

B. Product Components

The product was broken down into the following components:

- data storage
- data management
- data access
- archiving
- data display.

C. Concept

Data logging is based on the concept of logging groups which cover a number of similar measurements to be logged in a similar way, such as humidity from different regions of the LEP tunnel. All measurements within a logging group have, by definition, the same parameters (logging period, retention time, start and end dates). Logging groups are identified by a unique number. This organization of data is transparent to the user selecting measurements for display, where they are presented according to the equipment systems and subsystems to which they belong.

D. Environment

The database is currently running on a SUN SPARC 2000 station. The logging and archive programs, written in C and Pro*C, run on an HP-UX workstation. The user interfaces may run on any X-terminal that can connect to the control network.

III. PRODUCT DESCRIPTION

A. Data Storage

An ORACLE database was chosen as the repository for all logged data, following the analysis done by the SL control group. The structure of the tables, as well as the storage method used, was that devised by the above group, i.e. the use of a set number of pre-filled time slots for each group of measurements read, whose corresponding rows are updated sequentially with time stamps and measurement values [1]. This strategy was adopted to facilitate future attempts at data correlation.

A set of tables is used to store data for a logging group; this consists of one table for logged data, one for the time stamp, and one for the time slot management. There are as many sets of tables as there are logging groups. A strict naming structure of the tables is used: Tx_LOG for logged data, Tx_TIME_SLOTS for time slot data, and Tx_TIME_MAN for time slot management data, where x is the logging group identifier.

B. Data Management

This enables users of the data-logging system to specify the measurements to be logged and to set the logging parameters according to their needs. A logging group can be created by choosing measurements from the reference database on a form-based user-interface. In a similar way modifications to a logging group's parameters can be made. A designated table on the logging database is then updated with the logging group number in order to inform the creation and modification handler that a change has been requested.

For measurements that are managed by services other than the TCR and which are therefore not described in the STRefDB, the previous facility cannot be applied. For these, a set of scripts has been developed in order to allow for the creation of a new logging group and for the modification of logging parameters. These scripts are run by the data-logging specialist, as no tool has been developed to handle data validation, program malfunction, or communication with the logging programs.

One module of this component then builds or alters the logging tables required for a logging group according to the chosen parameters. The TCR reference database is accessed in order to obtain the logging system parameters. The required table sizes are calculated using the number of measurements to be logged, the logging frequency, and the retention period. Control tables in the logging database are updated with the logging parameters of the system so that the data access programs can determine what to access and where to store the data. There are two control tables, one which holds the parameters relating to the logging groups, the other which contains the parameters (physical address, conversion factor) relating to the measurements in each logging group. Any errors in the table manipulations are signalled and can be recovered.

C. Data Access

This component is responsible for obtaining the measurement values from their source, and for storing them in the database. There exists only one program, the logging black box, which is used to read any type of equipment whose access method has previously been determined. One instance of this program is run for each logging group. These

programs are launched by a logging server which then monitors their behaviour (Figs. 1 and 2). This server must always be running.

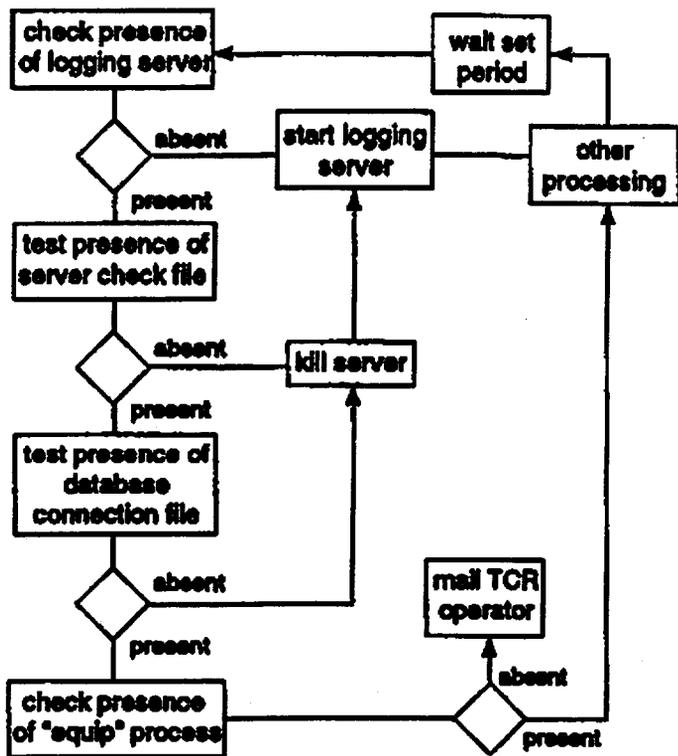


Figure 1. Checks on the logging server program

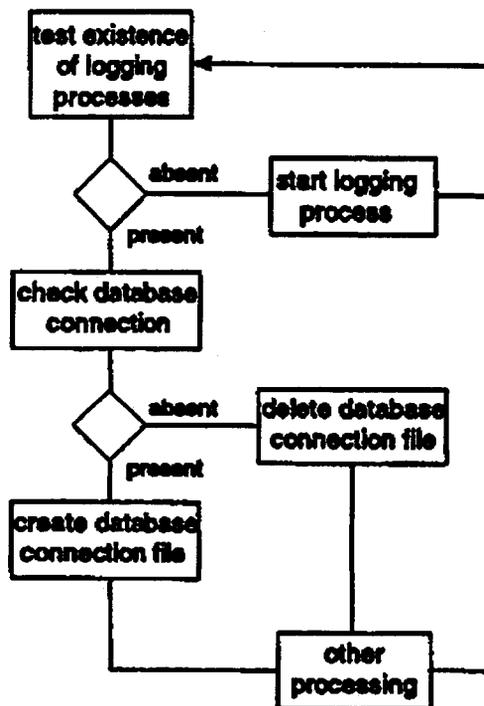


Figure 2. Server checks on logging processes

In order to determine how many logging black boxes to run, the server program reads the logging control tables. Once a black box program has started, it must first determine what kind of equipment it will be accessing, since data from different sources are read and interpreted differently. It will then determine which data elements to access and where the data will be stored; this is done once again by consulting the control tables. Should access to the database become impossible, the data is written to the file system and will be written to the database once the latter is available (Fig. 3).

D. Archiving

Since data is kept online for rapid access for typically one to two months, it is archived regularly to the file system. This is done for each logging group by the archive program, which determines the number of logging systems running and their data retention periods by reading the control tables. Archiving consists of exporting the table data to the file system and compressing the data. The name of the compressed file contains the logging group code and the archive date, thus permitting the data to be automatically restored should this be required.

CERN has a legal obligation to keep a 10 year record of the pH and temperature of the water which it discards into the local rivers. For other data, the archive retention policy is in the process of being determined.

In addition, the archive program also updates summary tables for groups that require them.

E. Data Display

Once the data is stored on the ORACLE tables, it is publicly accessible by any tool that interfaces with the database, e.g. EXCEL can be used to retrieve data and display it graphically. However, since such tools do not present a sufficiently well-tailored interface to the user for selecting the measurements for display, and also lack functionality such as zooming, a specific module was developed to satisfy these goals.

The data display module is divided into two parts, one for managing the selection of measurements, the other for handling the graphical display. The data selection part consists of a menu-driven user-interface which guides the user by presenting the available measurements using names with which he is familiar rather than the table and column names

themselves. The information on which the menus are built is obtained once again from the logging control tables. Once the required measurements have been selected, the graphical display program is called with the appropriate display parameters. The display program uses in-house graph templates built on top of the Xplore package, allowing the use of XRT Motif widgets without the need for X-Windows programming. The graph displayed can be updated with new data if so requested. Should the data required for display not be available online, the archive is searched for a file referring to the data requested. If the file is found it is restored to the database and the request is processed accordingly.

This program uses the X-Window protocol for data display and thus must be run on an X-Window emulator if used on PCs and Macintoshes.

IV. PERFORMANCE, PROBLEMS, PERSPECTIVES

The logging programs are generally reliable, though certain problems have been experienced during the evolution of the product over the past two years.

Accessing a remote database requires that the unavailability of the machine running the database, or the database server itself, must be correctly handled. The product uses a number of standard tools, either provided commercially or made in-house. Each new release of a tool may lead to incompatibilities or may imply a modification of some part of the logging application. These can sometimes be overlooked and cause problems that are not immediately detected. Access to the measurements, which are distributed over a wide area across a number of networks, may not always be possible. Errors in obtaining data are now logged themselves in a separate file for each logging group; these can then be analysed to determine whether equipment is functioning correctly. The expansion of the application to cover more than the originally planned number of measurements to be logged led to changes in the configuration of the machine running the logging programs and to an optimization of the files used for error reporting and archiving. Original design weaknesses are gradually being corrected. Our experience leads us to conclude that the use of one single facility for SL and TCR does not permit optimization for both parties; it is envisaged to create two independent set-ups.

The product has been of significant use to many different services at CERN and as a consequence is constantly evolving. For example, new requests for logging different data sources mean adaptation of the data access modules. Though diagnostics on the logging processes have steadily been improved, better treatment of data access errors is overdue. Moreover, with the advent of new control software [3] in the technical services domain, a move towards more event-driven logging will be made, and this will of course have a significant impact on the product.

References

- [1] R. Billen et al., "LEP Accelerator Logging System Using On-line Database", Nucl. Instrum. Methods Phys. Res. A352 (1994) pp. 296-299.
- [2] P. Lienard, "Evolution of the SPS and LEP Communication Control Network for the SPS and LEP Accelerators" in Proc. ICALEPCS'93, Nucl. Instrum. Methods Phys. Res. A352 (1994) pp. 170-172.
- [3] P. Ninin et al., "The Technical Data Server for the Control of 100 000 points of the Technical Infrastructure at CERN", these Proceedings

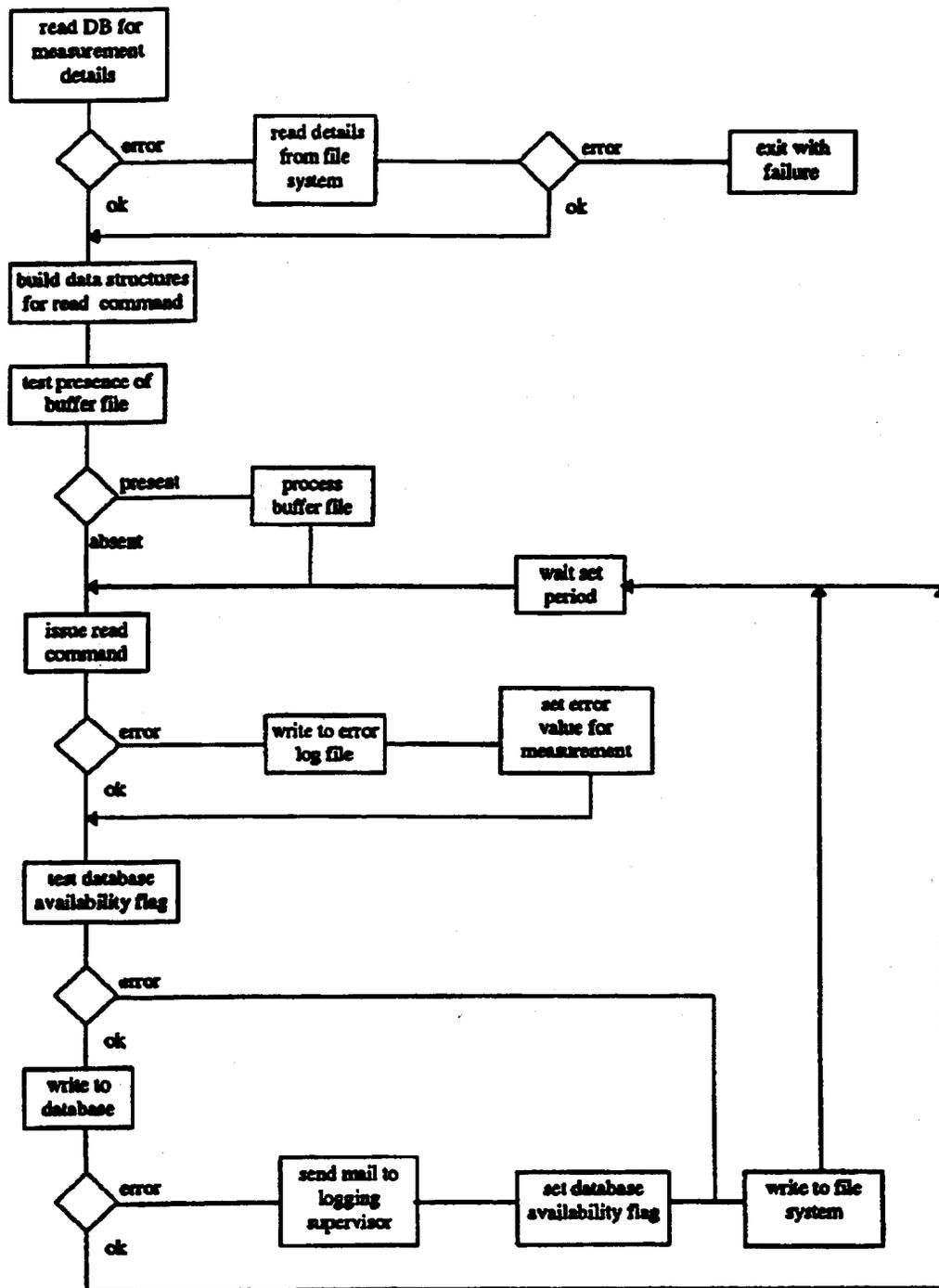


Figure 3. Data access and storage module

WWW FOR INFORMATION ON CERN ENERGY CONSUMPTION

H. Laeger and S. Lechner
CERN, Geneva, Switzerland

Abstract

The electrical energy consumption at CERN and the related costs are of concern for several reasons, not least for their impact on the budget. It is believed that the availability of information on the consumption and the cost of distinct sectors to those who can influence them, i.e. a large part of the CERN population, will stimulate individual initiatives for reductions in consumption and cost. A method has been developed whereby logged data is retrieved from a database and presented in comprehensible histograms. Data can be viewed either as variations over standard periods (day, week, month) or as integration over standard periods. Specific solutions have been devised giving a rapid response time, as well as for generic scaling with optimal resolution for data which may vary over several orders of magnitude. Access to global data is available only to the management and is password-protected, whereas access to all the distinct sectors of consumers is granted to everybody on the CERN site. The implementation is made using the hypertext technique and a WWW server. This paper describes the main graphical tool, the concept of data preparation for instant reply, implemented reliability procedures and some initial experience with its use.

I. INTRODUCTION

By the very nature of its installations CERN is a big consumer of electrical energy, mainly for the accelerators, but also for the basic technical infrastructure. The energy cost is high and represents close to 10% of the CERN material budget.

Since the 1973 oil crisis particular efforts have been made with good results to reduce the energy consumption of some of the larger installations by the introduction of energy-saving operation modes such as pulsing beam transfer lines and tightly adjusting the length of the SPS beam energy flat top to actual beam requirements.

Information on electrical energy consumption, be it for the whole of CERN, for the respective share of its two national suppliers (French EDF and Swiss EOS), or for individual accelerators, technical services or laboratory areas, has in the past been available only to a restricted number of people and only as integrated monthly figures.

A first step towards wider and deeper information for improved energy management, and in particular to ensure the correct response to contractual restrictions on consumption on some critical days in winter time was made a few years ago by putting the actual consumption of larger users at the disposal of all CERN control rooms. This information was then rapidly accessible via the LAN and could thus be consulted by all PC users connected to it.

Following suggestions from the community of physicists, the CERN Energy Management Panel expressed the desirability of extended information such as:

- access to information on energy consumption for the entire CERN population (so far, users of Macintoshes and X-terminals outside the control rooms have been excluded)
- presentation of consumption in the form of histograms and integrated values over time
- presentation of cost, again in the form of histograms and integrated values
- a structured overview of the actual consumption of all larger installations.

It is expected that the availability of useful real-time information on energy consumption and cost, sufficiently detailed that the contribution of individual systems can be spotted, will lead to adaptations and finally to reductions in consumption and in cost.

II. REQUIREMENTS AND CHOICES

A. *Software Development Strategy*

The procedure adopted for the creation and implementation of this tool was based on interactive progressive prototyping. The reason for this is that the subject is of high sensitivity and rapid reactions from a test group of people were thought vital to verify acceptance of procedure, form and content.

B. *Use of World-Wide Web*

The need for access on any platform pointed right away to the use of a WWW server since WWW browsers exist for any platform. However, as this information is only intended for the CERN population, the server has to be invisible from outside CERN.

Moreover, as it was requested that some global data concerning contracts with the two national suppliers of electrical energy to CERN should be made available only to a limited number of management staff, a mechanism of password control had to be implemented.

C. Data Retrieval

The required data for these displays can readily be retrieved from a database that is filled by an existing data-logging system [1]. For reasons of response time and reliability, the real-time data is also retrieved from the database and not directly accessed from the equipment. In the present context it is considered acceptable that data which may be up to a few minutes old should be declared 'real-time'.

D. Response Time

A fundamental requirement for any human-computer interface for rapid response time led to the adoption of pre-processed and stored graphs, ready for instant display upon a user call. The price to be paid for this essential advantage is disk space, nowadays not expensive.

III. DESCRIPTION AND ORGANIZATION

A. Reference Data for Configuration

For correct configuration of the system, reference information is needed on electrical energy consumers, such as technical identification and addresses which uniquely identify the measurements in the data-logging system, is needed. Furthermore, the energy supplier (EDF or EOS) to which each consumer is connected as well as consumer tariffs are relevant for cost plotting.

The password for full information display is treated as part of the configuration.

As all the dynamic data to be retrieved are stored in an ORACLE database, the obvious way of storing static reference information was in ORACLE tables. For this purpose four tables were created: one describing consumer information, one each for EDF and EOS tariffs and a fourth one for valid passwords. In addition, one SQL-forms interface was built for easy updating of the tariff and supplier tables.

