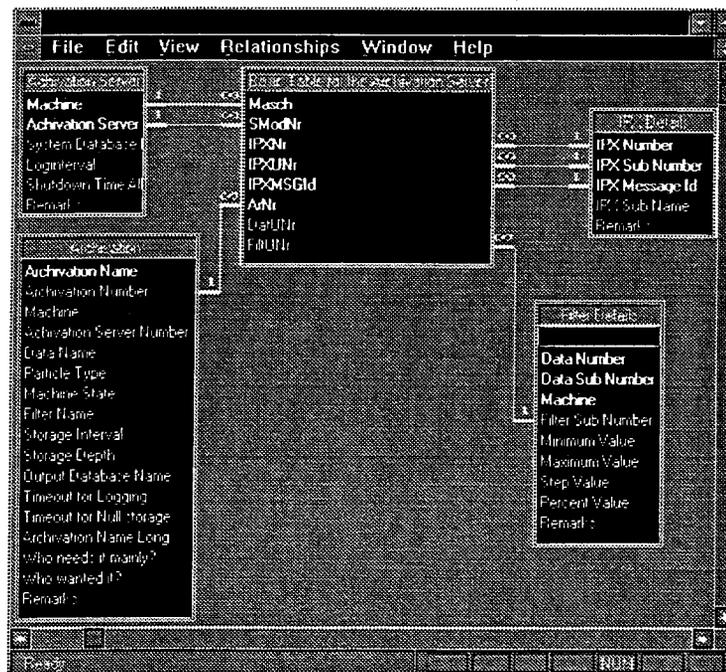


Figure 2: Definition of the preassembled table for the archivation servers.

This table is preassembled for speed of initialization. It is also easy to implement defaulting (for filters) in such a way.

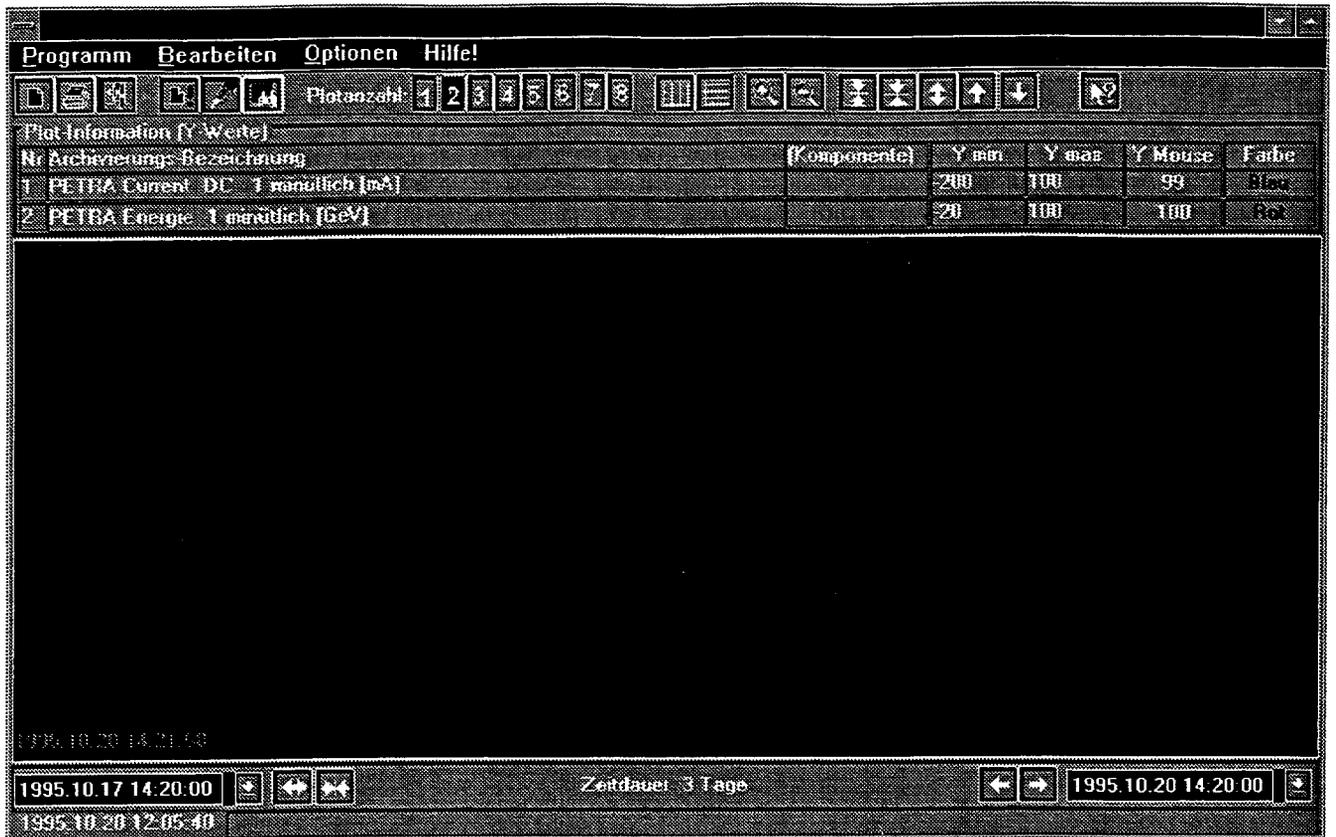


Figure 3: Viewing tool for the database.

As an example one can see the PETRA current (above) and the PETRA energy (below). The data were archived at a one minute interval and three days are shown. In the energy curve one can see a break in valid data. It is visualized by an interrupted curve surrounded with data points highlighted with circles.

# Accelerator Study System Based on RISC/UNIX

M Shirakata, H. Sato, T. Toyama, K. Marutsuka and D. Arakawa  
National Laboratory for High Energy Physics (KEK)  
1-1 Oho Tsukuba Ibaraki 305, Japan

## INTRODUCTION.

A mixed system using RISC/UNIX and LynxOS can provide advantages for control systems for accelerators, due to its flexibility, calculation power and cost/performance ratio. A workstation-based accelerator study system is planned for the KEK Photon Source (KEK-PS). The expected performance will provide a more sophisticated environment for accelerator studies and operation. The unified data acquisition and control system should shorten the time needed for the accelerator studies and should produce high quality beams as a result.