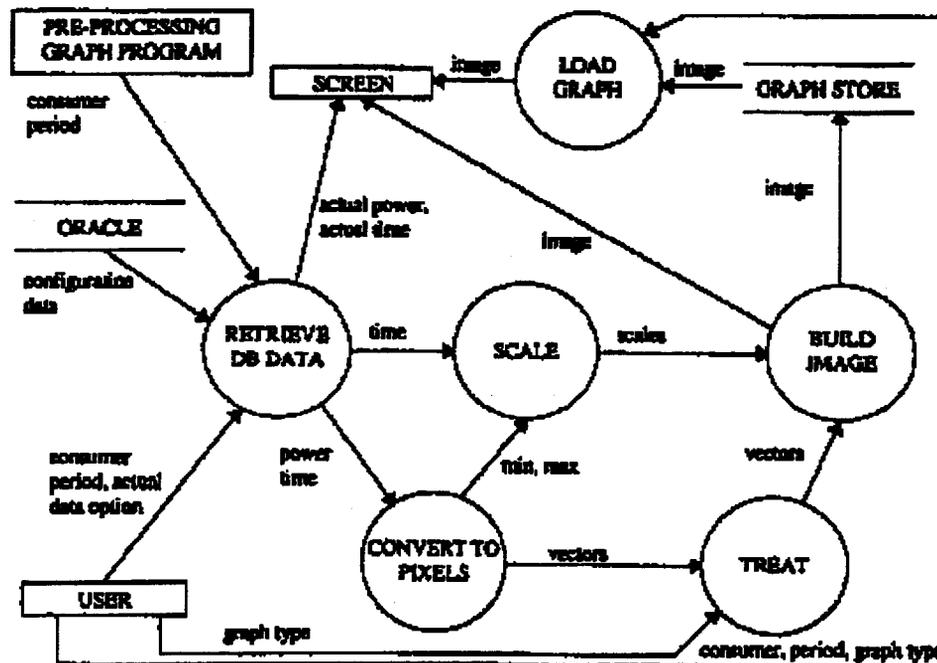


Figure 1. Data flow diagram for graph selection, production and display

B. Data Management

The graph data (time and value) are retrieved from an ORACLE database (Fig. 1) already filled by an existing data-logging system. A set of routines written in Pro-C (SQL-like syntax) permits the retrieval of data for a particular consumer whose description was read before in the configuration tables.

For cases where data acquisition problems occurred at data-logging, with consequent gaps in the database, the data retrieval routines will fill these gaps with the last-available set of data.

Owing to the limited size of the graphic display and the large number of data, some data reduction needs to be performed. The treatment consists in replacing all individual data with the same x-axis values (pixel granularity for time) with their average. The resulting set of points constitutes the final vectors to be plotted.

C. Graphics

The selected graphical library used to plot data is Thomas Boutell's library 'gd' [2]. This library provides low-level functions such as pixel drawing, line drawing, string writing on a graph, as well as graph saving in gif format. It had to be enhanced by a few specific routines (vector plots, legend display, automatic scaling).

D. Scaling

One of the main problems to tackle when displaying data which may vary over several orders of magnitude is that of scaling, all the more since in our case an automatic scaling scheme was desirable.

For horizontal automatic scaling a limited number of natural periods have been selected, i.e. one day, one week, one month. Major subdivisions are then chosen: two-hour periods for a day, six-hour periods for a week, and one-day periods for a month. The C routine built for this scaling task is straightforward.

In the vertical axis, for ease of readability, only a limited number of pre-defined scales are used. These scales are based on multiples of 10^n of the rounded linear $\sqrt{2}$ progression (1, 1.4, 2, 3, 4, 6, 8). The extremities of the scales can only have these values; furthermore if the minimum value is below 25% of the scale maximum, the scale minimum will be set to zero. For scale subdivisions, the same scheme has been adopted with multiples of 10^n of 0.5, 1, 2, and 4 for major units, with a minimum number of 4 and a maximum of 10 major units presented.

The implementation of the above algorithm is simple. Once minimum and maximum values of the vectors to be plotted have been found, the scales are selected from the set of pre-defined authorized values. The same routine is applied to the consumption and to the cost scales.

E. Graph Layout

The general layout chosen for all graphs has been fixed to a 750×400 pixel display, which fits most computer screens and WWW browsers. Pre-defined areas are dedicated to scales and to legends. Graph borders and time-scales are drawn in black, energy consumption scales and plots in red and cost scales and plots in blue. In order to make the plots also readable on black and white screens as well as on printouts, consumption and cost are distinguished by different line styles. The graph title is set with consumer name and graph dates.

The whole graph appears with a transparent background on a WWW browser.

F. Plotting

The resulting vector after data retrieval and treatment is directly plotted as a histogram using a step-function drawing routine. For integrated values, the same drawing function is used but the vector is integrated beforehand by a C routine. Scaling and plotting are treated separately.

G. Pre-Processed Graphics

Initial prototyping showed that the response time was unacceptably long when the user has to wait for the time it takes to access the database, to retrieve the data and to build the graph. Typical response time for a 'day' graph with about 10 000 data couplets was 5 seconds, 15 seconds for a 'week' with about 70 000 couplets and 45 seconds for a 'month' with 250 000 couplets. We therefore opted for a procedure with pre-processed graphs, which reduces the response time to that required for the connection to the server and for the transfer, decoding and display of the graph. Normally, the user gets the requested display within a few seconds. The 'real-time' data, and the graph of the actual day are not prepared in advance and there the response time is higher (an average of 30 seconds and 10 seconds, respectively).

The global strategy is to store graphs of the last five weeks, of the last 31 days excluding the current day and of all past months including the last 31-day period.

The program written to create pre-processed graphs begins by reading configuration tables and tries to process all 60 graphs in sequence (last day, current week and last 31-day period for every defined consumer). However, in case of problems (ORACLE unavailability essentially) no graph is stored at all. Graphs are processed at a given time at night; afterwards verifications are done periodically to look for missing graphs in which case processing is reinitiated. Assuming ORACLE data retrieval problems remain for a whole day, only the graphs for the last day will be missing and current week and month graphs will show incomplete data.

Since the defined policy is to keep the last 31 days and the last 5 weeks, older 'day' and 'week' graphs are deleted in order to minimize the disk space used.

H. User Interface Entry Page

The entry page (Fig. 2) into the system called 'Energy Consumption and Cost Estimation for CERN Accelerators and Experiments' displays all selection points as hypertext links that lead to the corresponding graph page for the display of histograms.

It was initially intended to display the real-time data as well as the hypertext selections for the individual users. For reasons of response time, we opted for a more rapid display of the selection points which now contain an additional button for access to the actual data.

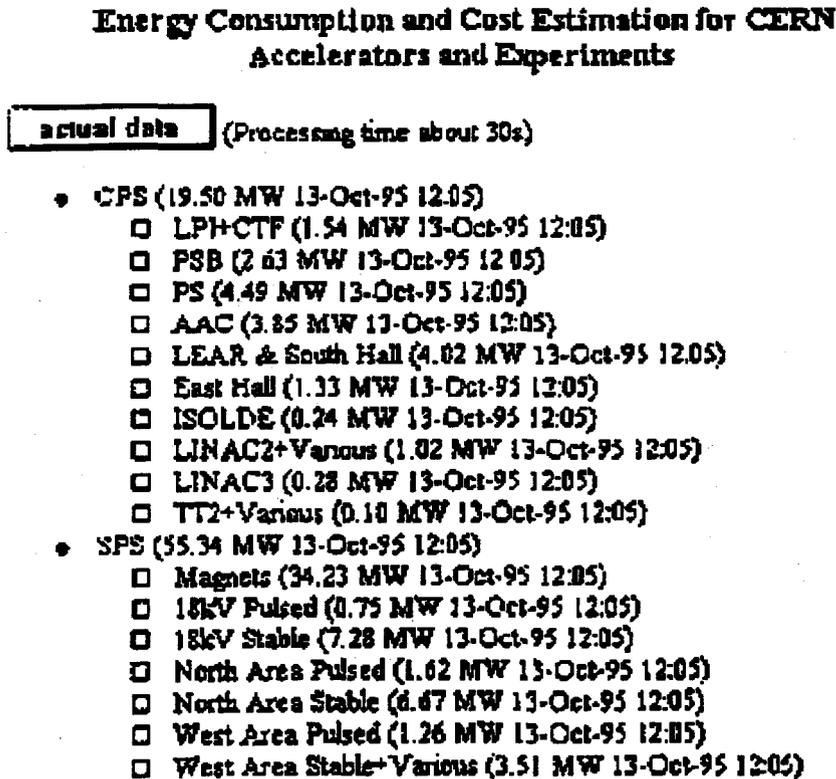


Figure. 2. Part of the entry page with real-time data

Moreover, as access to information on the global energy consumption of CERN was not to be displayed to everybody, a mechanism of password access for a limited number of management staff had to be implemented in the form of an additional button on the entry page.

Finally, a hypertext link called 'Comments' at the bottom of the page allows users to address mail easily and directly to the project administrators.

I. Graph Page

The graph page (Fig. 3) used for different selection points and graph types (histogram or integration) is the same except for the page title.

Users can choose between one of the last 31 days, one of the last 5 weeks, or any month including the last 31-day period. Once an option is selected, the requested graph is displayed on the same page just below the menu options. Every graph page presents a direct link to the entry page and to the other graph type (histogram or integration) for the same point. If another option for the same point is requested, the resulting graph replaces the previous one.

Whatever options are chosen it is always possible to get information on energy cost and cost estimation since the graph page title contains an hypertext link called 'Cost Estimation'.

From any position, a direct link to the CERN home page is present for new navigations starting at this point.

J. Implementation of the User Interface

Entry and graph pages treat data input from the user (button pushed, value selected) and have therefore been implemented as Common Gateway Interface (CGI) [3] C scripts. Much effort was put into obtaining an application compatible with most browsers especially NSCA Mosaic and Netscape and their different versions.

CERN Energy Consumption and Cost Estimation : CPS

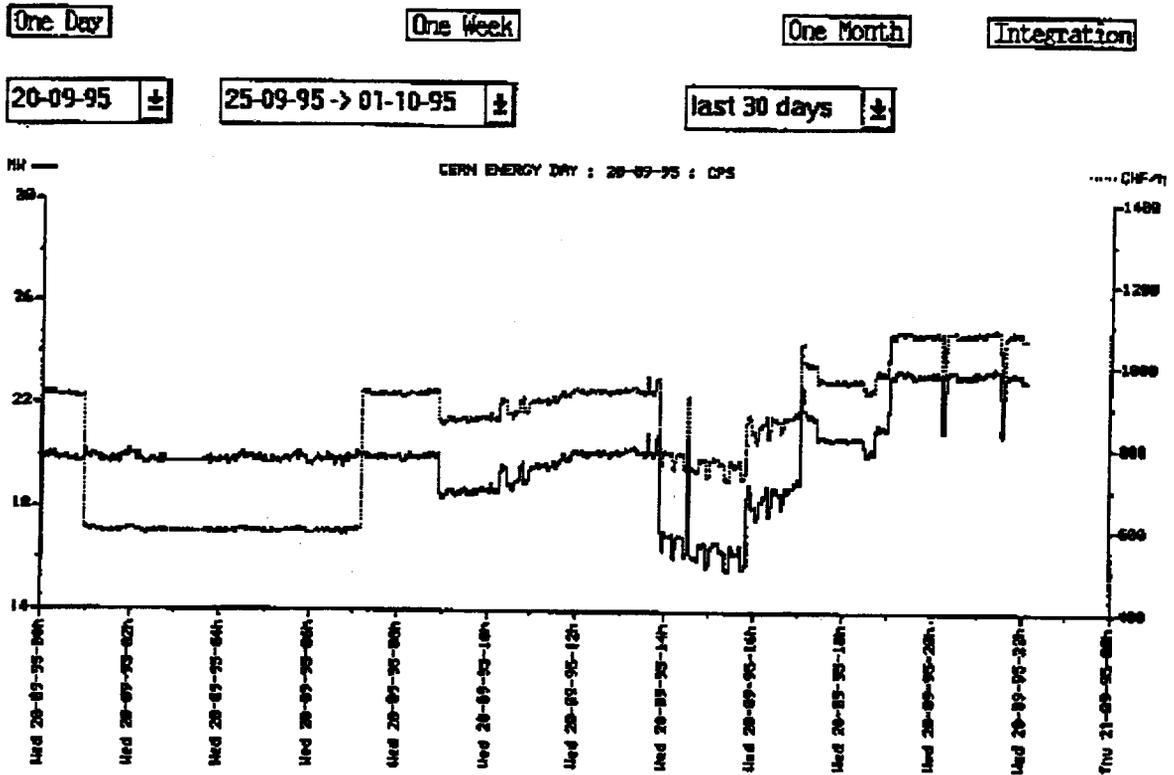


Figure 3. Example of graph

K. Entry Page

The default call to this page displays all information mentioned above using the C basic print function. When real-time data are requested a call to ORACLE is made for the last logged values for the selection points and the result is displayed on the screen. For access restriction, the entered password is compared with the valid one and, if correct, full information is displayed (Fig. 4).

L. Graph Page

The graph page is called by the entry page which sends information such as the requested point and the graph option to the WWW server. Once the user requests a particular graph, additional information such as day, week, or month and their corresponding dates is available and the graph is plotted. If the requested graph exists as a stored gif file it is just redisplayed, if not, it is built in real time. When several graphs are requested one after the other, the selection point and the type of graph remain as hidden parameters and the other options are changed by the user.

To avoid use of multiple submit buttons on the same page which may not be recognized by all browsers, each button has been replaced by a small gif image representing the button. These images, linked to the HTML command 'input type=image src=...', act as buttons.

IV. EXPERIENCE AND OUTLOOK

The tool is currently in its final testing phase. Availability is very high, limited essentially by the fact that the WWW server is installed on a HP-UX machine which runs a multitude of development programs. This can be improved by installing the application on a more appropriate machine.

Response time is now close to optimum; it remains to be seen if the delay of 'real-time' and 'actual-day' data is accepted by the users.

User comments are so far only available from the limited number of test persons; they are essentially positive and welcome the availability of the information in the proposed form. The application has been built for easy adaptation to new user requirements and it is expected that once it is used by the entire CERN population we will need this facility.

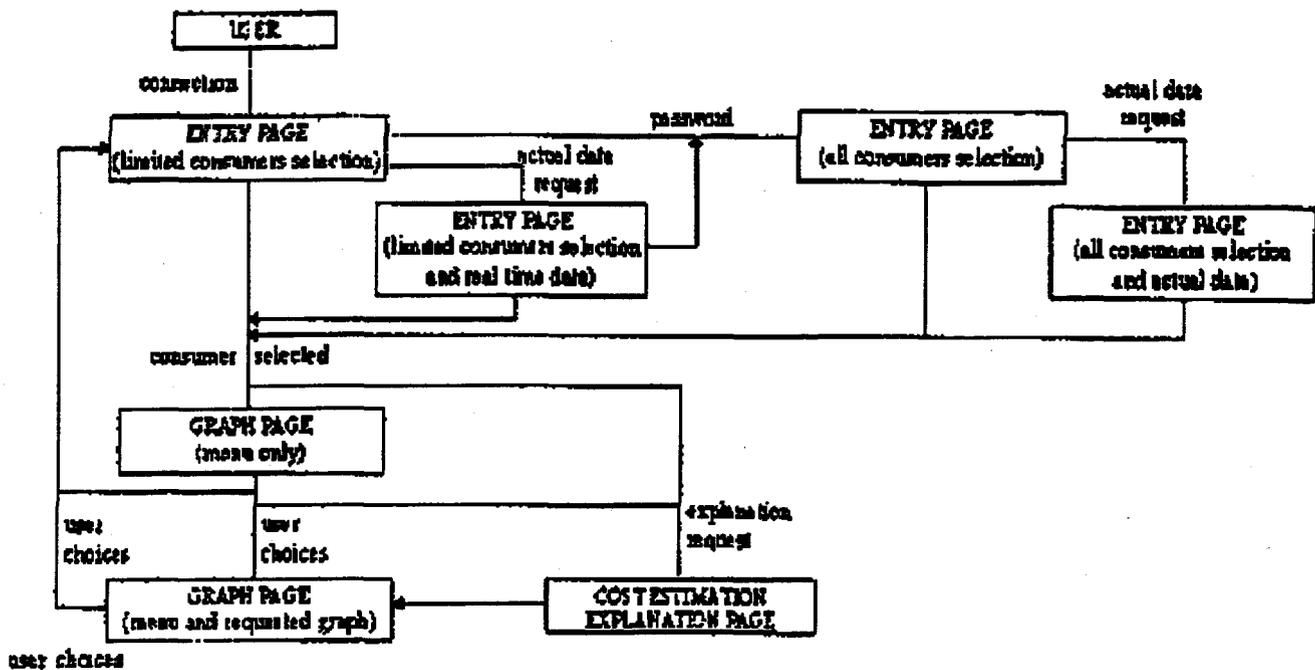


Figure. 4. Navigation overview

The data-logging system which has been in operation for a two years sets different intervals for data-taking according to individual preferences. For energy measurements some are set to one minute, others to five minutes, most to ten minutes. Because of the 'real-time' use of this data by the present application, all intervals for energy measurements have now been set to two minutes. The subsequent increase in the number of equipment calls and in storage capacity required for the database is well within system capacities and so far has not presented particular problems. The disk space required for the pre-processed graphs for all 'day', 'week', and 'month' graphs for a year amounts to about 20 Mbytes.

The intensive testing during the prototyping phase of the energy displays revealed a relatively high number of gaps of different duration where data was not logged by the data-logging system. This stimulated efforts to increase the reliability of the data-logging system. A Technical Data Server [4] to be used for the supervision and control of the technical infrastructure of CERN has been defined to also serve this purpose.

V. CONCLUSION

The tool described in this paper provides an easy and reliable way to distribute information on CERN energy consumption and cost all over the site, whatever the computing platform used. The final testing stage has shown that it is reliable and that changes can be made easily. All of its constituent modules proved to be well interfaced and, to a large extent, they can be changed independently.

Clearly, the mechanism for the display of information on CERN energy consumption and cost is only a tool. However, it may stimulate individual actions and together with other management incentives may achieve the basic aim of the reduction in energy consumption and cost.

References

- [1] R. Martini, H. Laeger, P. Ninin, E. Pfirsch and P. Sollander, "Data Logging for Technical Services at CERN", these Proceedings
- [2] gd is copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs.
- [3] The Common Gateway Interface, <http://hoohoo.ncsa.uiuc.edu>
- [4] P. Ninin, H. Laeger, S. Lechner, R. Martini, D. Sarjantson, P. Sollander and A. Swift, "The Technical Data Server for the Control of 100 000 Points of the Technical Infrastructure at CERN", these Proceedings

Control of VEPP-4M Magnetic System

S.Karnaev, E.Kuper, A.Ledenev, B.Levichev, I.Protopopov, Io.Zaroudnev

Budker Institute of Nuclear Physics

Prospekt Lavrentjeva, 11, Novosibirsk, 630090, Russia

Abstract

The use of BINP-designed intelligent controllers for high-stability magnet power supplies allows us to solve successfully the problem of beam acceleration in the VEPP-4M collider. There are two versions of the controller. The first has sixteen-output sixteen-bit DAC; the second has a nineteen-bit DAC, an error amplifier and a shunt. Both versions have digital linear interpolation and MIL/STD-1553 interfaces. The aspects of the magnetic system control concerning hardware and software are described. Results of a field behavior study in magnetic elements with the use of these distributed intelligent controllers are presented.

1 INTRODUCTION

The VEPP-4M collider provides experiments with electron-positron beams, back-scattered compton γ -quanta and synchrotron radiation. The circumference of the accelerator is 366 m. The beams in the ring are guided by more than 100 dipole magnets, quadrupole lenses and steering magnets.

The collider is an accelerator with an energy range of from 1.8 GeV at injection to 6.0 GeV. Therefore, a simultaneous rise of the magnetic fields by a factor of 2.5 - 3 is necessary for performance of high-energy physics experiments within a range from 4.7 GeV to 5.4 GeV. This beam energy rise is controlled by the Intelligent Device Controllers (IDCs) based on microprocessors.

Control of the IDCs is performed by the CAMAC-embedded ODRENOK computer [1], which is integrated into the VEPP-4 control system [2]. ODRENOK loads predetermined settings into IDCs and controls them by commands via a MIL-1553-B multidrop data bus.

2 MAGNET STRUCTURE CHARACTERISTICS

The magnet system of VEPP-4M consists of arc cells (FODO), dipoles, quadrupoles, sextupoles and octupoles. Auxiliary coils in dipoles, arc cells and steering magnets are used for beam orbit correction. Moreover, quadrupoles have built-in coils for gradient correction. The VEPP-4M magnetic structure is symmetrical relative to the collision point. Table 1 gives an overview of the magnetic system.

Most of the elements are controlled by their own power supplies. All the magnets and power supplies have been developed at BINP. The total number of power supplies is about 330.

All arc cells and some of the dipoles are powered by the same power supply. Quadrupole coils (F and D) in the arc cells are used for control of the tunes. The coils of the same type in all arc cells are energized by the same power supply. The ring chromaticity is mainly regulated by the sextupole coils in the arc cells. These coils are powered by four supplies.

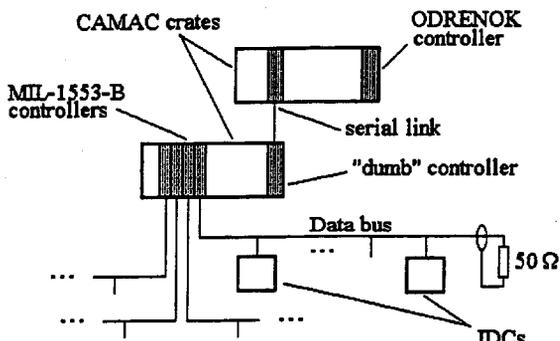


Fig.1 The configuration of the control system.

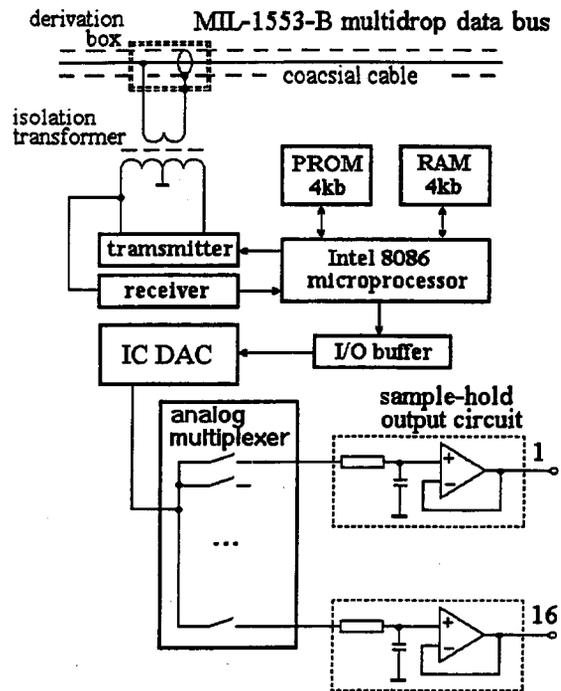


Fig.2 Block diagram of 16-output DAC.

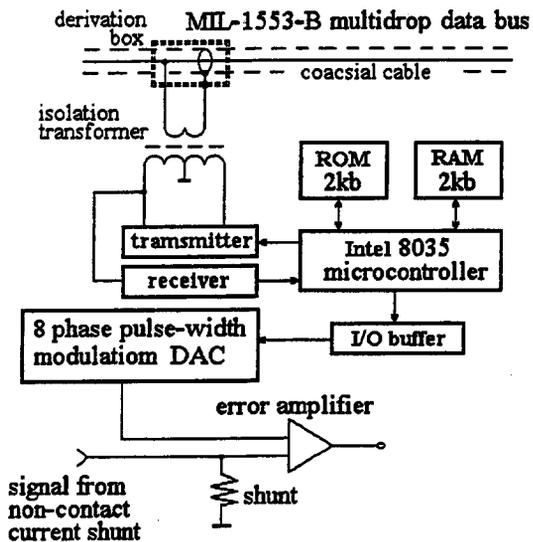


Fig.3 Block diagram of single-output DAC.

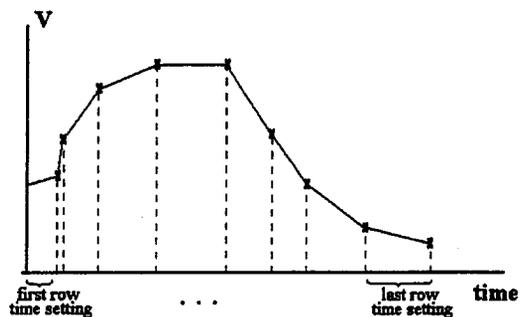


Fig.4 Example of interpolated output signal.

Table 1- The VEPP-4 magnetic system

Magnet type	Number of magnets / pow. suppl.	I_{\max} , (A)	H_{\max} , (kGs)
arc cell (DO, FO)	66 / 3	6400, 550, 170	5.3
dipole	4 / 2	1600	6.6
	2 / 1	1100	14.5
	8 / arc cells	6400	9.5
quadrupole	2 / 1	1500	6.7
	2 / 1	1700	7.5
	14 / 7	200 - 400	2 - 4
	8 / 2	1350	8.1
sextupole	arc cells / 4	500 - 700	
wiggler	2 / 2	2700	
snake	2 / 1	2000	
sextupole	6 / 6	25	
octupole	2 / 2	8	
steering magnet	21 / 21	8	
skew-quadrupole	6 / 6	25	

3 CONTROL SYSTEM ARCHITECTURE

There are four data buses: two are connected to 16-output IDCs, the others are connected to single-output IDCs. Such a configuration permits the use of common commands for IDCs of each type. For example, the "write in" command has a different format for IDCs of different kinds. Thirty remote modules may be connected to each bus.

CAMAC embedded controllers developed at BINP are used to control the data buses. The controllers are located in a peripheral crate connected to ODRENOK through a 1.6 Mbit/s serial link driver and controller [3].

Derivation boxes without protective resistors are used to couple the stubs to the data bus (Fig.2). The length of the stubs is less than 1 m.

The length of the buses is about 300 m. They pass over all the power supply control rooms around the collider. The transmission rate is 1 Mbit/s. Each bus has a 50 ohm matching resistor at the end.

4 HARDWARE

To control the collider magnetic system, two types of IDCs are used. The first is based on the Intel 8086 microprocessor. It has 16 analog output signals produced by an integrated circuit DAC. These modules are used in order to control low-current power supplies. Fig.2 shows a functional block diagram.

The second type is based on the Intel 8035 microcontroller. It has one control output, which provides an amplified error signal. This type is used for the control of middle- and high-current power supplies. A functional block diagram is shown in Fig.3.

Both versions have static internal RAM. A sequence of up to 80 so-called "rows" can be loaded into the RAM. Each "row" consists of 16 settings (or one setting for single-output IDC) and one time setting. There exists the possibility of decreasing the time scale by a factor of 16. That enables control of the "fast" magnetic processes. An example output signal is shown in Fig.4.

In addition, the microprocessor calculates a duration in "tick-portions" for each "row" as a "calculate interpolation" command comes. Sample-hold output circuits are controlled with a frequency of 256 Hz. "Start", "stop" and "continue" commands are simultaneously sent to all IDCs connected to the same data bus for the synchronization of magnet control.

Table 2

Main features of IDCs

Version of IDC		16-output	single-output
Scale (binary)		14 + sign	19
Output voltage (V)		± 6.5	8.192
Error		.01%	.001%
LSB		400 mV	15 mV
Interpolation time between two settings:			
"fast" scale		64 ms - 4 s	
full scale		1 s - 63 s	
Power	+5 V	2 A	2.5 A
consumption	+24 V	0.2 A	0.2 A
	-24 V	0.05 A	0.05 A
	V		

5 MAGNET CONTROL

To operate the magnetic system (handling, cycling, beam energy rise) different programs use the Resident Executive Program (REP) [2]. REP loads tables or single settings into the IDCs and sends control commands to IDCs via MIL-1553-B.

All of the magnets are made of non-laminated low-carbon iron. Magnetic field to coil current ratio changes during the energy rise for two reasons. The first is a saturation of magnet yokes. This saturation may be corrected by a corresponding correction of the current. The second reason is associated with a delay of magnetic flux penetration into the iron during the process of the energy rise. The deviation of the ratio is determined from the equation (1).

$$d = \left(1 - \frac{H}{H_{st}}\right) \times 100 \quad , \quad (1)$$

where

H is the dynamic magnetic field,

H_{st} is the static magnetic field at the same current.

A deviation larger than 0.2 may cause the “death” of the beam. This factor limits the speed of the beam energy rise to 10 - 15 MeV/s, so that it takes 5 - 6 minutes to increase the beam energy from 1.8 GeV to 5.3 GeV.

The use of IDCs allows us to reduce the required time by a factor of 3 - 5. Supplementary intermediate settings are loaded into the IDC's memory to correct individual peculiarities of magnets. These intermediate settings may be defined for each magnet from prior measurements. The magnetic field of each magnet is measured with the use of the same Hall-probe. An example of such a correction for one magnet is shown in Fig.5.

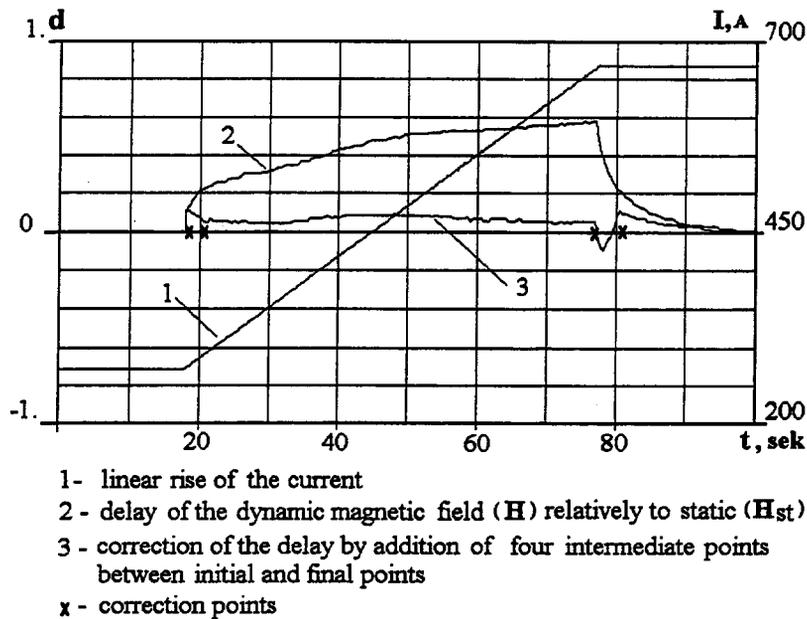


Fig.5 The magnetic field behavior with coil current rise in EM3 dipole.

6 REFERENCES

- [1] G.Piskunov, *Autometria*, N4 (1986), p. 32-38 (in Russian).
- [2] A.Aleshaev et al., *The VEPP-4 Control System*, these Proceedings.
- [3] V.Kozak, *Preprint INP 88-24* (in Russian).

The PIAFE Project - Command and Controls of PIAFE Phase 1.

The PIAFE Collaboration, presented by Solveig ALBRAND,

Institut des Sciences Nucléaires, IN2P3-CNRS/Université Joseph Fourier, 53 avenue des Martyrs, F-38026 GRENOBLE Cedex, France.

THE PIAFE PROJECT.

What is PIAFE ?

Atomic nuclei with a large excess or deficit of neutrons with respect to stable nuclei are known as "exotic". Such nuclei are highly unstable and thus radioactive with short lifetimes. The study of exotic nuclei will be one of the main topics of research in nuclear physics in the coming years. Indeed, several projects are already being studied or are already being built around the world in order to produce accelerated beams of exotic ions.

The opportunity that SARA, the heavy ion accelerator of the Institut des Sciences Nucléaires (ISN) is adjacent to the high flux reactor of the Institut Laue Langevin (ILL) has been taken in the proposal PIAFE [1] ("Production, Ionisation et Accélération de Faisceaux Exotiques") for the production of beams of neutron-rich ions with masses between 80 and 150, with a particularly high intensity. An ion source consisting of a target of a few grams of uranium 235 would be placed close to the reactor core of the ILL. The resulting fission products would be ionised and accelerated to 30 keV. After mass separation the ion beam would be transported in a few milliseconds to SARA via a 400 meter tunnel, for further acceleration. Energies between 2 and 14 MeV/amu could be attained.

Further information is available at <http://isnwww.in2p3.fr/piafe/piafe.html>.

The Scientific Interest of PIAFE.

Beams of exotic ions permit the observation of nuclear reactions which cannot be otherwise obtained. These reactions offer many new and exciting possibilities to nuclear physicists, in particular the validation of the current models of the atomic nuclei very far from the stability line. Other areas of physics are also concerned, such as the study of nucleo-synthesis in astrophysics and several aspects of solid-state and surface physics. Radioactive beams can also be used in the production of radio-isotopes for medical physics. Studies of the transmutation of radioactive waste are also proposed.[2]

The Main Points of Phase 1 of the Proposal.

The first phase of the project will verify the operation of the source. The beam produced will contain singly-charged ions at 30 keV, so low energy physics of exotic ions will be possible, such as measurement of ground state masses, moments and the spectroscopy of radioactive nuclei.

Figure 1. shows the overall layout of the installation.

The uranium ion source, placed in a "glove finger" beam tube of the reactor is designed to produce a current as intense as possible, conform to security requirements. The target, 4 g. of ^{235}U in the form of uranium carbide, is dispersed in a matrix of porous graphite. It is submitted to a flux of on average $3 \cdot 10^{13}$ neutrons per square centimeter per second to obtain 10^{14} fissions per second. The fission products are slowed by the graphite and the power dissipated will heat the source up to about 2400 °C. This temperature represents the best compromise between the maximisation of the diffusion of fission fragments out of the source and the elimination of heat from the source by radiation. A higher temperature would tend to evaporate the graphite and thus reduce the source lifetime. The source is placed in a container which will probably be made of rhenium. The container must be able to withstand heat, chemical attacks from the constituents of the source and of course resist the aggression of the neutrons. A second container in the form of a metallic grill completes the heat shielding and mechanical protection of the reactor beam tube. The source also contains electrodes which will ionise the fission products and accelerate them to 30 keV. Uranium ion sources of this type have functioned since 1967 in Sweden [3] although at a much lower neutron flux.

The installation includes a mechanism for the introduction and removal of the source from the reactor and also must provide for the loading of a new source and the disposal of a used radioactive source into a disposal bin.

After extraction from the reactor the ions must be transported to the experimental beam lines. The beam line contains 2 dipoles (bending magnets) 2 quadrupoles and 2 electrostatic lenses for focusing and 2 slits. The beam line also contains 5 beam diagnostics of various types. The ensemble consisting of the first lens LE1, the first dipole (D1) and the slit F1, acts as a pre-separator. Only ions with a mass of $\pm 4\%$ of the reference mass will pass the slit F1. Many unwanted isotopes will be stopped here, and this part of the beam line will be the most active, so it is housed in a separate bunker. D1 must also have a special structure (large gap) to permit the passage of the source and its trolley. For the same reason, LE1 will be mounted on a jack to permit vertical movement. The rest of the beam line is more classical: the beam is transported to the spectrometer from where it is dispatched to the physics areas.

The presence of highly radioactive gases close to the source also determines the techniques which must be used for the pumping. In addition to the usual electromagnetic valves closed during primary pumping, the beam line will be divided into zones in order to attenuate the gas flow during operation. In the beam direction the contamination of the transport line by radioactivity must be limited and in the other direction the source must be protected from an accidental pressure increase, for example, from an experimental area. To achieve this, separation diaphragms will be placed after each slit. The opening of each diaphragm will be slightly larger than the slit to avoid irradiation, and their length will be at least 150 mm. Each zone will possess its own dedicated pumps. No conventional pumping is however possible in the source, but the average vacuum must be better than $2 \cdot 10^{-5}$ mbar in this zone.

Current Status of the Project.

Preliminary studies for PIAFE began in Grenoble in 1992. The project quickly aroused international interest and the PIAFE collaboration is at present composed of 11 different laboratories (5 French, 2 German, 1 Belgian, 1 Swedish, 1 Danish and 1 Russian). The scientific interest and the originality of PIAFE have been recognised by an independent committee of international experts (D, F, GB, NL, USA). They have recommended support and rapid realisation for the project.

At present an 18m. beam line exists at the ISN as a model for the transfer of singly charged ions at 30 keV from the ILL: the first results are very encouraging. Good progress has also been made in the development of the ECR source which will capture these ions and render them multi-charged.[4] Both of these studies concern PIAFE phase 1.

Although the Scientific Committee of the ILL is favourable to PIAFE, the Board of Directors has yet to give its agreement to the project. A report has been submitted to this committee and the formal decision should be taken during the month of November 1995. If this decision is positive, financial support for PIAFE phase 1 should become available from the various supporting government agencies. PIAFE phase 2 will of course be considered only if phase 1 is successful in attaining radioactive beams with the required intensities.

CONTROL AND COMMAND OF PIAFE PHASE 1.

Introduction.

The control and command part of PIAFE phase 1 has just entered into the requirements gathering stage. Only a general description can be given at this time. The proximity to a nuclear reactor, coupled to the fact that highly unstable particles will be extracted from the PIAFE source itself, impose particular constraints on the control system hardware architecture as there will be zones where it is not possible to place active material. It seems fairly certain that the overall architecture will be based on a "standard model" LAN of PCs because both of the main participants in the collaboration (the ISN and the ILL) have experience with this hardware and also because budget constraints imply that there must be low-cost solution. In fact it is hoped that some ageing PCs in each laboratory may be recycled as data-servers; high performance machines are really only needed for the user interfaces. It is also certain that industrial PLCs will control the vacuum. Here also it is possible that the actual system used at the ILL could be extended to include PIAFE.

Four functional subsystems can be distinguished and are described in the following sections.

Beam Transport and Diagnostics. Beam Tuning.

Apart from the constraints of radioactive protection mentioned above, the beam handling and diagnostics control should be completely standard.

- Control of the power supplies of the magnets and other elements (a precision better than $\pm 10^{-4}$ is required)
- Supervision of certain elements with an automatic action in the case of detection of abnormal incidents.
- Control and command of diagnostics for the beam tuning - movements and current acquisition.

In addition the system must exchange information with the reactor control system and be designed to allow easy extension to PLAFE phase 2 in the future.

There will be a central control position, but some decentralised control may be required.

Command and Control of the Uranium Fission Source.

When the uranium source is in place permanent surveillance will be imperative. A malfunction would be potentially dangerous for the reactor. On the other hand the action which would be taken after the detection of such a malfunction is of such consequence that the system must not raise false alarms. Several methods are proposed and they may, and indeed must, be used simultaneously to permit a majority vote.