## COMPUTER NETWORK.

The computing environment is a distributed system, connected by a network. We can now use a sophisticated network environment easily and at a relatively low cost. Since the accelerator is a large machine, the equipment for monitoring and control has to be distributed around it, as the signals from beam monitors are small compared with noise from such as pulsed magnets, and long cables would make the situation worse. Such signals are collected and digitized in the auxiliary rooms round the ring and transferred to the centre through the network.

The accelerator needs a highly-networked data handling system. In high energy physics, much effort has been spent in developing fast data transfer systems, due to the gigantic data sets from the large experiments which need to be transferred and processed. The spread of more general networks has brought the requirement for ever higher-speed data transfer and the accelerator requirements can now be satisfied by commercial firms. For the data links, Ethernet normally has sufficient speed, is reliable and is relatively cheap. If the speed is insufficient it can easily be upgraded to a faster network, such as FDDI, by changing cards. It is also worthwhile to use the RISC/UNIX system in computers connected by Ethernet for accelerator data transfer, rather than developing a special system oneself.

At the KEK-PS, the control system consists of a VME/satellite computer system [1], running VersaDOS and connected by a MAP network, which gives a data transfer rate of a few tens of kilobytes per second. Although VersaDOS was developed for distributed real-time computing environments, there is no plan for its updating and the MAP transfer rate is too slow for the increasing requirements for data transfer, so it should be replaced in the near future. A good candidate for this is UNIX on an Ethernet system because such a combination gives flexibility, network advantages and a graphical user interface. Early in 1995, the LANs at KEK were almost entirely replaced to cope with the expected future requirements. Subsequently, some workstations have been introduced to work in parallel with the VME/satellite system, as shown in Fig.1.

## APPLICATIONS.

Partial replacement of the monitor and control system has begun. We started by making a communication port to the present VME system from the Ethernet environment. It became possible to transfer commands and data between UNIX and VersaDOS through shared memory, but this is only temporary and is not perfect. This is the only port connecting the two systems and its capacity is insufficient for continuous monitoring or any application that requires a large data transfer. In addition, some VersaDOS commands cannot be invoked through this system.

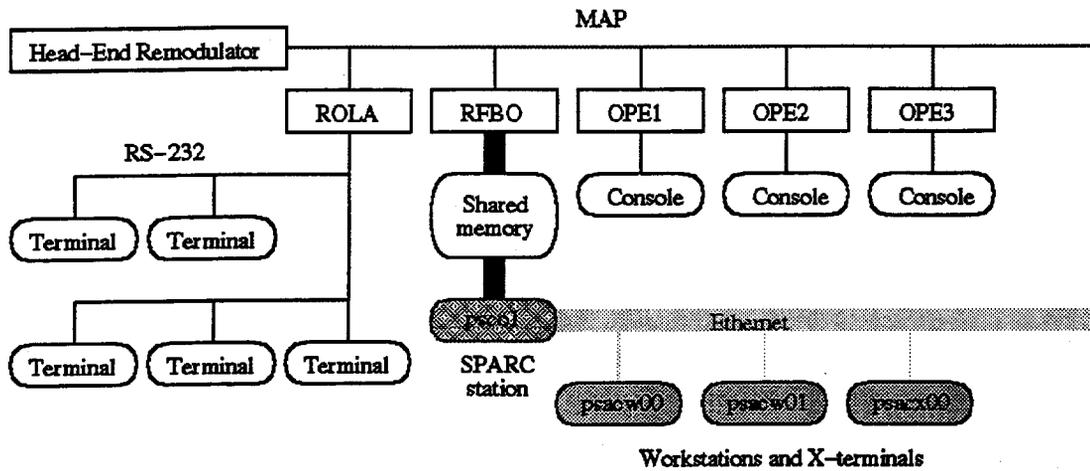


Figure 1. Schematic diagram of KEK-PS control

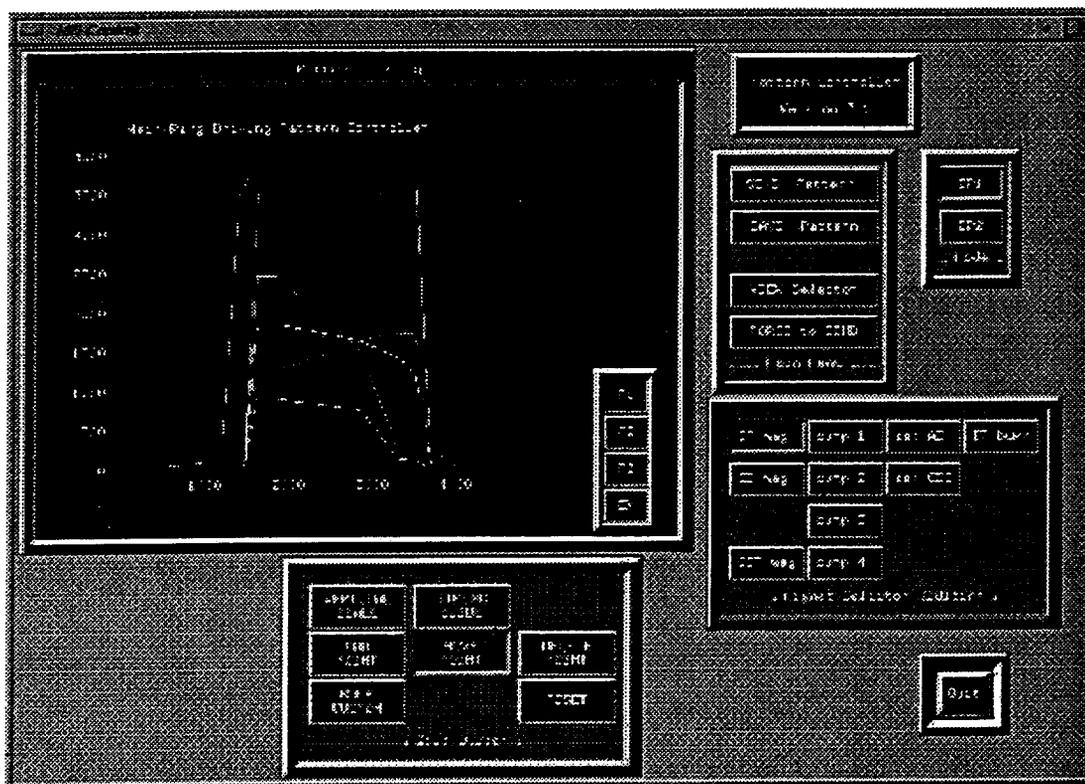


Figure 2. GUI of main ring magnet pattern controller

Magnet excitation pattern can be changed graphically with mouse or direction keys.