- **Optical methods :** Monitoring of the temperature of the central part of the source will monitor its correct operation and position. A metallic mirror will be lowered into the beam line on the vertical axis of the source, between the valve V1 and the magnet D1 (see figure 1). Note that the beam never passes in this part of the tube which serves uniquely to manoeuvre the source. Light from this mirror will be analysed by a 2 wave-length pyrometer.
- **Beam Analysis :** The presence of a beam is a very good test of the correct functioning of the source (the source well placed, electrical isolation in place etc.). Once the beam is correctly established and passes the pre-separator (ensemble D1-F1), a destructive measurement of the beam taking a few milliseconds every few seconds could be used, just downstream from the slit F2.
- **Electrical Methods:** When the reactor is stopped neither of the two previously mentioned methods can be used as the source will be cold and there will of course be no beam. When no accelerating voltage is applied to the source the conductors could be used to verify its correct insulation. When this voltage is applied, control of the currents will permit the detection of possible short-circuits or defects in the electrodes.

The Manipulation of the Source.

The source will be manoeuvred on a trolley by a rack-and-pinion steering drive. Collinear with the source, at the exterior of the reactor, is an area known as the "trolley station". It is housed in a separate bunker and serves as its name implies, as a parking place for the trolley during the experiment and also for its introduction. Just in front of the trolley station, the beam line is traversed by a vertical chamber which will be used for the introduction of a new source and the removal of a used source. This chamber will house an articulated robot arm equipped with a pincer. During the manipulation of the source the electrostatic lens must be displaced to allow the trolley to pass. D1, as mentioned above, will have a large enough gap to permit the passage of the trolley.

Figure 2. shows the sequence which will be followed to place the source in the reactor beam hole.

- A) The source will be mounted in its container and placed manually into the introduction chamber (above the beam line) on a rolling support.
- B) The support will move the source under the robot arm, which will then take it up.

C) After withdrawal of the support into the lock chamber, the door of the introduction chamber will be closed, and the lock chamber will be pumped out. When the vacuum allows, the valve V5 will be opened, and the robot will lower the source onto the trolley waiting below.

D) The trolley will then be moved and the source placed in its operating position.

The source will stay in position for 3 reactor cycles (90 days). After use it must be removed from the reactor and allowed to cool for several months.

Figure 3 describes the procedure for source removal.

A) The trolley will take the source out of the reactor and move it to the position above its "dustbin". The robot will grasp the source and raise it to allow the trolley to regain the station.

B) After removal of the trolley the robot will turn the source through 90°.

C) The robot will lower the hot source into the disposal container which can then itself be isolated and removed to a protected zone.

The trolley manipulation is an adaptation of the system currently used for the Lohengrin experiment at the ILL.

The Vacuum Control System.

The vacuum system control will resemble that of SARA. Elements will be controlled using an industrial PLC with the user interface on a PC. The PLC will supervise the system and will take action if an incident is produced, although certain security related actions will of necessity be hard-wired. The philosophy will be to keep the commands as flexible as possible whilst preventing forbidden manoeuvres by the pre-programmed logic.

The opening of the security valve (see figure 1) is particularly delicate as on the one hand permission will be required from the reactor control system and on the other hand no pumping is possible in the beam tube itself.

Another particularity of the vacuum system is the series of reservoirs required to stock the pump exhaust gases, which will be radioactive. Three reservoirs are planned:

- One will be in use.
- One will contain gas in course of deactivation; three months storage should be sufficient.
- The third will be to stock the gases from the primary pumping cycle, i.e. before the beam is present. These exhausts will contain only radioactive dust which will be filtered. After monitoring, this gas can be released into the atmosphere. This third tank will also act as a back-up.

Much of the vacuum system equipment will become radioactive and must be housed in controlled zones. This means that the maintenance of the pumps (oil-changes) must be telecommanded. If a pump breaks down it will simply be replaced.

People, Money and Time.

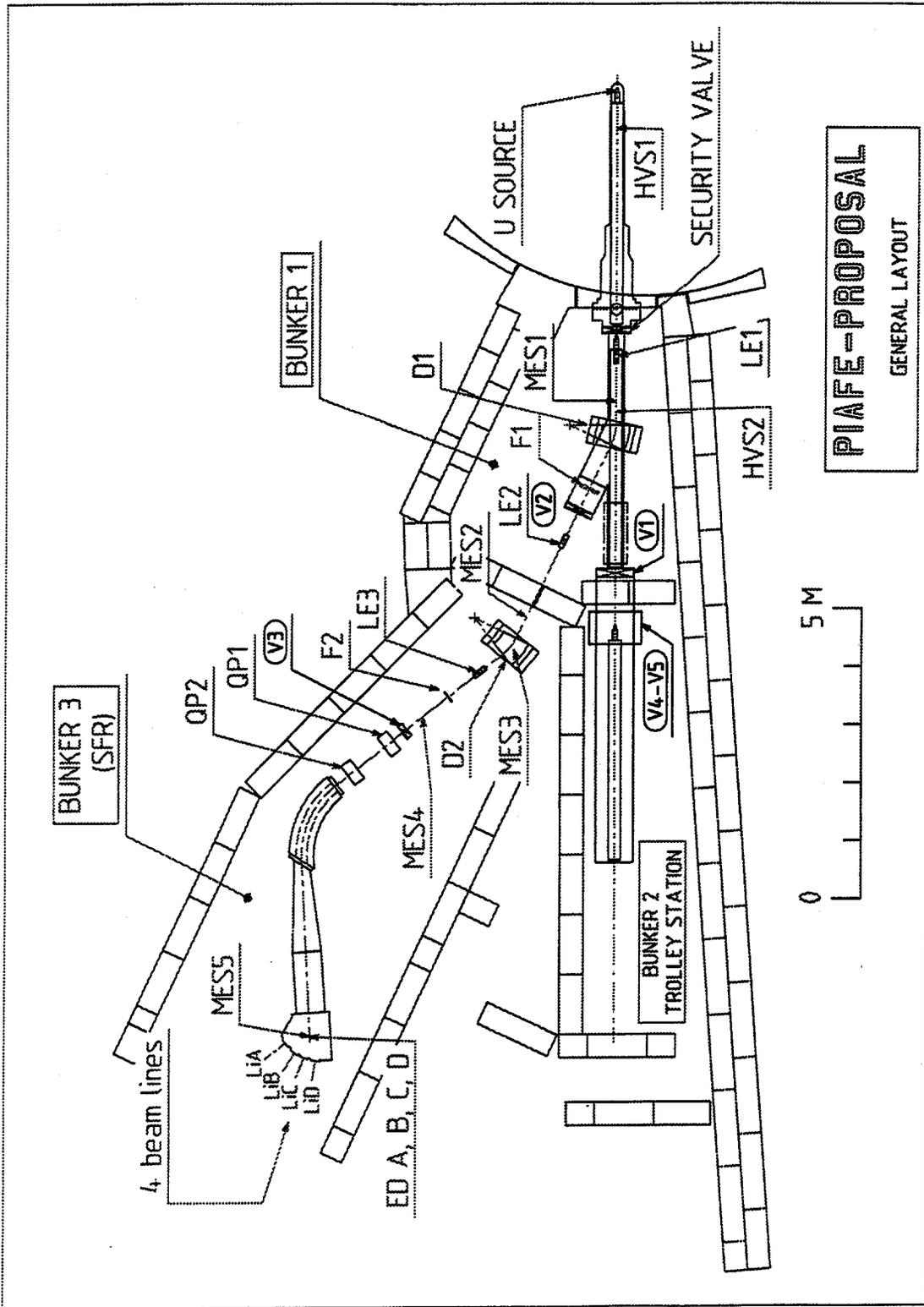
It seems unrealistic in a research environment to define the manpower needed for this project. Rather one should consider the manpower available, with its acquired expertise and include these factors as criteria in the system design. It is to be hoped that various groups from the different members of the collaboration will undertake the parts of the work for which they are most suitable. In particular the manipulation of the source should be done by a team with experience in this domain.

The latest estimate for the total cost for PIAFE phase 1 is 12 MFF (about 2.5 M\$) not including salaries. The controls and commands budget is expected to be 1.75 MFF, including all the robotics and the beam diagnostics.

It is hoped that the first beams could be delivered for physics in the second half of 1998.

REFERENCES

- [1] The PIAFE Project at Grenoble. J.L. Belmont et al.. International Workshop on the Physics and the Techniques of Secondary Nuclear Beams. Dourdan, March 23-25 1992. Edited by J.F. Bruandet, B. Fernandez and M. Bex, Edition Frontières.
- [2] The Physics case of the PIAFE Project. PIAFE Collaboration. ed. H. Nifenecker (ISN June 1994)
- [3] J. Jacobson, B. Fogelberg, B. Ekstrom and G. Rudstam, NIM B26 (1987) 223-226
- [4] The ISOL-MAFIOS Source. R. Geller, C. Tamburella, J.L. Belmont . 6th. International Conference on Ion Sources, September 10-16 1995, Whistler, B.C., Canada



PIAFE-PROPOSAL
GENERAL LAYOUT

Figure 1.

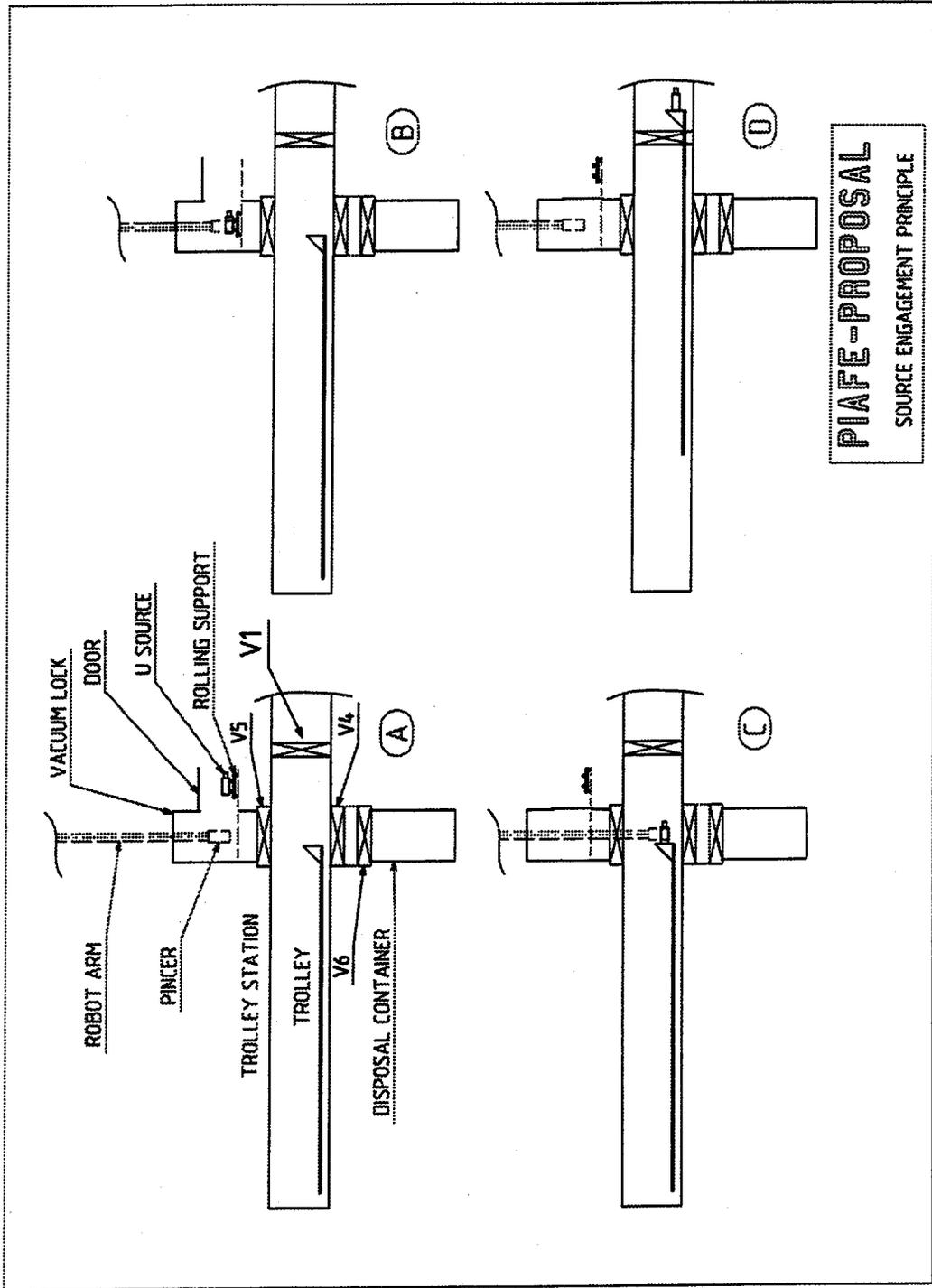
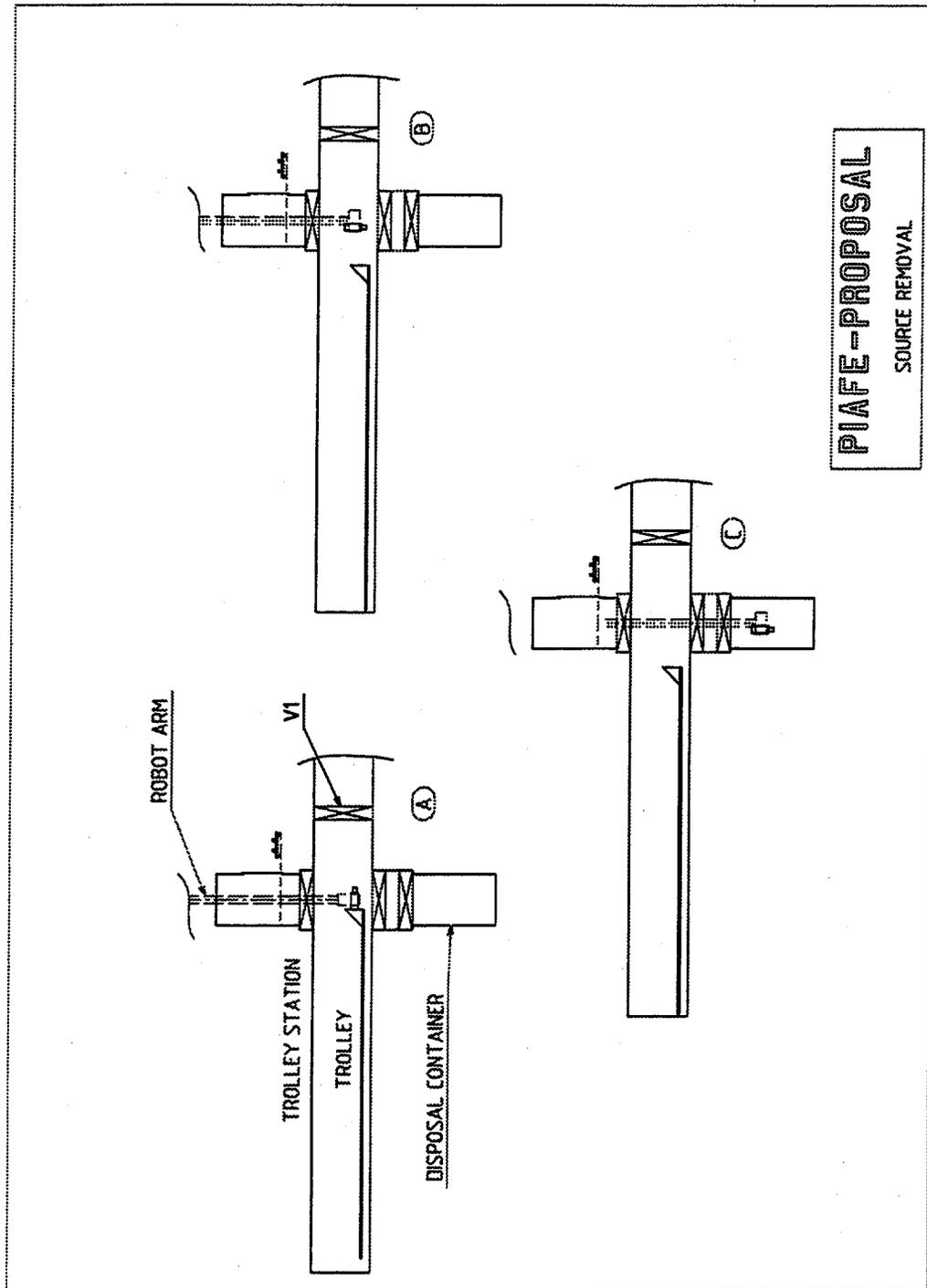


Figure 2



PIAFE-PROPOSAL
SOURCE REMOVAL

Figure 3.

VEPP-4 Control System

A. Aleshaev, A. Batrakov, S. Belov, A. Dubrovin, Yu. Eidelman, A. Kalinin, S. Karnaev, M. Kollegov, V. Kozak, E. Kuper, A. Ledenev, B. Levichev, Sv. Mishnev, A. Naumenkov, G. Piskunov, I. Protopopov, D. Shatilov, E. Simonov, V. Smaljuk, A. Smirnov, S. Tararishkin, Io. Zaroudnev
Budker Institute of Nuclear Physics
Prospekt Lavrentjeva, 11, Novosibirsk, 630090, Russia

Abstract

The VEPP-4 control system includes twelve CAMAC-embedded 24-bit in-house developed ODRENOK computers (CC24) [1], interconnected in a star network. The central node provides the boot for the peripheral computers, intertask communications between them and file server functions. Real-time intercommunications between some of the computers are based on point-to-point interfaces. Equipment control electronics is mostly CAMAC (nearly 60 crates). A real time multitasking OS permits running up to 10 tasks with fast dynamic change of core image. Control of VEPP-4 is accomplished by the simultaneous performance of more than 50 tasks. PCs are used as operator consoles, data processing computers, printer servers and back-up systems.

1 INTRODUCTION

The VEPP-4 accelerator facility [2] is composed of a 6 GeV collider VEPP-4M of 365 m in circumference, a 2 GeV multi-purpose storage ring VEPP-3 and a 350 MeV electron/positron injector. The injector consists of a 50 MeV LINAC and a 350 MeV B-4 synchrotron. The LINAC RF is supplied by pulsed GIROCON generator. There are pulsed transfer lines from the LINAC to B-4, from B-4 to VEPP-3 and from VEPP-3 to VEPP-4M (Fig.1).

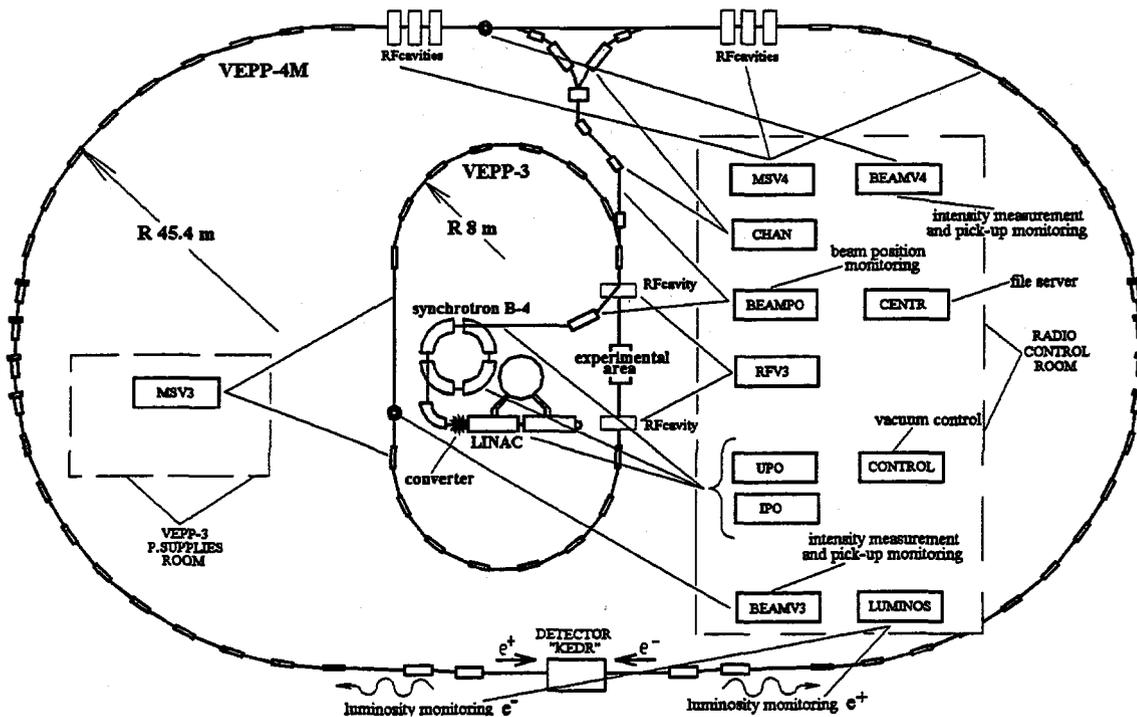


Fig.1 VEPP-4 facilities and control computers layout.

The collider provides the experiments with electron/positron beams, back-scattered Compton γ quanta and synchrotron radiation. VEPP-3 is an injector into VEPP-4M and is used simultaneously for experiments with synchrotron radiation and a polarized gas internal target.

The up-to-date control system has been developed since 1986 at the same time as the upgrading of the VEPP-4M main ring. All hardware and software have been developed and produced in the Budker Institute of Nuclear Physics.

2 ARCHITECTURE

2.1 Structure

The control system has three levels: the central node, the local computing layer and the equipment layer.

The local computing layer is composed of 11 intelligent CAMAC crates containing ODRENOK computers which are interconnected in a star network. All computers are connected to the central node via 250 kbit/s serial-links. The node provides for the initial downloading of the computers, file service and "slow" interprocess communications via a mail box. Special point-to-point 1.5 Mbit/s serial-links are employed for high-speed intertask communications in different local computers.

Usually each subsystem of the accelerator facility is controlled by its personal ODRENOK. An overview is shown schematically in Fig.2. All the active crates and the central node are located in a control room, except for the computer which controls the VEPP-3 magnet system. Table 1 presents the up-to-date configuration.

The front end electronics is mostly distributed among passive CAMAC crates connected to the active crates through serial-link drivers and controllers [3]. The distance between the active and the passive crates is up to 150 m. Special serial controllers are employed for connections with non-CAMAC electronics, used for control of the RF-systems and for beam diagnostics in the transfer lines. Coaxial cables are used for all links. The equipment control layer is dispersed around the facility.

The magnetic system of the VEPP-4M ring is controlled by distributed intelligent controllers [4] connected with CAMAC by MIL-1553-B links (Fig.2). The use of these controllers gives us improved control of the magnet system during the rise of the beam energy from 1.8 GeV to 6 GeV.

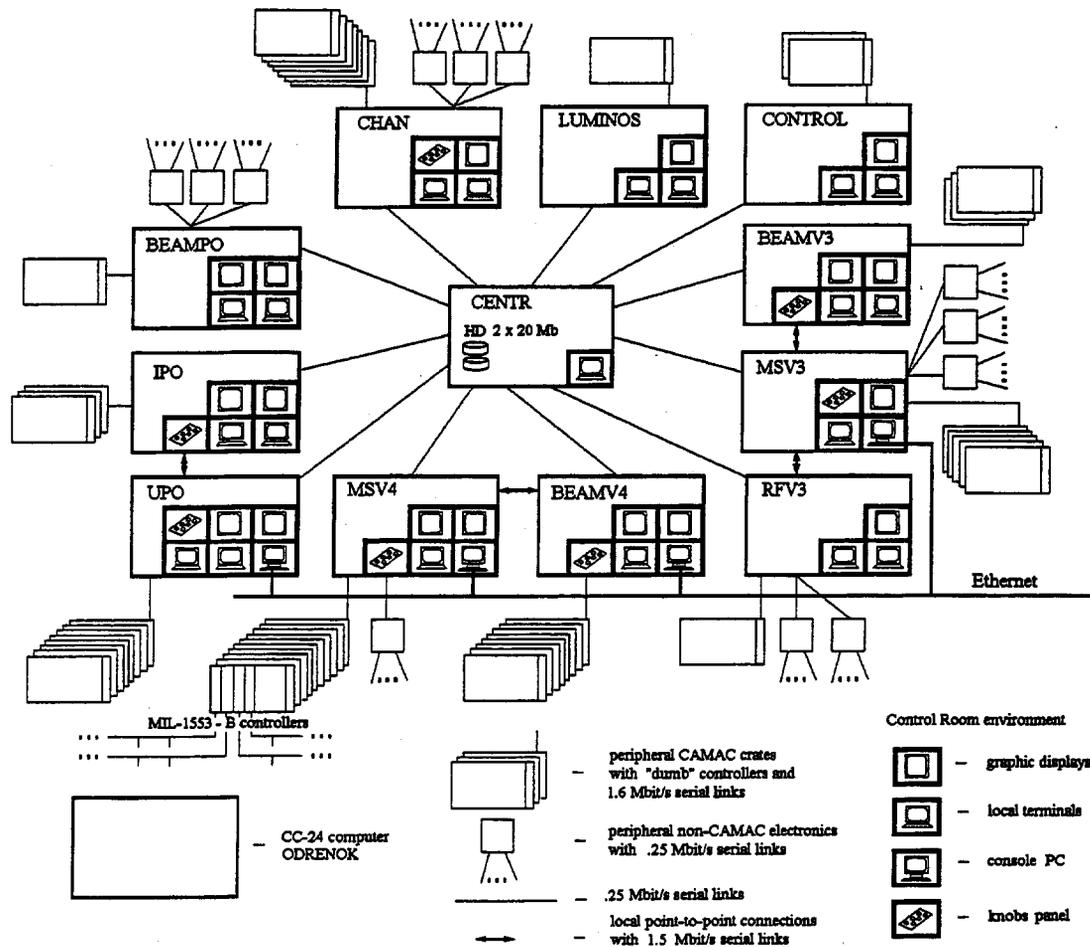


Fig.2 VEPP-4 control system architecture.

Table 1
VEPP-4 control system configuration.

	ODRENOK name	Allocation	CAMAC crates	Number of CAMAC modules	outputs	inputs
1	MSV4	control of VEPP-4M	12	70	370	950
2	BEAMV4	beam diagnostics at VEPP-4M	9	50		280
3	MSV3	control of the magnetic system of VEPP-3	5	49	125	290
4	RFV3	control of the RF system of VEPP-3	2	21	30	15
5	BEAMV3	beam diagnostics at VEPP-3	4	32		100
6	UPO	control of the pulse injector	9	75	135	30
7	IPO	pulse injector monitoring	4	47	10	110
8	CHAN	control of VEPP-3 - VEPP-4M transfer line	8	65	135	110
9	CONTROL	vacuum system monitoring	3	24		150
10	LUMINOS	luminosity monitoring at VEPP-4M	2	10		40
11	RADIAC	radiation monitoring	3	24		36
		Total	61	467	805	2111

2.2 Local computing layer

The local computer layer is based upon the CC24-computer, named ODRENOK. This computer was designed in 1983-1985 as an autonomous CC24 crate controller emulating the instruction set of ODRA-1300 (a clone of the ICL-1900 series mainframes) computer. The main features of ODRENOK are as follows:

- word length 24 bit;
- address space 4 M words;
- RAM size 64K 24-bit words;
- performance 0.5 MIPS;
- hardware implementation of floating point operations;
- USER/SYSTEM modes, multi tasking support, virtual memory;
- extensions to the initial ICL-1900 architecture: firm ware implementation of OS kernel, CAMAC-oriented instructions, vector instructions;
- 2M CAMAC module installed in the controller position;
- microprogram implementation on the AMD 2900 bit-slices

Each active crate has a four-port RS232 module for connection of up to four alphanumeric terminals, a number of color graphic display controllers, a RAM-emulated module (768 kb or 1.5 Mb), a network interface, and peripheral crate drivers. Now the alphanumeric terminals are gradually being replaced by IBM PCs, which are used as operator interfaces and supplementary disk machines.

2.3 Equipment control layer

A basis for the equipment control layer is a series of CAMAC modules developed at BINP. The series includes different types of DACs and ADCs [5], switches, etc. Tables 2 and 3 list the main parameters of some modules. Sixteen-output switches with electromagnetic relays are used to turn on and turn off power supplies and for interlocks. This type of CAMAC module has a 1ms switching time.

Table 2
Parameters of CAMAC embedded ADCs

Type of ADC	20-256	101-SK	850-SK	ZIIS-4M
Conversion type	integration	fast	fast	fast
Conversion time	2 - 240 ms	1 ms - 2 ms	50 ns - 2 ms	< 2 ms
Scale (binary)	13-20	12	8	10
Input voltage (V)	± 8	$\pm 1.2 - \pm 10$	$\pm 1.2 - \pm 10$	± 2
Number of inputs	1, MUX control	4	4	4
On-board memory (kbytes)	3	12	3	-
Module width	1M	2M	2M	1M

Table 3
Parameters of CAMAC embedded DACs

Type of DAC	CAP-16	CAP-20	GVI-8M	PKS-8
Type of output	analog	analog	delay pulse	pulse-width
Scale (binary)	16	19 + sign	16	16
Output range	$\pm 6.5V$	$\pm 8.2V$	0 - 838 ms	0 - 6.55ms
LSB	200 mV	15.6mV	.1 ms - 12.8 ms	.1ms
Number of outputs	16	1	8	8
Module width	2M	1M	2M	2M

3 SOFTWARE

3.1 Program environment

A multitasking real-time Operating System (OS) ODOS [6] was specially designed for ODRENOK. It enables us to run up to 10 independently-compiled memory-resident tasks. This OS supports intertask synchronous and asynchronous message transfers, remote file access, connection of up to 4 local terminals and one "system" virtual terminal attached through the network. The OS permits independent dynamic loading, running and termination of tasks.

TRAN (FORTRAN version) with an in-house developed compiler is used for application programs. Under ODOS its features fit well for memory mapping and real-time module control.

3.2 Data-base organization

The DataBase (DB) contains all the information about the devices of the accelerator complex. It has static and dynamic parts. The static database is stored in RAM-modules and on the hard disk of the central node. The dynamic database is a number of arrays in the Resident Executive Programs (REPs). DataBase Editing (DBE) programs provide access to the static database, for saving the operation modes and modifying the data files.

The organization of the static database is illustrated in Fig.3. The lower layer of the "topology" and "devices" branches is a linear array of text, integer and real variables, which includes names, addresses, status flags and calibration parameters. The "setting" branch includes the files, which are the operation modes for different parts of VEPP-4 accelerator facility.

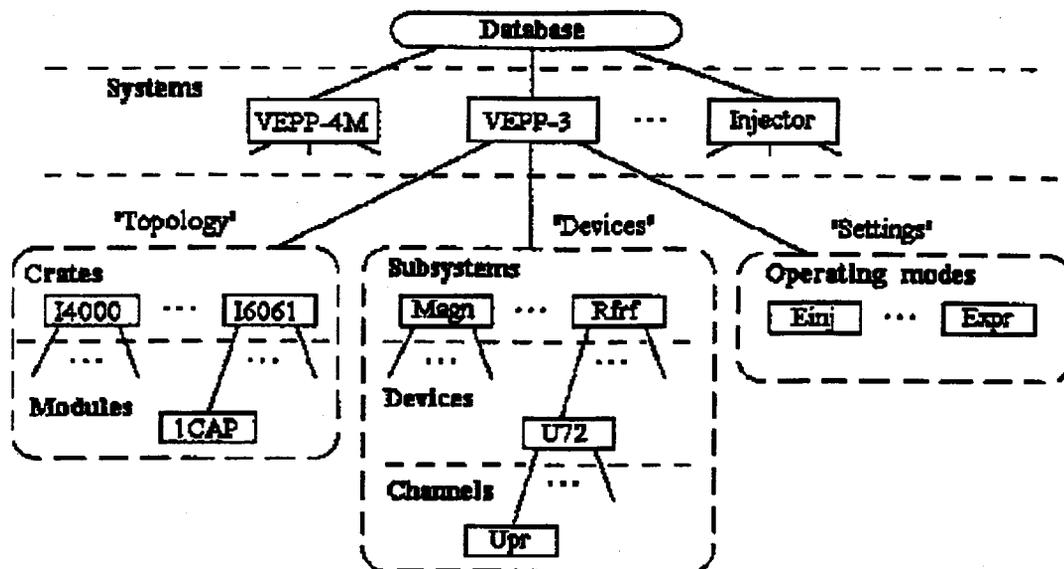


Fig. 3. Static database organization

REP is the resident modular program in each local computer (Figure 4). It contains the data, control algorithms and carefully optimized communication subroutines. All front end electronics are controlled by REP which protects the system from improper operation. Any newly developed electronics can be easily added to the program as plug-in modules. Control actions are performed by writing the required values into the dynamic database. REP allows us to have about 2000 - 4000 accesses/s to CAMAC modules.

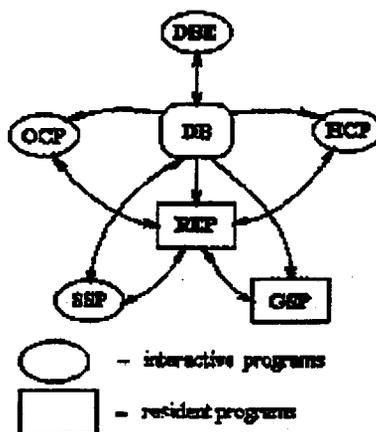


Fig. 4. Diagram of basic control software in the ODRENOK computer.

3.3 Applications

All applications were developed by the VEPP-4 staff requiring about 15 person-years. In addition, a large amount of time was spent on the development of special tools for commissioning and modeling programs.

The Operator Control Programs (OCPs) are implemented to handle the devices of the facility. Control values are entered from the keyboard, data are displayed as a column of digital values of device parameters. Saving and Setting programs (SSPs) save/set the operation modes.

Graphical Surveillance Programs (GSPs) display the parameters to be monitored in an illustrative color form. It allows the operator to identify easily any deviations from the assigned operation parameters. The status of the facility is shown in specially-configured graphical display images.

High-level Control Programs (HCPs) provide the automatic operation of VEPP-4 without operator intervention. HCPs perform:

- a) electron and positron beam accumulation and acceleration in the storage ring;
- b) beam transport to the collider from VEPP-3;
- c) the rise of beam energy in VEPP-4M from 1.8 GeV to 6 GeV.

A mailbox in the central computer is responsible for connections between the HCPs in different computers. The mailbox consists of a number of modules, each of them corresponding to a particular ODRENOK. Each module contains status information about the subsystems controlled by that ODRENOK ("operational ready", beam energy, beams current, etc.) and includes control commands for other computers (for example, "reverse polarity", "rise energy", etc.)

4 CONCLUSION

Although the present control system successfully provides for the operation of VEPP-4, it is obvious that an upgrade is necessary. The first goal is the installation of ethernet links between the local process controllers. This will allow us to use a top level based on workstations with standard software without loss of functionality. The second goal is wide use of distributed processors at the lowest control level.

5 REFERENCES

- [1] G.Piskunov, *Autometria*, N4 (1986), p. 32-38 (in Russian).
- [2] 1994 BINP Yearly Report, BINP, Novosibirsk (1995), p. 81-87 (in Russian).
- [3] V.Kozak, Preprint INP 88-24 (in Russian).
- [4] S.Kamaev et al., Control of VEPP-4M Magnetic System, these Proceedings.
- [5] E.Kuper, A.Ledenev, ICALEPCS 1991, KEK, Tsukuba, Japan, p. 382-385.
- [6] A.Aleshaev, Preprint INP, Novosibirsk (1989) 88-24 (in Russian).

Control of the QUENCH Protection System at HERA

R. Bacher, P. Duval, S. Herb, K.H. Mess and H.G. Wu
Deutsches Elektronen Synchrotron (DESY), Notkestrasse 85, 22603 Hamburg, Germany

Abstract

We describe the control of the Quench Protection System for the superconducting magnets of the HERA proton storage ring in DESY. General control (operator to hardware) follows "standard model" -like concepts with multiple consoles communicating with front ends via the ethernet. The front end control is based on redundant VME CPUs running the VxWorks real-time operating system. The data-link to the lower level quench microprocessor and PLC based alarm control center is connected via the CAN fieldbus. A network server task communicates with consoles running MS WINDOWS and Visual Basic via UDP. Important quench data are automatically archived following critical events, allowing follow-up expert system analysis while the machine is brought again into operation.

Introduction

The HERA proton ring consists of 422 superconducting dipole magnets and 224 superconducting quadrupole magnets. These magnets are cooled by 4.5 K liquid helium. A current of 5025 A goes through all these magnets for a proton beam energy of 820 GeV. The superconducting magnets are ultimately protected via hardware as opposed to relying exclusively upon software logic, the quench detection electronics being monitored independently. Passive quench protection is achieved by using bypass diodes. Active quench protection is carried out by using transducers to measure the current flow in bridges to detect quenches, and then activating the corresponding dipole heaters to protect the magnets and operating the main current switches. Furthermore, redundant components are integrated into the system as much as possible, at both at the hardware and software levels. The present system uses front end VME-based CPUs running VxWorks and makes use of the CAN fieldbus and PLC modules. A microprocessor-based computer provides local intelligence for monitoring, testing and fault state detection of all the quench hardware components. A PLC-based microprocessor functions as an alarm control center collecting alarm signals, such as quench electronics status, power status, status of switches and alarm status from beam loss monitors. This processor makes critical decisions regarding beam dumps and machine operation. Various hardware test commands can be executed from the control system consoles. The current state of the quench protection system is collected asynchronously at the consoles from the VME CPUs via the ethernet. The front-end CPUs and the quench microprocessor and alarm control center are linked via the CAN bus. Logically the VME CPU acts as a front end supervisor, collecting data, coordinating the behavior of the quench microprocessors and alarm control center and serving as a network server.