The UNIX applications first took the user interfaces from the existing system. After the transformation of the GUI, subsequent replacement of the hardware can be made transparent to the operator. The applications described below have become much more effective, both during operation and machine studies. Graphical interaction already speeds up parameter changes and increases operator efficiency, even though they are not yet complete.

#### *Magnet Control.*

The KEK-PS magnet control system has been transferred to the UNIX system and partly unified. Fig. 2 shows the GUI display for the magnet cycle pattern controller.

The power supplies for the magnets are distributed in the auxiliary rooms and the excitation patterns are given from the DAC modules located in the VME/satellite system. Ethernet has not reached all the auxiliary rooms yet, so most of the DAC modules are left on the original system. Although only the user interface was replaced, operation became easier and more efficient, especially for the magnets for slow beam extraction. With the screen editor, the operator can edit the magnet pattern directly on the display. It is easy to learn and easy to use. Since all the functions were transferred from the former interface, the operators do not have to interact with VersaDOS.

#### *Closed-orbit correction.*

The closed-orbit correction (COD) system [2] has also been transferred to UNIX. The KEK-PS main ring has 56 beam position detectors and 28 steering magnets, in both vertical and horizontal planes. Taking into account the lattice parameters, the required correction fields are calculated from the orbit position data. In the early days, although the orbit was measured at 56 points, it was difficult to solve the 56x56 matrix to obtain the correction fields in a restricted time and memory space, so only half of them were used. Now it is easy to solve it, resulting in an orbit control to within 1 mm, as shown in Fig. 3.

During accelerator studies, it is often necessary to make a bump or correct the orbit due to a change of operating conditions. Unfortunately, this is not simple with the existing system, since the beam position monitors and the steering magnet power supplies in four auxiliary rooms are controlled by the VersaDOS system. It will take time to convert the whole system to UNIX, although most of the modules are available.

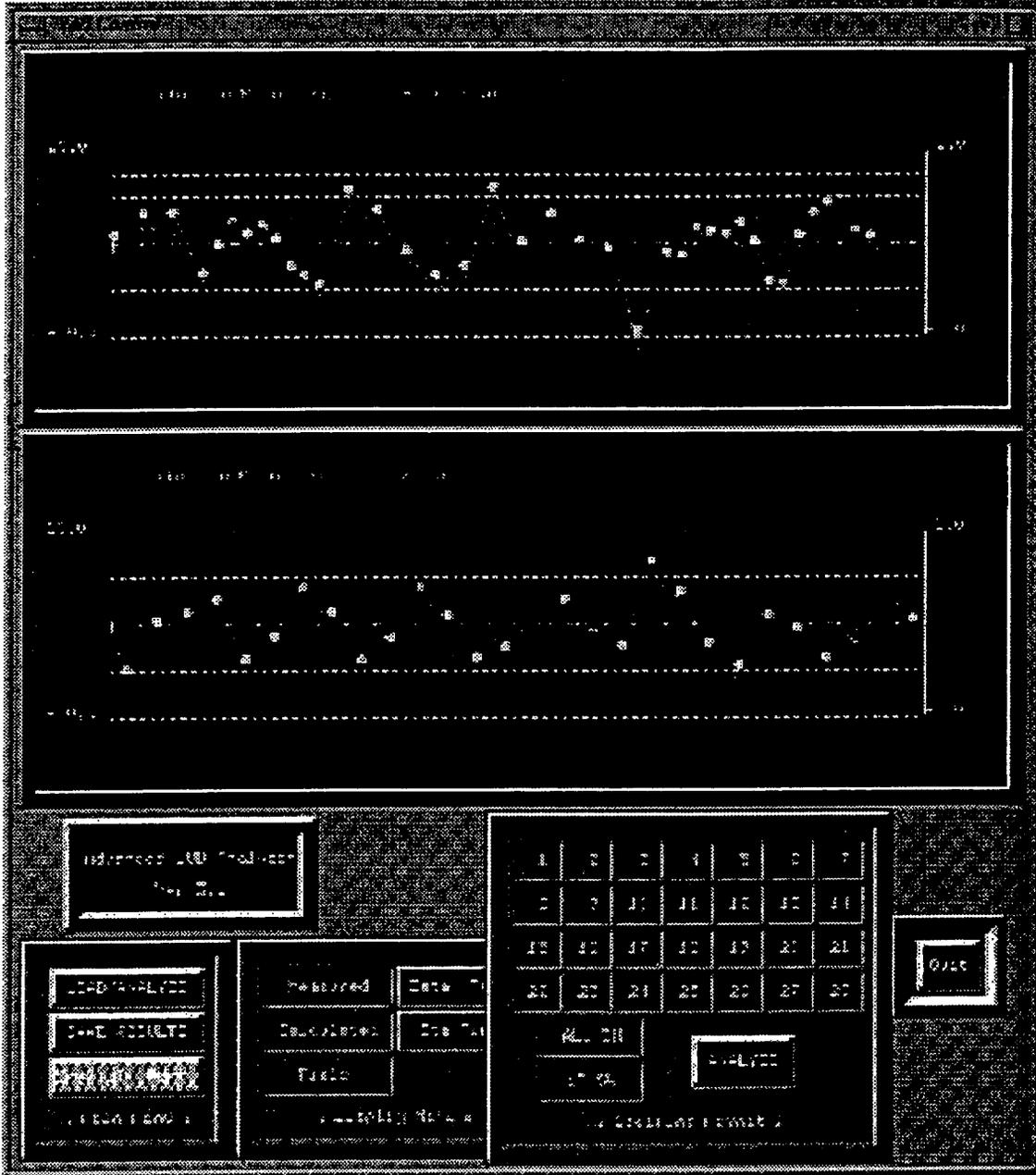


Figure 3. GUI of COD analyzer

Yellow points: Measured orbit position  
 Red points: Calculated error fields  
 Green lines: Reconstructed orbit from error fields

### *Injection Error Monitor.*

Unlike the applications described above, the injection error monitor [3] has been transferred entirely to the UNIX environment.

The standard data acquisition system UNIDAQ [4], which was based on LynxOS, was also available on HP742rt/HP-RT. HP742rt is a VME board computer and HP-RT is a Hewlett Packard operating system. The requirement for read-out capability increases with the length of the accelerator cycle and the data size. A RISC/UNIX system has a high cost/performance ratio for data analysis and networking, but the standard UNIX system is not suitable for fast data acquisition because it does not have preemptive scheduling, which is required for a real-time system. A number of vendors are now offering real-time UNIX which satisfy the above requirement.

The speed of data acquisition from CAMAC by HP-RT on a UNIDAQ system has been measured [5] to be:

CAMAC access time - 20  $\mu$ s for single action  
(Block action not yet supported)  
Interrupt handling - 65  $\mu$ s.

The injection error monitor requires the measurement of the beam position at two points in the ring, where the betatron phase advance is not a multiple of  $\pi$  in order to get the beam trajectory in horizontal and vertical phase space.

A fast position monitor is shown in Fig. 4. It used four wall-current type position detectors with signal pickups. The beam signals are stored in dual-channel ADC modules, LeCroy 6841, which have a 128 kB memory for each channel. Four of these are used for position detection at two positions in the two planes. The signals from these are transmitted to the HP742rt through the VME/CAMAC bus connector and analysed to give the beam position. The data is reduced and transferred to the UNIX system, using NFS.

The cycle time of the main ring is about 4 seconds. A maximum of about 800 turns of phase space trajectory can be traced per cycle, as shown in Fig. 5

The maximum number of traces is limited by the data transfer rate from the LeCroy 6841 to the HP742rt, the signal processing software and the memory size of the ADC modules. In practice, only several tens of turns are necessary to define the injection error and calculate the correction required. This monitor has played an important part in the injection studies at the KEK-PS main ring.

### SUMMARY.

The evolution in computers and networks has brought many advantages for accelerator control systems. The time has come for the control system for KEK-PS to be upgraded, and the work has been started. The workstation-based control system provides useful accelerator study tools. Three of these have been implemented: the injection error monitor, the closed-orbit analyzer and the magnet pattern controller. The last two still have some inconveniences resulting from the earlier control system. It is hoped to convert further to the UNIX system in the near future.

### ACKNOWLEDGMENTS

The authors would like to thank Dr. J. Kishiro, M. Yoshii and Y. Shoji for their support. We also wish to thank Dr. S. Machida for his useful discussions and suggestions. We express our gratitude to the KEK-PS accelerator group.

## REFERENCES

- [1] VME-COMPUTER BASED CONTROL SYSTEM FOR THE KEK PROTON SYNCHROTRON, T. Katoh et al, Europhysics Conference on Control Systems for Experimental Physics, Villars sur Ollon, Switzwerland, 1987, and KEK Preprint 87-91.
- [2] Y. Shoji et al, Acc. Sci. & Tech. 1991, p332.
- [3] Injection Error Monitor for KEK 12 GeV PS, M. Shirakata et al, KEK Report 93-11, November 1993.
- [4] UNIDAQ Document Set, SDC-93-573, UNIDAQ collaboration, SSC Lab., 1993, UM-HE-93-29.
- [5] VMEbus-based computer and Real-time UNIX as infrastructure of DAQ, Y. Yasu et al, KEK Preprint 94-19, May 1994.

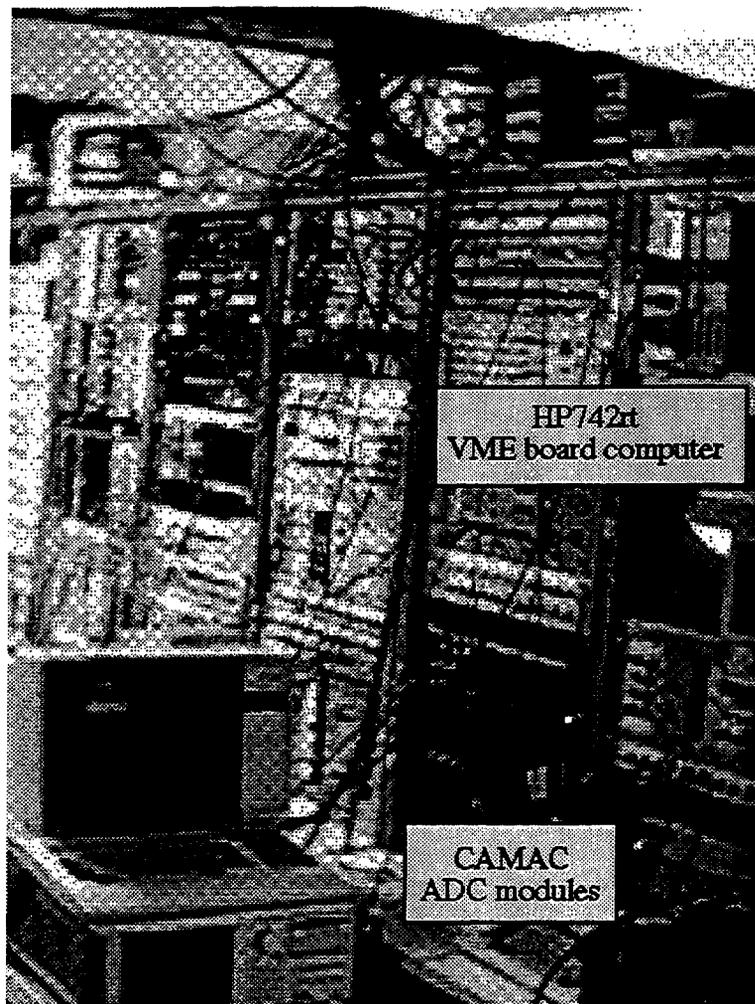


Figure 4. Fast position monitor

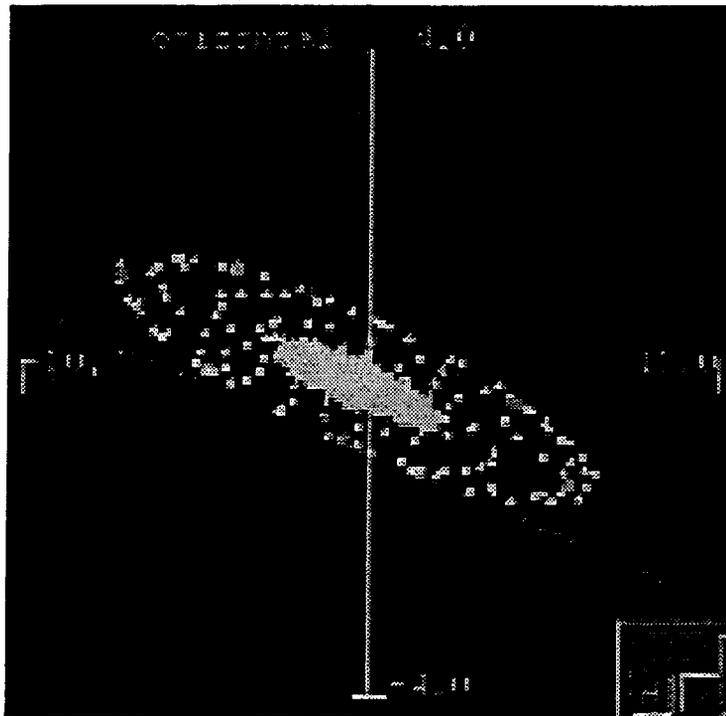


Figure 5. Horizontal betatron oscillation

Color points: Trajectory in phase space  
(First 10 turns are emphasized by other colors.)  
x-axis: Position [mm]  
y-axis: Angle [mrad]  
Lines: Controllable coordinates by injection  
magnets.  
Line cross indicates the beam injection.