A more detailed description of the quench protection system at HERA is given elsewhere [1]. In this report we concentrate on the design of the software control, starting at the level of the VME crate and progressing to the GUI at the console level. Specific details of the hardware controllers such as the PLCs will not be presented here. A schematic of the system architecture is given in figure 1.

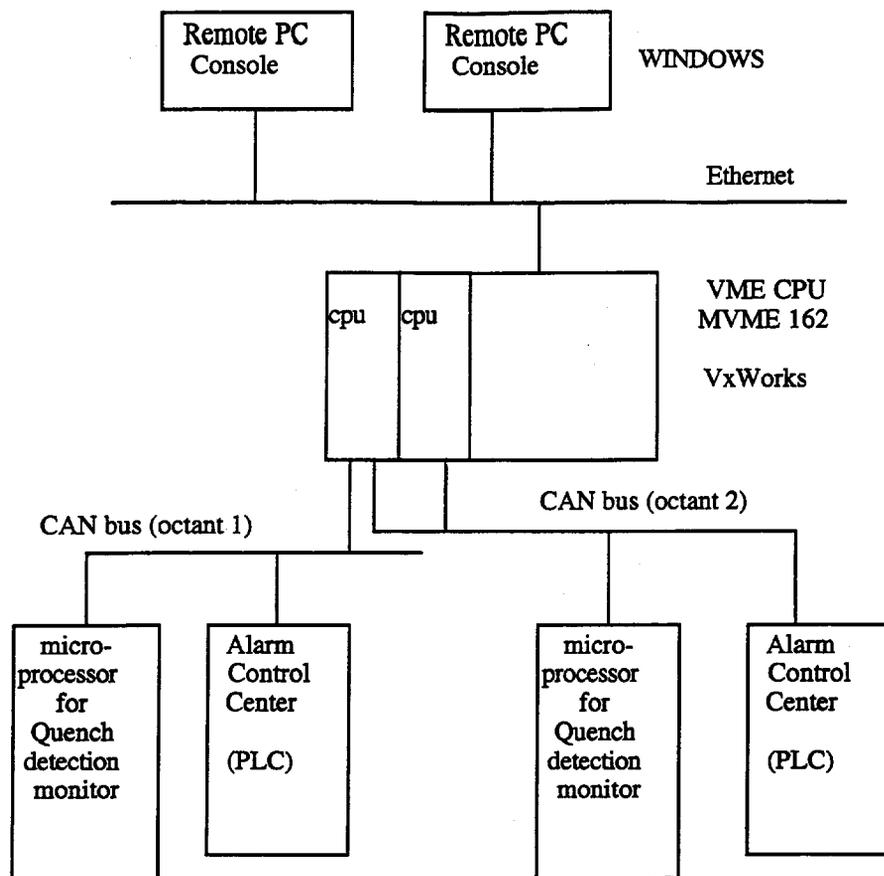


Figure 1. Schematic drawing of the hardware architecture for a quadrant

The Console Software

We use PCs running MS WINDOWS 3.1 in the machine control room as consoles. Several application programs written with VISUAL BASIC are responsible for controlling and monitoring the quench protection system. Data exchange between application programs and the front end servers is via ethernet. We use a socket-based RPC-like protocol, developed for the HERA PC control system [2]. Changes of state are noted promptly at the console via asynchronous messages from the front end. Furthermore, the front end appears as an integrated device at the console, allowing specific commands, such as to activate a heater test, to operate specified main current switches, or even to dump the beam to be performed by a mouse click on a synoptic display. The total response time (mouse click to hardware setting and return) for command execution is typically less than one second.

Front End Hardware

A total of eight MVME162 CPUs are used for front ends, two CPUs being housed in a VME crate in each of the four HERA halls. They are used to control and monitor quench operation and the alarm center. One of these CPUs is always executing as the master, the other as a redundant partner. The data-link to the lower level quench microprocessor and PLC-based alarm control center is provided by the CAN fieldbus. Two separate CAN buses are connected to each of these CPUs. One CAN bus connects one alarm control center and four quench microprocessors. Two of these four microprocessors provide redundant control for the dipole magnets, and the others deal with the quadrupoles, and groups of dipoles and groups of quadrupoles. The CAN data transmission speed is set to 500 kHz, and the overall time of one CAN telegram transmission is 250 μ s.

Live insertion cards are used for the VME CPUs. One can remotely or locally power one CPU on or off, without disturbing the operation of the others. All front end hardware is powered through an uninterruptable power supply (UPS). Furthermore, each front end in the hall operates independently of the others and is capable of operating in a stand-alone manner.

Front End Software

The front end software has been developed for the VxWorks real-time multitasking operating system. A network task services remote consoles and runs at rather low priority. As mentioned above, the HERA PC control RPC protocol is used for updating registered consoles. Commands which change the state of the machine can only be executed by allowed users.

A PLC task is responsible for communication with the alarm control center, by sending a request command via the CAN bus every second. A hardware redundancy check is also performed upon request from the alarm control center. According to the result, it will send an alarm to the alarm center to dump the beam if warranted. A message queue is used for collecting console commands from the network task. A test task can be executed for testing transducer and heater electronics if the console application sends a test request.

A CAN message receive task collects CAN bus data, and sorts and distributes them to mailboxes; it operates on semaphores according to CAN-IDs. During quench detection, it will perform heater status checks and send heater commands to the quench microprocessor if necessary. Quench telegrams come at speeds of 200 Hz, the maximum speed from 4 crates at once being 800 Hz. This means that the CAN data transmission must take place less than 1ms. The 250 μ s message transfer time mentioned above is more than adequate. A massive quench situation has been simulated with programs written for VxWorks, to test this extreme situation. No performance problems were detected.

A monitor task, running at 10 Hz, checks the status of all other running tasks and the data integrity with that of the redundant CPU, as well as the status of the CAN bus connections. According to this check it marks itself as 'OK' or 'NOT OK', and sends its status information to its partner CPU via the CAN bus. Upon receiving this CPU status information, it will determine whether it should continue to act as slave or take over as master.

Only the master CPU responds to the console commands and sends commands via the CAN bus to the quench microprocessor and alarm center. Although the slave acts as a passive listener, it runs exactly the same code and possesses all current information.

In the unlikely event of a CPU hang, a 'COLD BOOT' is still possible by toggling the power on the live insertion card remotely from the console program.

The Quench Archive

A dedicated client application running on a PC is responsible for archiving important data upon quench or beam loss alarm. It monitors the quench status and initiates a data archive if necessary. The archive data contain the complete state of the machine at the time of the quench or other critical events. The experts can then use these data at their leisure while the machine is brought up into operation again. The archived data are read by the same application program in the same way, but the data link is to an archive file, instead of the front end program.

Performance

The current system has been in operation since April 1995, and operates as designed. Although the whole system is composed of so many different hardware and software components and involves data exchange from different platforms on the field bus, it has proven to be a reliable system. During the half-year of operation, the built-in master-slave redundancy has not been invoked.

Acknowledgements

We would like to thank all colleagues who have been involved in this project.

References

- [1] K.H. Mess et al. "Quench Protection at HERA", proceedings of HEACC 92 in Hamburg
- [2] P. Duval, "Implementation of PCs in the HERA Control System", ICALEPCS 95 in Chicago, these proceedings

Controlling telescope observations from the astronomer's own desk: the case of TNG

A.Balestra, F.Pasian, M.Pucillo, P.Santin, C.Vuerli
Osservatorio Astronomico di Trieste
Via G.B.Tiepolo, 11
I 34131 - Trieste - Italy

ABSTRACT

High-level ground observation facilities for astronomy are located in isolated sites where atmospheric and light pollution conditions are the best possible. Normally astronomers travel to such observation sites, but in the last decade some major observatories started a remote control program on selected telescopes; more recently, the concept of allowing remote observation from secondary control centers has become a reality. The availability of bandwidth-on-demand networking services and the construction of new telescopes having integrated control systems, allow the user at least conceptually to perform observations from his own desk.

Teleoperation of telescopes is one of the topics of a project submitted to the European Union for funding and it has recently been approved: the aim is to monitor and control remotely two different telescopes located on the Canary Islands using a common system and standard ISDN services. One of these telescopes is the Italian TNG, which is discussed here as a test case. TNG has been designed and is being implemented with remote control as an essential requirement.

1. THE STATE OF THE ART IN TELESCOPE REMOTE CONTROL

In order to achieve the image quality needed for top-level research in astronomy and astrophysics, the effect of Earth's atmosphere on the observation needs to be minimized. The solutions are either to implement costly space-borne observing facilities or to install ground-based telescopes in remote regions, with low humidity, good weather conditions, and no atmospheric and light pollution: in other words, ones having "good seeing conditions". The latter has been traditionally the way astronomical observing facilities have evolved and ground-based telescopes still have a fundamental role to play in the development of human knowledge.

For the traditional astronomer to perform an observation in one of the best ground-based observatories means reaching isolated observing sites, where the observations are actually carried out, through uncomfortable trips. The availability of an advanced telecommunications technology suggests nowadays a different and more practical solution to this problem: controlling the telescopes and performing the observations remotely.

The first attempts at controlling remotely astronomical telescopes go back to the first astronomical observations from space (in the late 1960s). Up to now, remote control facilities have been implemented by providing a point-to-point connection through which the telecommands to control the basic telescope and instrument operations are sent. If the bandwidth of the communication channel is large enough, the observational data can also be sent; this is of course mandatory in the case of space-borne experiments, while it is not common practice for the ground-based observatories using remote control. A review of recent remote control experiences is given in [1]. Among these, the European Southern Observatory (ESO) is to be noted: since the mid 1980s, some of the ESO telescopes located

in La Silla, Chile, can be remotely operated from the ESO headquarters in Garching, Germany and this has become standard practice [2-3].

On the contrary, the concept of distributed remote observation, *i.e.* bringing the tools to perform and control an astronomical observation to the user's site, is relatively new. In one case (from UK to Mauna Kea, Hawaii) the concept of *passive distributed observing* is introduced: standard Internet lines are used to monitor the observing activities and communicating with the local staff physically performing the observations, while no real-time activity is supported [4]. A second experiment has been tested in 1992 between the Trieste Astronomical Observatory (OAT) and the ESO facilities in La Silla, using the remote control system at the ESO headquarters as a relay. In this case, a 64 kb/s point-to-point temporary connection has been set up between Trieste and Garching to support the test, in addition to the permanent point-to-point connection between Garching and La Silla. The test system allowed the real-time control of the telescope and instrument setup and functionality and the organization of the observations from dedicated computers located at the OAT site. The different aspects of this experience have been reported in [5-7].

2. THE TNG TELESCOPE AND ITS CONTROL SYSTEM

One of the first telescopes of the new generation, *i.e.* having an integrated control system, is the TNG (Telescopio Nazionale Galileo), a 3.5 m telescope which will be fully operational at La Palma at the end of 1996. TNG is being built as a joint effort of the Italian astronomical community under the coordination of the Observatory of Padua, with Prof. C.Barbieri as the Head of the project. The mode of operation for TNG is envisaged to be based on an end-to-end data flow model, and on assisted observing, with the possibility of flexible scheduling. Information on TNG and its instrumentation can be found in [8-12].

Optically and mechanically, TNG can be considered as basically derived from the ESOs NTT, while the design of its control system is new and original. The TNG control system [13] is based on a distributed computer architecture over which two main software environments are layered: the Telescope Software System (TSS) runs on the local processors for telescope and instruments, and provides the direct interface to the hardware; the Workstation Software System (WSS) runs on the control Unix workstations, monitors the TSS activities, and provides the higher-level interface for the user. The choice of using a distributed system at the local processor level maximizes the performance of the system while minimizing the cabling; at the workstation level it enhances reliability (the WSS is able to recover from failures semi-automatically, without losing control over the TSS), while allowing a multi-instrument multi-user environment.

Due to flexibility and reliability requirements, the WSS [14] has been implemented distributing both the code and the data on the workstations supporting the system at the telescope. The code has been distributed using the textual symmetry paradigm, *i.e.* all workstations run the same code but behave differently according to their startup configuration and/or to the messages they receive during operations. Data are distributed using a unique replicated data-base structure divided into a number of subsets, each maintained by a different workstation: the access to data not pertaining to the local system is guaranteed by a message exchange mechanism.

The system is designed in such a way that the physical location of the various workstations running under WSS is irrelevant. This means that remote control of the TNG is implicitly embedded in the design [7], since the control workstation running the WSS code and sharing the data base structure can either be located on the telescope LAN, or be remote connected via a bridge or a router.

The structure of the WSS is shown in Figure 1, where three workstations (the Telescope workstation, the Instrument workstation and the User remote workstation) are shown: each of them runs WSS, but some of its tasks (depicted in white) are inactive.

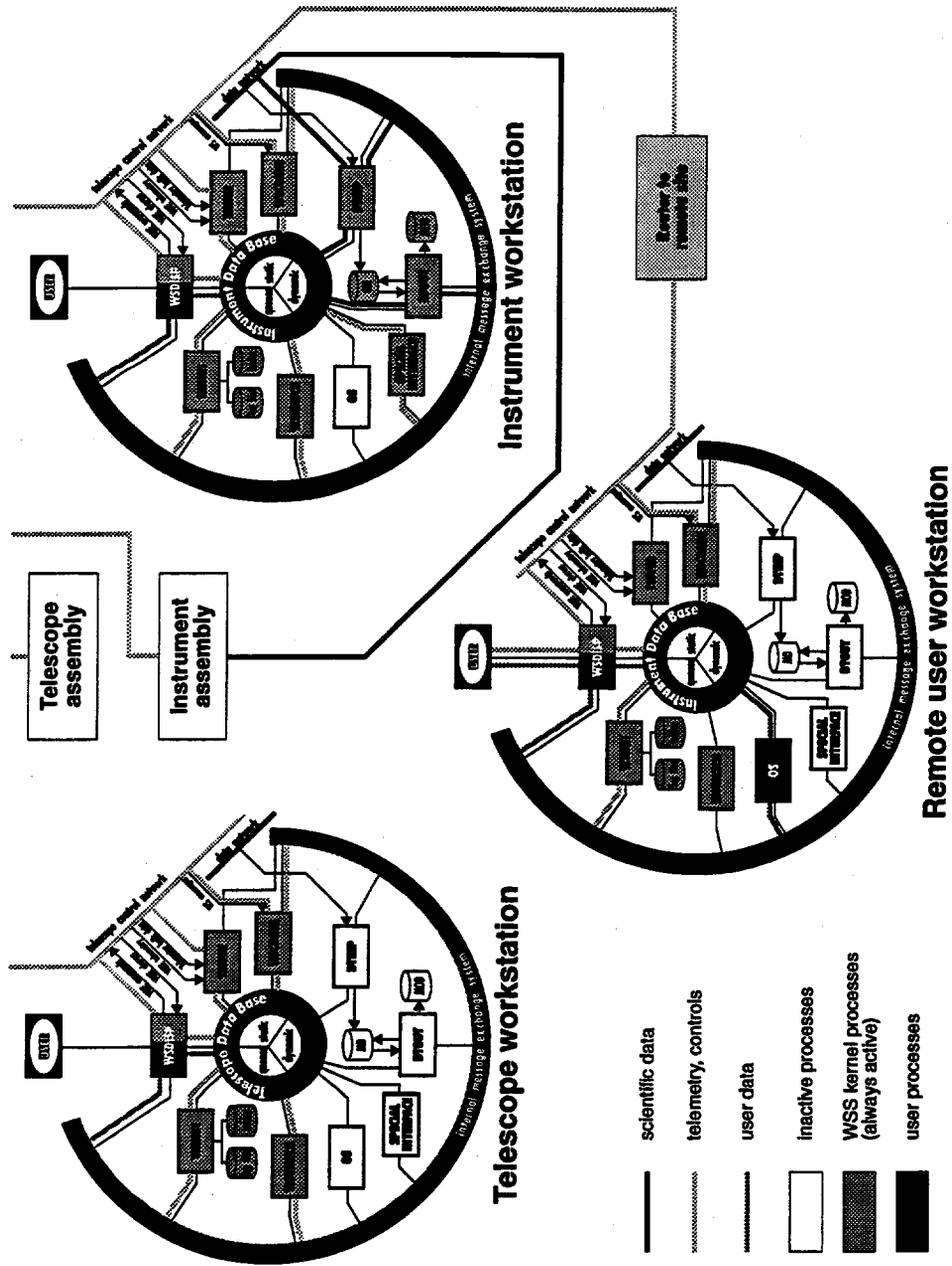


Figure 1

3. THE "REMOT" PROJECT

REMOT (Remote Experiment MOnitoring and conTrol) is a project recently approved by the European Union; its objective is to develop a system architecture to allow remote control of scientific experiments and facilities that require real-time operation and multimedia information feedback and using available or deploying new communications infrastructures. The availability throughout Europe of bandwidth-on-demand services allows the user to control experiments and facilities from his desk, provided that the overall system is designed with this concept in mind. The purpose of the project is to build a generic teleoperation system, using as much as possible available elements from other projects or off-the-shelf, together with *ad-hoc* software modules. The communications infrastructure will be IP, complemented with real-time protocols and the availability of ISDN and ATM technology shall also be taken into account. Products from the Internet domain shall be used whenever they do not conflict with real-time requirements.

Representatives from two user communities participate in the project: the astronomical community, whose top-quality telescopes are located in isolated sites to optimize seeing conditions (as discussed above) and the plasma physics community that is concentrating the experimental facilities in a few places in order to save costs. The astronomical community is represented in REMOT by an Institute operating a full-fledged observatory (the Instituto de Astrofísica de Canarias (IAC)), an Institute defining and building the control system for a telescope (the Osservatorio Astronomico di Trieste (OAT)), and an Institute representing the astronomical user (the Laboratorio de Astrofísica Espacial y Física Fundamental (LAEFF)), plus three additional Institutes (CDS, KIS, Lund, VILSPA) to set up requirements. TCP Sistemas y Ingeniería is associated to the project as an industrial partner.

The system will be built using a Client/Server architecture. The Client side, running at the user premises, will hold the user interface and display facilities, the commanding interface and processing and communication facilities. The Server side, located at the service provider facilities, will include the actual interface to the provider system, the processing, communication and management services and the "local operator" interface that will provide supervisory facilities together with the normal user interface. The concept of the REMOT architecture is shown graphically in Figure 2.