# BUS AND RING SYSTEM INTERCONNECTIONS FOR DATA ACQUISITION AND CONTROL

V.I. Vinogradov, INR RAN, RF.

## Abstract

The traditional bus systems based on 8/16 bit microcomputer- and register-oriented interfaces for data acquisition and control are analyzed. Bus- and nonbus-type system and network interconnections are compared. The possible transition from traditional bus to indirect point-to-point interconnections for control systems is discussed. The low interaction-rate nodes of message-passing systems can be exchanged for high-rate ones in operating control networks, with direct control access to distributed equipment based on shared distributed memory subsystems and a new generation microprocessors ( $\mu$ P).

Advanced distributed-memory data acquisition and control systems based on a new generation 16/32-bit general purpose microprocessors and Digital Signal Processors (DSP) with RAMLINK-type (SCI-like) distributed memory interconnections are proposed and discussed. Advanced distributed  $\mu$ P systems are proposed and analyzed on the basis of a new generation of DSP and distributed memory.

The new international standards for distributed data processing and memory interconnections are discussed. But for control systems an international standard for fieldbuses has not been selected after more than 20 years, though there are some national standards developments. In this case register-oriented bus interfaces on traditional message-passing networks can be changed to distributed shared-memory direct-access interconnections.

Some new decisions for fieldbus and instrumentation input-output subsystems can be based on bus and ring interconnections, which must be developed according to measurement, acquisition and control (MAC) requirements, using new 32-bit address fields with direct access to distributed memory, data block transfer, a simple data transfer protocol and connecting of different types of existing and new devices. The new generation of 16/32-bit  $\mu$ P- and DSP-based systems, with high speed links, can be a new basis for development of distributed memory systems with symmetrical or non-symmetrical structures. The ways of developing such a system are presented and discussed.

## 1. MICROCONTROLLERS, MICROPROCESSORS AND DSPs

Digital control and data acquisition and processing real-time systems address a need for monitoring and control of a large number of parameters. Any simple control system includes a controller which generates actuator commands, received from an operator, and readbacks, provided by sensors. The controller processes the signals to achieve a desired response and can thus modify the frequency of response of the system. The computation element can be realized with analog or digital components. Analog controllers process a signal, can be used for very high bandwidth systems and can be realized with inexpensive components. However they suffer from component aging and temperature drift and are limited to very simple algorithms.

DIGITAL CONTROLLERS approximate them, but are not affected by component aging and temperature drift and they provide stable performance with sophisticated techniques and algorithms. Programmable microcontrollers make it easy to upgrade, maintain and design systems for optimal and adaptive control. The choice of  $\mu$ P is critical in the performance and behavior of a digital controller. Available choices are microcontrollers, DSPs and general purpose  $\mu$ Ps, including high cost RISC architectures. A control system requires real-time signal processing. The signal and gain coefficients must be represented accurately without loss of resolution for the smallest and largest magnitudes. Floating-point arithmetic may be necessary if gain coefficients and signals are time-varying or with large dynamic ranges [1-3].

High performance is required if the sampling of signals at discrete time intervals requires high speed processing. The  $\mu$ P should finish the processing as soon as possible. Thus very fast instruction cycle and multiplication times are needed. Peripheral integration is important from system cost, simplicity of design and board-size points of view. Typical peripherals on such chips are I/O pins, parallel and serial interfaces, DACs and ADCs. Digital microcontrollers are primarily designed to replace hardwired logic for data acquisition and decision making in control systems. The special architecture and high performance of Digital Signal Processors (DSP) make it possible to implement a wide variety of modern Digital Control

algorithms and Data Acquisition systems. The most popular DSP's are from Texas Instruments, Motorola and Analog Devices.

MOTOROLA has created compatible families of  $\mu$ Ps since 1974, beginning with the 6800 family of microcontrollers, then next the 68000 line and now PowerPC RISC  $\mu$ Ps. Motorola's DSPs, based on the DSP56000 architecture, have compatibility across an entire product line. All family members benefit from the same extended triple-bus Harvard architecture. On-Chip Emulation (OnCE) was first proposed by Motorola and is useful for field analysis and repair. Wait and stop modes reduce power consumption. The Do loop instruction directs a  $\mu$ P to repeat a series of operations a specified number of times without any branching time.

Motorola offers 3 closely related families of DSP: DSP5610X, DSP5600X and DSP960X, along with development tools and peripherals. DSP56100 (16-bit) is optimized for the standards of digital cellular communications. The DSP56000 (24 bits) has become the standard for digital audio applications. The new member, DSP56004, is used in automobile audio systems. The DSP96000 (32 bit) family combines color graphics, sound and communications for medical, industrial and other applications. Previously limited to military, aerospace and scientific systems, DSP- technology is soon be used far more widely. The DSP56000 family is designed as a 16-bit programmable core  $\mu$ P for DSPs, supplemented with configurations of memory and serial communications interfaces, timers, a host interface and in-chip coder-decoders for analog signals [1].

TEXAS INSTRUMENTS produces several on-chip DSP families, including field-programmable microcontrollers and fixed- and floating-point  $\mu$ Ps. Field Programmed Modules (FPM) are Multi-Time Programmable (MTP)  $\mu$ Ps with EPROM and EEPROM all on a single chip. Ten year data retention, 10,000 write cycles and an internal charge pump that generate a 12V programming voltage means that the end product can be adapted. The TMS370 family has more than 10 FPM members. CMOS technology gives low current consumption. The 256 bytes of EEPROM on a TMS370C710 have the advantage of being modifiable in the field. Coming from a microcontroller family these devices are well suited to industrial applications. The sensor signal  $\mu$ P family TMS400 was developed to serve sensor signal conditioning applications with high accuracy and precision, where average system power must be in the microwatt range [2].

The FIXED POINT 16-BIT DSPs from TI give the lowest cost per MIPS in their class. The TMS320C5X generation of DSPs gives 50 MIPS of performance at 5V and 40 MIPS at 3V. A parallel logic unit provides high speed without modifying the ALU status or registers. Software wait states provide an external interface for use of slower off-chip memory and I/O.

The FLOATING POINT 32-bit DSP TMS320C30 generation integrates system control and intensive math functions for fast data movement and high-speed data processing. Powerful instructions, parallelism and buses provide flexibility in performance up to 50 MFLOPS. A high degree of parallelism allows one to perform up to 10 operations in a single instruction cycle (for example 2 data accesses, a multiply, an ALU operation and a DMA). The TMS320C32 is an enhanced low cost 32-bit DSP manufactured in 0.72  $\mu$ m EPICS CMOS technology. The C32 includes all the features associated with a general purpose controller (similar to a RISC/CISC  $\mu$ P), but also provides additional ones. The TMS320C40 generation of DSP is effective for intensive parallel data processing in real time and has on-chip communication ports and support for shared global memory and its own pair of extended buses which give direct processor-to-processor accesses. The device provides scalability, fault tolerance and reduced bus saturation.

ANALOG DEVICES INCORPORATED (ADI) has digital signal processing experience, including Mixed-Signal Peripherals (MSPeripherals) and Mixed-Signal Processors (MSProcessors). In 1982 ADI introduced the first DSP in CMOS technology with fixed- and floating-point building blocks, program sequencers, data address generators and register files. The first single chip CMOS DSP ADSP-2100 was introduced in 1986. The company defines a digitally integrated approach to analog systems of signal processing as a first step in a natural evolution in the area of Microcomputers ( $\mu$ C). There are two compatible ADI signal processor families: the 16-bit fixed point ADSP-2100 and 32-bit floating point ADSP-21000s. High performance serial ports (SPORTs) are bi-directional, double buffered interfaces and feature user configurable clocking, framing and word length. A Parallel Host Interface Port serves as an asynchronous interface between the signal and host  $\mu$ P, acting as a block of 8 dual ported registers (8/16 bit interface). An ADSP-2181 can respond to 11 interrupts (up to 6 external). Two Serial ports, SPORT0 and SPORT1, allow multiprocessor communications with word length from 3 - 16 bits (bi-directional) [3].

THE FLOATING POINT ADSP-21000 family has a complete set of arithmetic operations, including min/max, y/x etc. The parallel 9 port general purpose data register file transfers up to 9 operands to and from the computation units and memory. The program sequencer supports zero overhead looping, single cycle setup and exit for multiple program loops and delayed and nondelayed branching. The DSP-21020

handles 32-bit IEEE floating point, 40-bit IEEE floating point and 32-bit fixed point data formats. System tests and on-chip emulations are provided by an IEEE JTAG boundary scan serial port. The ADSP-21010 is a low cost 32-bit version of the ADSP-21020.

The SUPER HARVARD ARCHITECTURE COMPUTER ADSP-21060/62 family (SHARC) consists of 32-bit DSP microcomputers optimized for high performance applications, built on the same ADSP-21000 core (system on-chip), adding a dual ported on-chip SRAM and integrated I/O peripherals supported by a dedicated I/O bus. ADSP-210xx have a 25ns instruction cycle time operating at 40 MIPS. The SHARC core is compatible with the ADSP-21020.

## 2. BUS AND NON-BUS STRUCTURES

There are bus and nonbus approaches for data acquisition and control architectures. The bus one is based on parallel  $\mu$ P-system interconnections for SMP; the non-bus one can be based on switches and SCI standards for Massively Parallel Processing (MPP), Supercomputers and high-speed DACs.

CROSSBAR SWITCHES FOR DATA ACQUISITION can provide [1] very high-speed data processing on distributed systems and networks. There are packet switching, message passing and shared memory  $\mu$ P-systems for pipelined data processing and MPP. A crossbar interconnection is one of the ways for supporting multiple  $\mu$ P's sharing common memory having access time comparable with  $\mu$ P cycle times. As seen in the results of AT&T's Crossbars, these systems allow one to utilize maximum configurations and provide up to 30% improved performance over normal switching systems. The probability of successful connection of a  $\mu$ P to any memory unit is a function of the ratio of the number of  $\mu$ P's (N) and memory units (M), which function decreases to some asymptotic value as N/M. Crossbar-based systems provide, for instance, 15% better CPI value (the number of cycles per instruction) for four individual  $\mu$ P's having four memory units, as compared to similar bus-configured systems. Switch based systems can be constructed as either small or large networks with packet switching [4].

SMALL-SCALE NETWORKS are useful in a wider range of systems than MPP, particularly applications in which I/O is dominant. Fast buses are not very cost effective because every bus slot must have all the high-speed electronics, though they may use it only a fraction of the time. In small systems many parts can be used simultaneously by different processors. Each part can be made to a narrow width and the requirements for a low-cost high-speed interface can be met by full duplex serial connections. One such approach is DS-link (from SGS-Tompson Microelectronics) for Transputers, an asynchronous serial protocol. A serial bit-stream together with its clock is coded onto 2 wires. Each DS-link requires 2 signals in each direction.

A PACKET SWITCH based on STC104 is a 'wormhole' router with a 32-bit duplex DS-link interface, supporting locally adaptive routing. It consists of 32 link slices, each of which provides an input and output port. The IMS 9000 transputer integrates 4 serial link interfaces together with a superscalar  $\mu$ P, FPU, cache and memory interface [5]. The virtual channel processor accepts high-level communication commands from the  $\mu$ P and translates them into sequences of packets. A packet starts with a header and contains at most 32 bytes of data. The STS101 implements a DS-link and provides a parallel interface to other 16/32-bit  $\mu$ P's (synchronously or asynchronously).