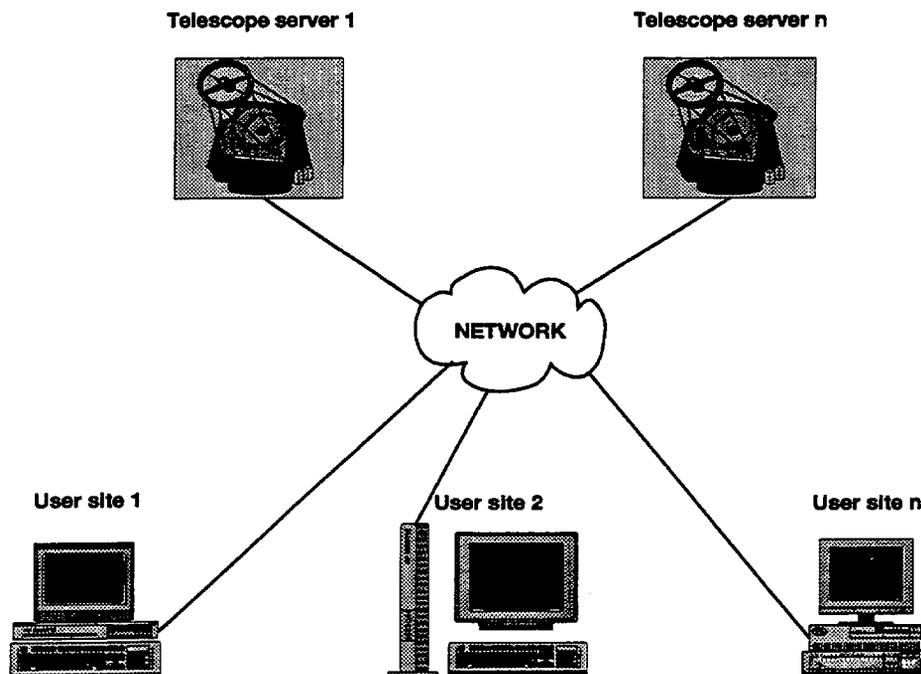


Figure 2

The services that REMOT is planning to provide as a demonstration can be roughly divided into three categories:

- **monitoring activities**, aiming at following the basic operations to be held at the telescope, just to make sure that operations are running smoothly. No control operation is performed, therefore no real-time activity is really required; low-speed services are sufficient;
- **remote control activities**, aiming at controlling the activities of telescope and instruments, including performing real-time operations. Guaranteed bandwidth for the support of real-time operations is an absolute requirement; but the speed does not need necessarily to be high;
- **remote observing activities**, including monitoring and some of the control activities, such as the logical setup of basic modes and functions of telescope and instruments. Guaranteed bandwidth for the support of real-time is required; medium- to high-speed is needed depending on the services provided. If the transfer of scientific data files (lossless compressed) is provided, a high-speed link is mandatory.

Using the IP solution will make remote scientific activities available across a wide range of platforms and media. Low speed services will be tested over Internet, real-time services and high speed transfer services will be tested by using IP over ISDN, or possibly over ATM.

A specific point is related to the expanding WWW Internet facilities available today. They could be used to cover partially some of the services involved, especially in the field of user interfaces, as discussed in [15], probably combined with additional facilities. The possibility of including an adaptation of a WWW server and client respectively will be analyzed. Due the nature of WWW, its use for real time audio and video communications would probably be prohibitive, so this kind of services needs another types of developments and tools.

A candidate for performing real-time control and transporting real-time multimedia data (such as interactive voice and video) is the new Internet protocol RTP (Real Time Protocol). RTP is a protocol that facilitates the transport of real-time data over packet switched networks, in particular IP. Efforts have been made to make RTP transport-independent so that it could be used over AAL5, CLNP or other protocols.

The REMOT project will allow remote observations and the simultaneous use of network facilities (*e.g.* access to astronomical archives and databases for the retrieval of information and historical data) directly from the user's institute. This solution will allow the control of observations, immediate processing of acquired data and comparison with archive data to be performed in the familiar everyday working environment. There are furthermore a number of other positive impacts: the possibility of immediately processing acquired data at the user's site reduces the need for powerful computing infrastructures at the telescope; no point-to-point connections are needed and the expense of communications can be shared among all institutions using the observing facilities on a time-connected basis; expenses for telecommunications can be recovered by the individual institutes from the money saved by avoiding scientists' trips to the observing facilities.

Integrating the TNG telescope in the REMOT project is conceptually simple, since the TNG control system has been designed and implemented to include transparent remote control as one of its features. As a first step, the implementation of the TNG concept on ISDN-based connections will be tested; later, an appropriate software interface between the TNG WSS and the new generalized user interface which will be provided by REMOT will be built, in order to remotely control all of the telescopes included in the project through a unique interface. In any case, the TNG project will bring its own experience to test a concept that may dramatically change astronomical observing in the future: controlling a telescope from the user's own desk.

ACKNOWLEDGEMENTS

The REMOT project is being financed by the European Union (DG XIII), and involves a number of institutes in Astronomy (IAC, LAEFF, OAT, and CDS, KIS, Lund, VILSPA) and Plasma Physics (KFA, Utrecht), besides industrial partners (TCP/SI and ETSIT-CMA). The TNG project is funded by the Italian Ministry of University and Scientific and Technological Research (MURST) through the Council for Research in Astronomy (CRA).

REFERENCES

- [1] D.T.Emerson, R.G.Clowes, eds., *Observing at a Distance*, World Scientific Publ., 1993
- [2] G.Raffi, M.Ziebell, Remote Control of 2.2-m Telescope from Garching, *The ESO Messenger*, 44, p. 26-29, 1986
- [3] A.A.Zijlstra, J.Rodriguez, A.Wallander, Remote Observing and Experience at ESO, *The ESO Messenger*, 81, p. 23-27, 1995
- [4] R.G.Clowes, Remote Observing with the JCMT and UKIRT, in: *Observing at a Distance*, D.T.Emerson, R.G.Clowes, eds., World Scientific Publ., p. 1-8, 1993
- [5] A.Balestra, P.Bonifacio, M.Centurion, M.Comin, M.Franchini, P.Marcucci, M.Nonino, F.Pasian, L.Pulone, M.Ramella, P.Santin, G.Vladilo, C.Vuerli, A.Wallander, The ESO/OAT Second Level Remote Observing Project: Final Test on ESO/NTT, *OAT Publ.* 1444, 1992
- [6] A.Balestra, P.Santin, G.Sedmak, M.Comin, G.Raffi, A.Wallander, NTT Remote Observing from Italy, *The ESO Messenger*, 69, p. 1-5, 1992
- [7] A.Balestra, M.Pucillo, P.Santin, G.Sedmak, C.Vuerli, Remote observing activities at the Trieste Astronomical Observatory - the second level remote observing with ESO/NTT and the Galileo Telescope, in: *Observing at a Distance*, D.T.Emerson, R.G.Clowes, eds., World Scientific Publ., p. 213-224, 1993
- [8] C.Barbieri, R.Bhatia, C.Bonoli, F.Bortoletto, G.Canton, A.Ciani, P.Conconi, M.D'A-lessandro, D.Fantinel, D.Mancini, A.Maurizio, S.Ortolani, M.Pucillo, P.Rafanelli, R.Ragazzoni, M.Zambon, V.Zitelli, The Status of the Galileo National Telescope, *Galileo Project Technical Report*, 36, 1994
- [9] C.Barbieri, The GALILEO Italian national telescope and its instrumentation, in: *Highlights of European Astronomy*, Kluwer, 1995, in press
- [10] F. Fusi Pecci, G. M. Stirpe, V. Zitelli, eds., *TNG Instrument Plan: II - A progress Report*, Osservatorio Astronomico Bologna, 1994
- [11] S.di Serego Alighieri, L.Benacchio, C.Bonoli, F.Pasian, Cyclic operation scheme for the TNG, *Osservatorio Astronomico Padova-Asiago*, 1994
- [12] TNG WWW Home page, <http://www.pd.astro.it/TNG/TNG.html>
- [13] M.Pucillo, The TNG Instrument Workstation and the handling of data, in: *Handling and Archiving Data from Ground-Based Telescopes*, M.Albrecht, F.Pasian, eds., *ESO Conferences and Workshop Proceedings*, 50, 1994
- [14] A.Balestra, P.Marcucci, F.Pasian, M.Pucillo, R.Smareglia, C.Vuerli, Workstation Software System - Architecture Design Document, *Galileo Project Technical Report*, 9, 1991
- [15] A.Balestra, P.Marcucci, F.Pasian, M.Pucillo, R.Smareglia, C.Vuerli, Using NIR tools for the interfaces to the help and archive systems at the TNG telescope, *Vistas in Astronomy*, 39, 1, 1995

STATUS REPORT OF THE PSI ACCELERATOR CONTROL SYSTEM

T.Blumer, D.Anicic, I.Jirousek, A.Mezger and L.Sekolec

Paul Scherrer Institute, Villigen, Switzerland

Abstract

The upgrade of the control system for the three accelerators and their associated beam lines is now near completion. Some of the major improvements of the last upgrade include the replacement of 16 bit computers PDP11 with RISC machines in VME. Aspects of performance, timing and realtime response are analysed and results measured in the real system are presented.

HISTORY AND STATUS OF THE CONTROL SYSTEM

The control system history shows continuous improvements and several major upgrades which reflect the continuous development of the accelerators. The present state now represents the third generation in a long line of evolution of the control system.

The first system started with one "big blue", a very precious central computer acting as an intelligent operator. Settings could be saved and restored, probe measurements analysed, and beam profiles captured for later analysis of the beam line using the data processing centre of the institute. At this first stage manual control was performed directly through dedicated hardware. The second generation introduced 16 bit minicomputers with CAMAC for communication and input/output to the process. Three such minicomputers were used, one for each of the three accelerators.

The proposal for the present upgrade of the PSI accelerator control system was first outlined in 1989 in Vancouver [1]. In 1993 in Berlin [2] more details followed as the project advanced and first results were presented.

Limitations of the minicomputer system, the development in computer hardware and the ever present demand for extension in size and quality of the system created the motivation to undertake the latest upgrade, where three distinct areas of change and improvement can be identified.

Firstly, the minicomputers, old and obsolete 16-bit machines (PDP11/44 with RSX11M+) had obviously to be replaced. At the same time we intended to increase the I/O bandwidth by one order of magnitude and restructure the front end software for better maintainability and more flexibility.

Secondly, the introduction of a distributed system based on messages for communication. This would allow global access to all parameters throughout the whole system, independent of the hardware configuration and the computer where an application is executed. It would improve reliability by separation of applications from the input/output on the front end computers. This is a solution where the demand for additional I/O performance can be satisfied, independent of whether the number of connected devices is increased or the update frequency is raised.

Thirdly, workstations with graphical user interfaces for the operator consoles were to be introduced. These would replace the old hardware panels by more flexible workstation displays and provide modern platforms for large and complex applications including a common standardised application process interface for the user.

STRUCTURE OF THE PRESENT CONTROL SYSTEM

The basic system upgrade is nearly finished, if measured by the degree of introduction of the upgraded front end computers. It is now completed to 80%. The system is now fully scaleable in size and performance.

The upgrade of the front end computers to the new machines was done in steps. The first step consisted of a reconfiguration of CAMAC crates in hardware and in the central database and the second of a test with the new front end. This was done in one service day. The new configuration was then taken into operation on a subsequent service day. No additional interventions were needed for the applications, except some bug fixes.

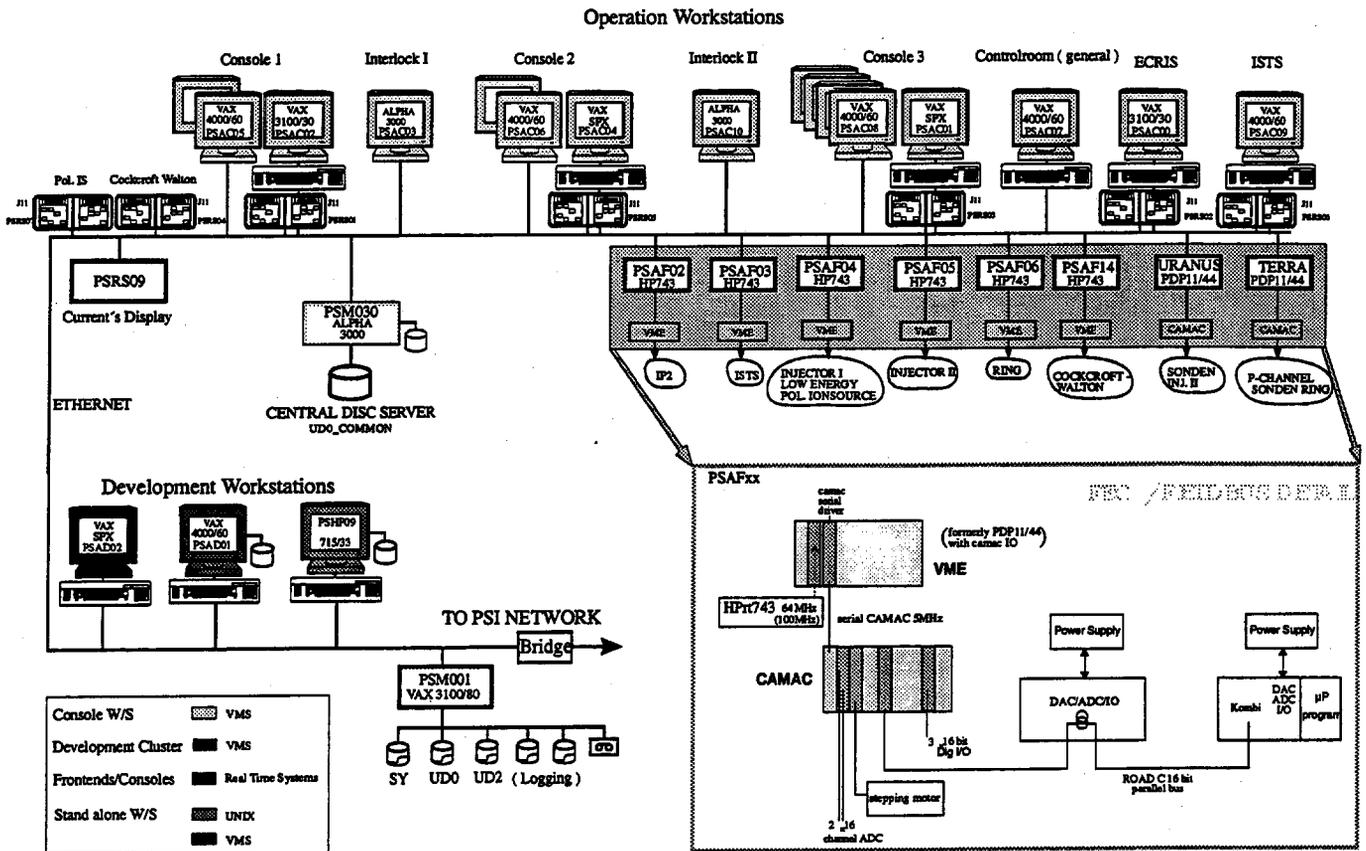
The last residual effects of the move will be cleaned up by late spring 1996.

HARDWARE DESCRIPTION

The front end computers are formed of HPrt743 (64MHz) RISC machines in VME. The accelerator devices are connected via CAMAC and local fieldbusses. The CAMAC crates are accessed via bit-serial loops (5 MHz). The serial loop is controlled by a CERN serial CAMAC controller [3].

Workstations provide the operator interface at the control room console. Dedicated nodes for knobs and touch panels are CAMAC-based PDP11 microprocessors (CES Starburst) connected directly to Ethernet. A DEC Alpha provides disk space for global data used in the system. Additional nodes are used for auxiliary applications. Development machines are separated from the machines needed for operation.

FIG 1: Present Control System Structure



COMMUNICATIONS AND SOFTWARE

The communication used in our distributed system is based on connectionless messages using the basic Ethernet protocol. This makes a loose coupling between FECs and workstations and ensures that no deadlock can occur. The format and the content of these messages are standardised on the basis of logical I/O access.

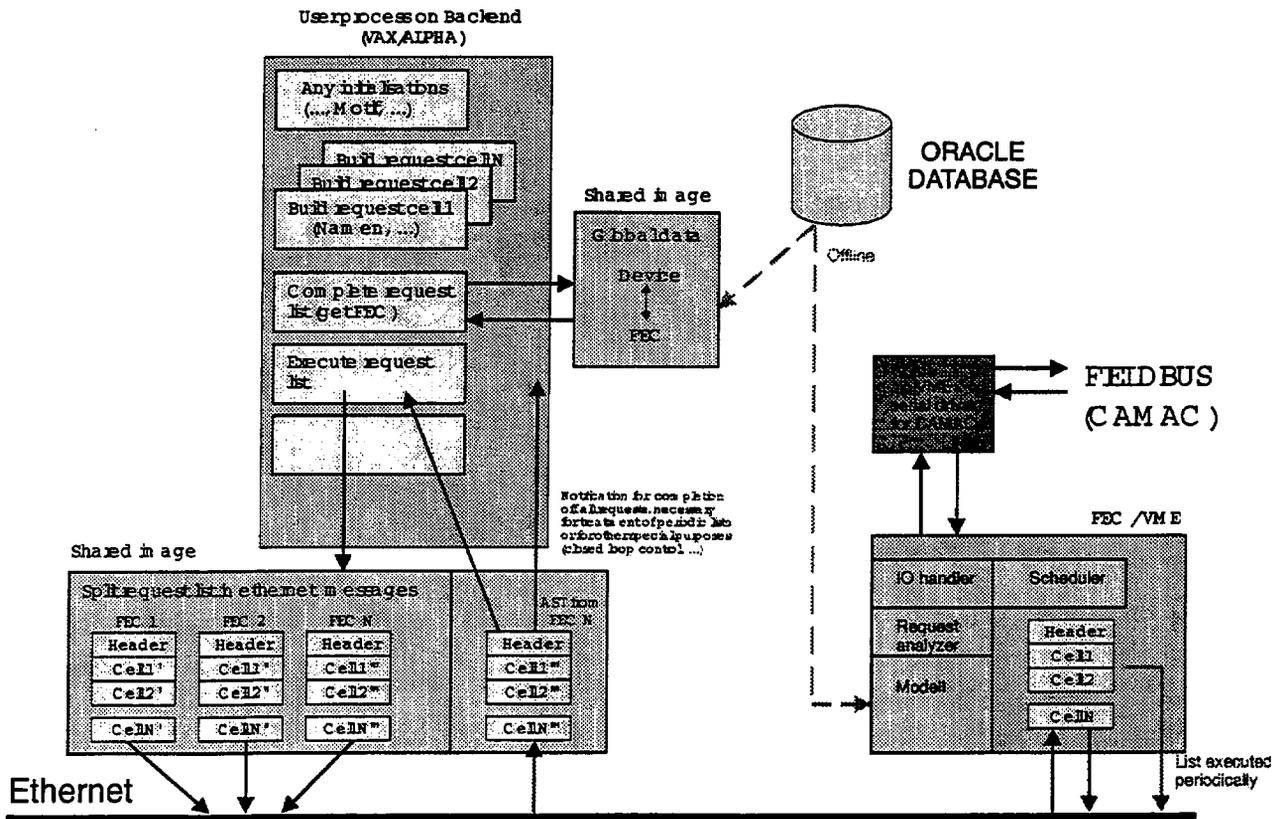
The messages on the network form an operating-system-independent interface between front end and requester. This was a pre-requisite for the replacement of the frontend machines. The front end software transforms logical I/O requests from the network into the appropriate hardware operations needed. This is done in a model representing the connected hardware. To construct this model the object oriented language Sather is used. The information for building the model is extracted from an Oracle database.