A LARGE NETWORK with packet switching is under research and development at CERN for future generation experiments. The IEEE P13335 standard covers the connections, cables, and electrical and logical protocols for serial interconnections, operating at speeds of 100 Mbit/s and 1 Gbit/s over copper/optical fibers. This standard has been developed in the Open Microprocessor Systems Initiative/Heterogeneous Interconnect Project (OMI/HIC). It has a goal of constructing modular, scalable, parallel low-cost systems, which support high-performance communications and transparent implementations of high level protocols. The standard is based on the STC101 and STC104 packet switches, connected to 32 DC bi-directional link ports via a 32\*32 non-blocking crossbar switch [6]. The ROUTER processes up to 200 Mpackets/s with a maximum bandwidth of 300 MB/s and latency of less than 1  $\mu$ s. As result they construct very large networks with different topologies (mesh, grid, tree, Clos) and investigate traffic, delivery of packets and latency. Packets are dispatched according to some predetermined schedule. A cost-effective solution for loading, starting and monitoring the links is the T225 transputer, handling up to 4 links at a time at full bandwidth. The OS-links from that controller are used to connect nodes to the host. One implementation is being made in a study for a second level trigger. The samples of data from a detector are buffered in memory units and passed to a  $\mu$ P farm. A local system uses a TMS320C40 DSP for combined buffer management and local processing. This system is built around DSP-links with switching by DS-links. The TMS320C44 DSP can receive data from up to 4 DSP-links. This special processing machine is under

construction and has 3 stages of routing chips. The first and the third stages of switching have 112 DS-links connected to a T9000. The second stage supports 64 concurrent circuits.

SCI-based modular systems can join many PCs or workstations and clusters of  $\mu$ Ps in Multi-Processor (SCI-LAMP) Architectures. Each node in this network has a local  $\mu$ P, cache, memory and I/O. The local memory is distributed into private and exported portions. Private memory is used only locally, and does not participate in the SCI protocol. Conversely exported memory can be seen by other nodes as part of a single global physical address space supported by SCI.

SCHEDULING of a distributed LAN is traditionally based on a centralized controller (such as a Condor OS) and message passing. Alternatively SCI can provide physically shared memory and cache-coherence among workstations across a network. Scheduling parallel jobs on a network of workstations introduces an additional level of complexity. The fast process migration and choosing of idle  $\mu$ Ps are important to parallel performance. The number of  $\mu$ Ps in multiprocessor systems assigned to an application might have to change when another application arrives or departs or when the degree of parallelism changes within an application. Better utilization than for fixed static partitioning is achieved with dynamic partitioning of  $\mu$ Ps to parallel jobs. If the application cannot adjust its number of tasks during execution, several tasks from the same application may have to share a  $\mu$ P, introducing context switching and other related overhead. An approach to a distributed dynamic  $\mu$ P allocation scheme with even-loading considerations for such jobs in an SCI-LAMP system has been proposed by Li [8].

### 3. SCI-BASED DISTRIBUTED SYSTEMS

Development of the first modular systems, beginning from simple register-oriented interfaces, was based on micro-computers and input-output devices for measurement, acquisition and control tasks. Bus-oriented architecture was the rule for both instrumentation and data processing. A single controller system was the rule for parallel bus (CAMAC-DW/BHW, HP-IB) and serial loop (CAMAC-SHL, HP-IL) structures. Many existing register-oriented protocols were used for measurement (HP-IB, HP-IL), data acquisition (CAMAC), control (FIELDBUS), robotics (BITBUS) and Instrumentation. Input-output subsystems [9] must be developed according to measurement, acquisition and control requirements (short address fields, direct control, data block acquisition and a simple data transfer protocol for connecting different existing and new devices). Next generation systems need a new approach to system construction based on distributed memory and modern microprocessors [10,11].

The new generation distributed microprocessor systems can be divided as follows into main subsystems: a) distributed data-flow processing interconnects; b) distributed memory links; c) distributed input-output/instrumentation links. The traditional means of interconnecting is parallel bus or serial interface. But many new possibilities can be used once one has ring interconnections. The SCI-type systems are one of the best ways to construct a shared memory system. The distributed memory subsystem can be based on a RAMLINK, which is now under development. It is a single controller subsystem with a simpler protocol than SCI, which can be used as an interface for control of object-oriented single-controller systems (fieldbus/fieldring).

The resulting document on SCI was approved by the IEEE on March 18, 1992. The study group concluded that any practical solution would involve the use of packet-based signaling over many independent point-to-point links, would eliminate the bus bottleneck problem and would address the problem of how to maintain cache coherence. The old concept of a single- or few-processor supercomputer has become uneconomical. The cost per computing operation is less with  $\mu$ P technology, so the goal of system research for some years has been to divide problems and spread them across large numbers of inexpensive distributed  $\mu$ Ps. The simplest way is loosely coupled  $\mu$ Ps, using many small ones each with its own memory, that pass data via "message passing".

The general model will be as in a shared memory system, which may be divided into pieces and distributed as are the  $\mu$ Ps. To reduce the effective access time to this memory cache memories keep copies of the most needed data near each  $\mu$ P (copies of parts of shared memory). Thus it is easy to pass data in shared memory. This model needs "cache coherent" protocols to support distributed  $\mu$ Ps with shared memory. Among different ways of interconnecting are ringlets and switch connections. To support cost-effectively efficient coherence protocols,  $\mu$ Ps are expected to manage cache fault handling; infrequent and complex updates would be done in software, but frequent cache updates would be done in hardware.

To reduce the cost of low-end systems, SCI's nodes support a ringlet connection. This requires some extra circuitry for address recognition and buffering in each node, but makes it possible to build inexpensive SCI systems. A wide range of applications can cover the whole range from high-end  $\mu$ Ps and their I/O systems to workstations and LANs. Very efficient LANs have been built using this model, and the usual

layers of network protocols can be added on top for compatibility with existing software. A distributed SCI system shares a 64-bit address space, where the high-order 16 bits are used to route packets to the appropriate nodes. The interconnections can be powerful switches or simple bridges that route packets between ringlets. The interconnects merely forward packets, and need know nothing about cache coherence. The initial links were 1 GB/s (16-bit 2 ns rate, differential ECL) and 1 Gbit/s (fiber optics up to a few km, coaxial cable up to about 20 m).

SCI packet protocols maintain cache coherence, providing for flow control, mutual exclusion, and forward-progress guarantees needed to support multiprocessor system software and applications. Support for message-passing is an important subset of these goals. The CSR work (IEEE Std 1212-1991) was started as part of SCI. A joint approach for Futurebus+ and SCI was developed, a modular metric mechanical packaging standard IEEE Std 1301-1991, which provides physical interchangeability for SCI modules from multiple vendors. A bridge between SCI and VME (IEEE1014) will be one of the resulting commercial products.

SCI is a replacement for computer buses, intended for the next generation of systems ranging from multiprocessors to workstations and PCs. But present data transfer devices are poorly matched to high performance memory systems. The low bandwidth at the I/O of the RAM packages forces one to use a wide array of RAM switch interleaving. The most effective possibility uses LVDS to connect RAMLINK chips in small rings with one controller per ring, which can interface a ring to SCI and schedules all ring activity. The controller uses a split-response packet protocol. The response time may be given either as a precise time or as an upper bound. The RAMLINK protocols simplified the SCI protocol, which can run at 500 MB/s.

Interfaces between the SCI-based tightly-coupled portion of a system and other modular systems (VME, FB+, PCI), serial buses or SCI-type I/O buses are based on bridges. Local I/O devices should share  $\mu$ P memory address space, but peripherals may be located on separate buses. Closely coupled bridges are intended to interconnect multiple workstations and servers within a LAN. A module can consist of several link (interface chip) interfaces (one for each bus) and could be a block in a  $\mu$ P. One could also be a switching component for a wireless bus.

To develop a version of the RT-SCI standard optimized for real-time applications is one of the new problems of distributed systems. For RT systems latency is more important than throughput. The SCI mechanism can be replaced by a strict priority one which ensures meeting application deadlines under specified conditions. Some applications can rely on Rate Monotony Scheduling, which requires that latency be a function of priority. Thus the next levels of IEEE standards are being developed [12-14].

- **P1596.1:** The specification defines SCI/VME bridge architecture for interfacing VME buses to an SCI. This will provide I/O support for SCI systems via VME.

- **P1596.2:** Cache Optimizations for Large Numbers of Processors (kiloprocessors), using the SCI-developed tree-structured coherence directories and fast data distribution mechanisms for SCI systems.

- **P1596.3:** Low-Voltage Differential Interface for SCI, specifies low-voltage differential signals for high speed communication between CMOS, GaAs and BiCMOS logic arrays, used to implement SCI. The project defined new signals that are appropriate to CMOS and other technologies to be differential with 0.25 V swing, centered on +1 V, at 2 ns signaling rate. SCI adopted the serial encoding that Hewlett-Packard devised originally for Serial HIPPI links.

- **P1596.4:** RAMLINK (High-Bandwidth Memory Interface), based on SCI Signaling Technology, permits access to the large memory chips. This work is converging toward point-to-point links at 500 MB/s using a simplified packet protocol.

- **P1596.5:** Data Transfer Formats Optimized for SCI, defined a set of data types and formats that will work efficiently on SCI for transferring data among heterogeneous  $\mu$ Ps in a multiprocessor SCI system.

- **P1596.6:** SCI/RT Study Group, considers ways of getting deterministic behavior in SCI systems for real-time applications.

By this time SCI has been adopted by many computer companies for internal use. Dolphin Server Technology has been formed to market SCI chips, development tools and high level models. Dolphin Interconnect Solutions' SCI-SBus adapter uses programmed I/O for messages smaller than 64 bytes. For larger messages a DMA engine conducts the data transfer. As the message size increases, a peak is reached at 85 MB/s for 64 kB messages. In its clustering of QUAD-P6 nodes, Siemens Nixdorf has announced its use of SCI as well. Among other companies which are ready to use SCI-type systems are Cray, Intel, AT&T and GIS. Cray Research's I/O channel and SCX system interconnect are moving to SCI. A new

supercomputer-class system interconnect provides high performance, scalable, reliable inter-system and system to peripheral communication (as an open standard). The SCX is ring-based and has been enhanced to address the reliability, flexibility and performance needs of distributed supercomputer environments.

Much modern computing consists of a number of modular systems (PC/workstation, clusters of workstations), connected by networks. One of the best possible architectures for the nodes in SCI is that of SMP-based workstations.

## SUMMARY

Using the SCI standard for control systems is still expensive, and the RAMLINK protocol can be used for subsystems with distributed memory on the first level of hierarchical  $\mu$ P/DSP-based control systems. The complex integration of data acquisition and control can be based on the same architecture of distributed memory and direct interconnections between  $\mu$ P- and DSP-based nodes on a network, simplifying programming. But SCI is possible within a collaboration, using for the R&D program experimental objects of a small accelerator. The different networks for experiment data acquisition and control can be based on SCI-networks, which connect ring- and bus-based  $\mu$ P subsystems.

## REFERENCES

1. Motorola M56000, M96000 Digital Signal Processing, Motorola.
- 2.- TMS320C50, TMS320C40 User's Guide, Texas Instruments;
  - MVP: The dawn of a new-era in DSP - introducing TMC320C8X, Texas Instruments.
  - Digital Control Applications with the TMS320; Application Notes. 1991; Texas Instruments.
3. ADSP2100, ADSP21000 notes, Analog Devices Inc.
4. O. Panfilov. Comparative performance analysis of crossbar switches and multiple buses. AT&T/GIS. ICSNET95 - the 11-th International Symposium. St. Petersburg. 1995.
5. Peter Tompson. Small-scale Networks for General Systems Interconnect. SGS-Tompson Electronics Limited.,UK., The 11-th International Symposium ICSNET'95. St. Petersburg. 1995.
6. INMOS Limited (1993). The 9000 transputer Hardware Reference Manual. SGS-Tompson Microelectronics Limited . Bristol. UK.
7. R.W. Dobson, A. Martin, S. Haab, R. Heeley, M. Zhu, J. Renner Hansen. Realization of a 1000 Node High-speed Packet Switching Network. The 11-th International Symposium ICSNET'95. St. Petersburg. 1995.
8. Saravanan Agasaveeran, Qiang Li. Distributed Job for Scheduling in SCI Local-Area Multiprocessor. The Fourth International Workshop on SCI-based High-performance Low-cost Computing. 1995.
9. V.I. Vinogradov. Parallel-pipeline Architecture for Real-time Data-flow Acquisition in Modular Systems. IEEE Nuclear and Plasma Sciences Society. 7-th conference on computer applications in Nuclear, Particle and Plasma Physics RT'91. KFA, Germany, 1991.
- 10.D.B. Gustavson, V.I. Vinogradov. Advanced Modular System Architecture with Distributed Memory Based on SCI. The 10-th International Symposium on problems of modular systems and networks (jubilee). ICSNET'93 Proceedings of Plenary reports, Puskin, Russia, 1993.
- 11.D.B. Gustavson, V.I. Vinogradov. Status and Development Problems of Distributed Modular Systems Based on SCI. The 11-th International Symposium. ICSNET'95 Abstracts. St. Petersburg, 1995.
12. IEEE 1596 - SCI Standard.
13. IEEE P1596.4 - RAMLINK Proposal.