The user support in the workstations (open VMS) consists mainly of a general user application interface providing the logical access for I/O to the accelerator devices. This is realised in the form of a shared image.

X-window-based MOTIF is used as a graphical user interface for animated displays. Graphx, a PSI developed graphic package, is used for static graphic applications.

Workstations, some of them forming a VMS-cluster, are used for the development of the control system software. The Oracle database resides on the VMS-cluster server. It centralises all the information for the description of the system.

FIG 2: Functional Layout of Control System HW and SW

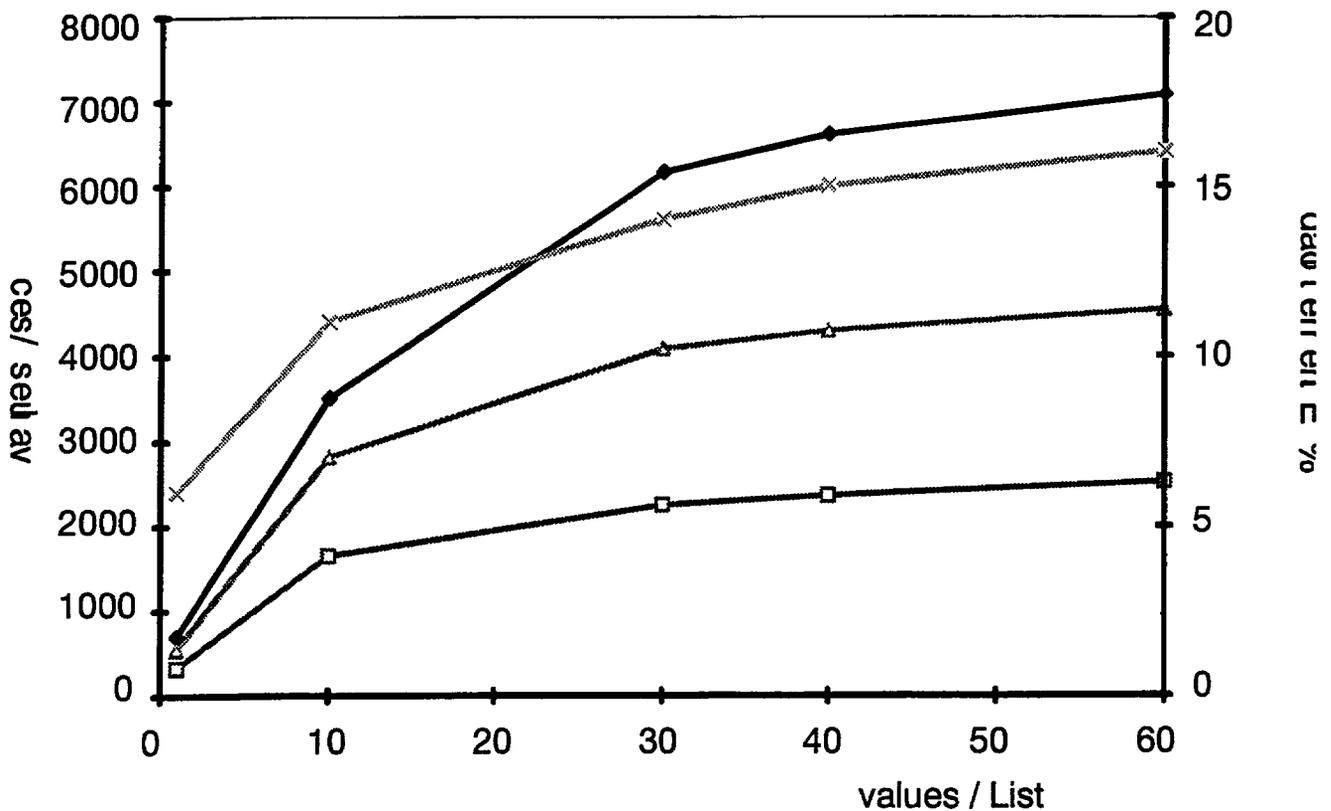


PERFORMANCE AND RESPONSE TIME IN THE SYSTEM

In our system the I/O performance of one front end is one of the dominant factors for the global system I/O bandwidth. Peripherals can only be connected to one front end machine, if this machine is saturated its peripherals have to be distributed over more than one front end.

The maximum available I/O performance for requests from Ethernet was measured. This was done by loading one front end up to saturation using different forms of characteristic requests (cases 1,3 and 4). For efficiency reasons I/O requests are grouped in lists of up to 60 cells, in these lists one device or value is associated with one cell. Measurements were therefore made for 1, 10, 30, 40 and 60 cells per list. The front end computers measured are the HPrt743 operating at 64 MHz.

FIG 3: I/O Performance of one FEC over the Network



- 1) The most efficient mode, used in displays for fast operator feedback (closed diamonds in fig 3). Here lists are dynamically predefined, a repetition rate specifies execution. I/O is performed without additional requests.
- 2) For writing data or for I/O with varying destinations every requested list has to be sent to the frontend. These figures are measured with one VAX 4000/60 acting as requester. The loop time is determined by the sum of execution time in the two machines. (Crosses in fig 3.)
- 3) This is the same case as 2, but with four simultaneous requesters. To measure the maximum performance of the front end it is now loaded up to saturation. (Open triangles in fig 3.)
- 4) Ethernet load corresponding to case 1. To decrease Ethernet load a shortened protocol for reply messages with one third the message length is also implemented. (Open squares in fig 3.)

From the above data the performance of the front end, as seen by the network, can be described approximately as the sum of a fixed overhead time per list and the time needed for the treatment of each cell within this list.

These figures now are:

- Previously defined list, Overhead 1.2 ms and 0.12 ms per cell.
- List defined at each request, Overhead 1.6 ms and 0.20 ms per cell.

The accelerators at PSI are operated in continuous wave mode, apart from the inherent 50 MHz structure there is no other time structure in operation, therefore the timing restraints mainly originate from control loops including magnet power supplies and from fast operator feedback and interactions. Therefore a response time between 40 and 100 ms seems adequate. Timing granularity for timed requests in the frontend is 20 ms.

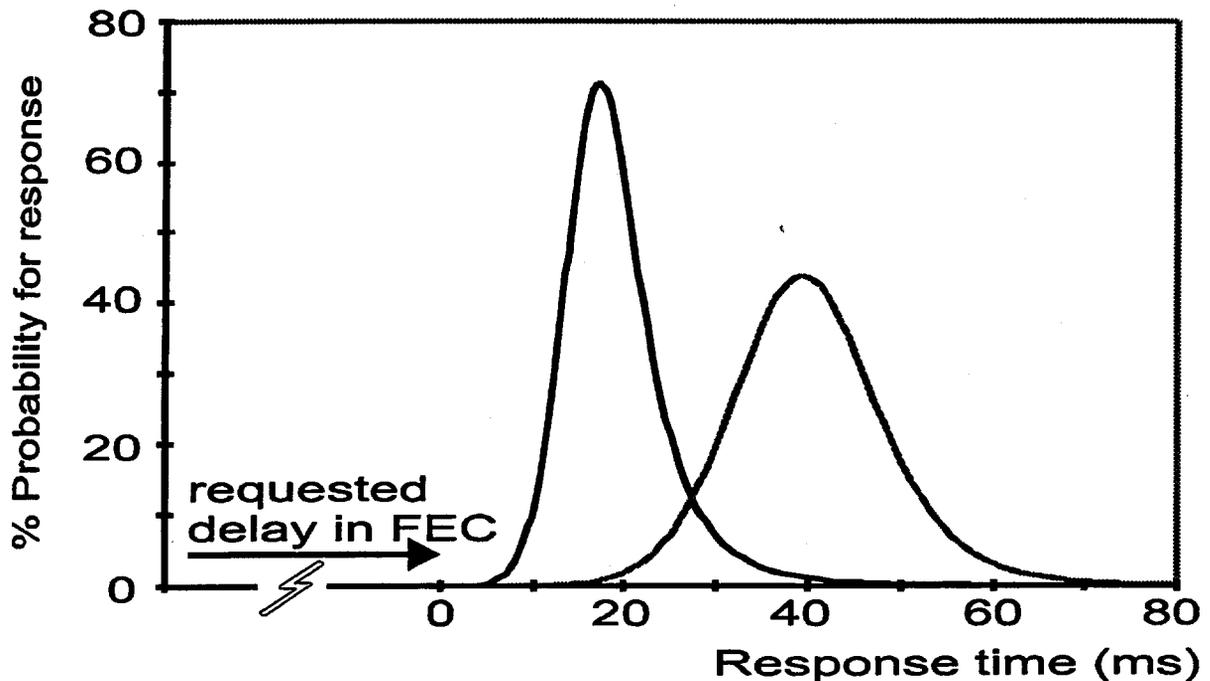
The variation of the turnaround time was measured for two typical MOTIF-applications with repetitive execution in a loop. In both cases the workstation application sends one compound request (write, wait and read) to two FEC's. The timing between write and read is done in the FEC. The loop restarts after reception of the read data and some processing in the workstation. This is a typical case of a control loop for multiply coupled parameters.

In case 1 (fig 4) all time-critical code was executed at AST-level. It showed that 90% of the response time lay within a window of 20 ms and that only 0.1% lay outside 40 ms.

In case 2 most of the time-critical code was executed at user level. It included signalling through X-windows. Here the respective figures are 30 ms for 90% and 50 ms for 0.1%. An additional delay from the increased code can also be noticed.

Both curves show a very small tail towards longer delays. This is caused by the non-deterministic nature of the workstation operating system, the network and the variation in load of the FEC.

FIG 4: Variation of Turnaround Time for repetitive MOTIF-Tasks on an Open VMS-Workstation addressing two FECs simultaneously



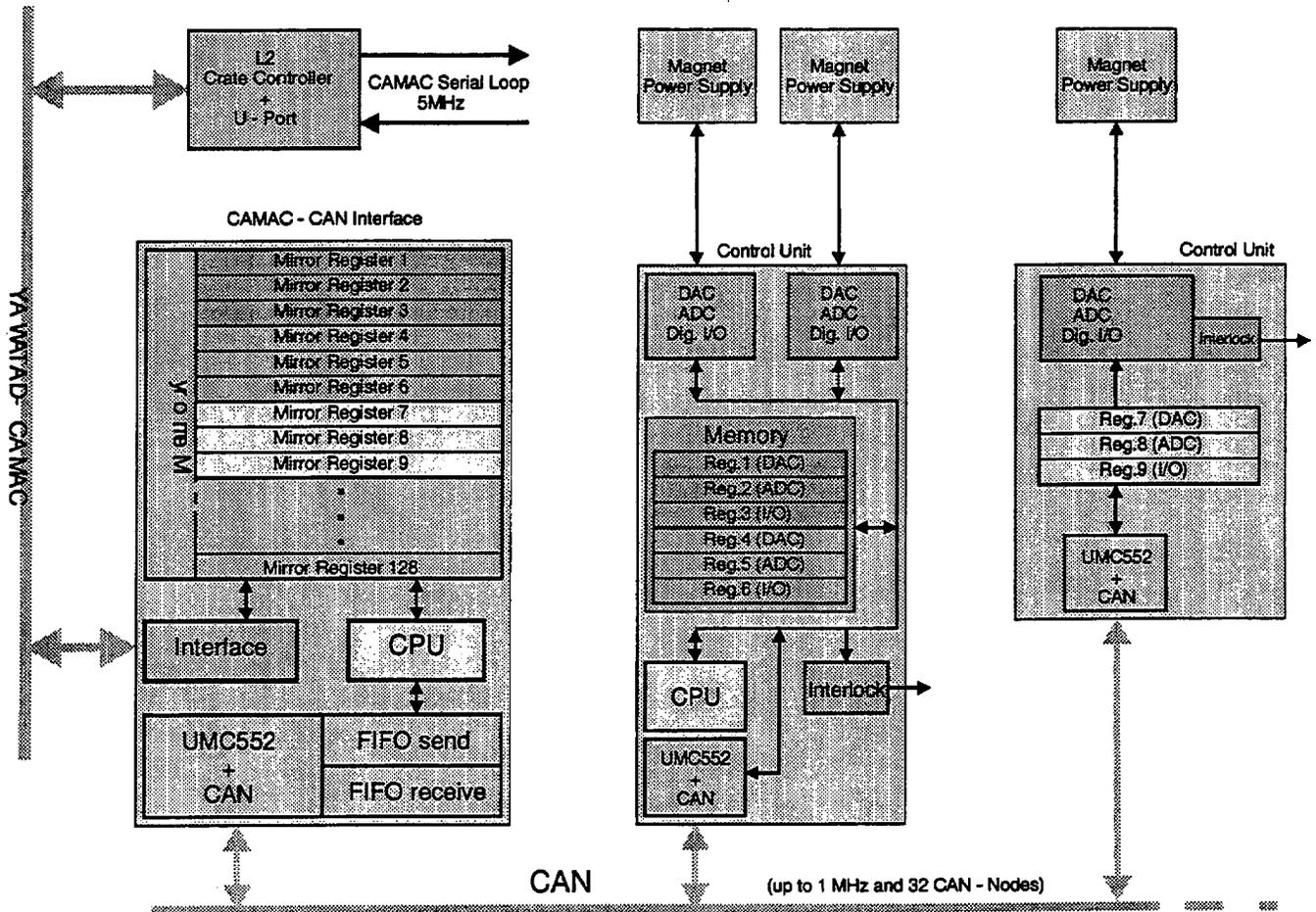
UPGRADE OF THE FIELDBUS

We still use a secondary fieldbus of a design that dates from the origin of the PSI accelerator complex. This parallel bus features 16 bit data, 8 bit function and address space. There is no error check, unless read after write is specified within an application. Galvanic insulation is achieved by transformer coupling of the peripherals. The bus is capable of 50 kHz, this corresponds to 100 kByte/s or 20 μ sec per 16 bit data transfer for random addressing. Since errors are difficult to detect and the cabling is delicate and expensive, we are replacing this fieldbus in order to get increased reliability.

The present prototype for evaluation uses the CAN bus. Other modern fieldbusses using programmable microcontrollers in the remote stations could be used instead. With the CAN bus it is not possible and also not expected to achieve the same bandwidth we have now. The high bandwidth available now is mainly used for filtering. We estimate that this task is anyway better done at the source, prior to any multiplexing mechanism. So for our applications we intend to overcome the handicap of a lower bandwidth by normally pre-treating, usually filtering, the data at the source.

The cascading of two different bus systems is always time critical. In order not to impede on the specific CAMAC timing, a solution with a dual port memory in the CAMAC CAN interface is adopted. The registers of the control units are cyclicly read and then refreshed in the dual port memory. For write functions the data is directly transferred to the control units using a write through mechanism.

FIG. 5: Prototype for Fieldbus Upgrade



ACKNOWLEDGEMENT

We would like to take this opportunity to remember Z. Sostaric for his contribution to our system. He died this summer in a tragic accident.

REFERENCES

- [1] T. Blumer et al., (ICALEPCS '89, Vancouver BC, Canada 1989) Nucl. Instr. and Meth. A 293 (1990) 54
- [2] T. Blumer et al., (ICALEPCS '93, Berlin, Germany 1993) Nucl. Instr. and Meth. A 352 (1994) 150
- [3] "A CAMAC Serial Highway Driver in VME", L. Antonov, V. Dimitrov, W. Heinze, CERN PS/CO/Note 91-22

CONTROL SYSTEM FOR AN INDUSTRIAL ACCELERATOR TECHNOLOGICAL COMPLEX.

V.N.Boriskin*, N.I.Aizatsky, YU.I.Akchurin, V.A.Gurin,
N.V.Demidov, A.N.Dovbnaya, V.A.Kushnir, V.V.Mitrochenko,
A.N.Savchenko, YU.D.Tur, V.L.Uvarov.

National Science Center, Kharkov Institute of Physics&Technology (KFTI), Ukraine

This paper deals with the control and protection system for an electron linac for industrial purposes (KYT). The KYT is a powerful source of accelerated electron beam and is designed to be used for radiation processes including sterilization of medical supplies. KYT was designed and constructed in KFTI. The pilot KYT has been operating since September 1993. This linac produces a 8-10 MeV electron beam with a power of up to 10 kW. The commercial production of KYT was started in 1994. A description of the system is presented.

HARDWARE AND TECHNICAL SUPPORT

The linac includes the high-voltage generator, the electron source, the accelerating system, a klystron with its high-voltage modulator and a scanning and electron beam extraction device. The control system provides the beam current and energy monitoring, the control of system parameters and fault diagnostics of the linac units, the protection for the accelerator system against damage by the beam, interlock protection for the modulator and klystron amplifier against incorrect operation, regulation of the currents in the magnet power supplies, regulation of the phase and power of the RF system, control of the irradiation dose at the target and monitoring of target transportation system.

The layout of the system is shown in figure 1. The hardware consists of a personal computer equipped with analog-to-digital converters (ADC). This is connected by a local area network to the units shown below, which are situated about 40 m from the PC. There is a synchronization unit (SU) and three microprocessor controlled units: one for the klystron amplifier modulator (KAM), one for the technical services (TS), and one for the target transportation system (TT). The control system of the experimental model of KYT was made using the CAMAC standard [1]. However, this was too complicated for industrial usage, so the new version of the system will not use CAMAC.

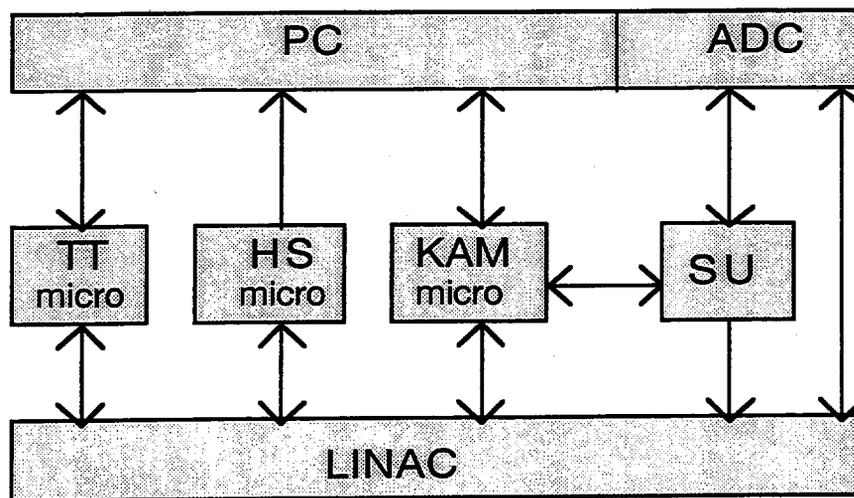


Fig. 1. Schematic of the control system.

The ADC units receive signals from analog pulse detectors. They have two switched channels with 100 ns period and two integrating channels. The SU assigns time-dependent programs to the pulse generation system, the control system subunits and the measuring equipment. The linac operation frequency can be selected within the range of 25 - 300 Hz, while in emergency situations or during a beam commissioning run it can be lowered to 6.25 Hz. Switching on the SU automatically brings back the operation regime which was in use when the SU was previously turned off.

The KAM unit monitors the operation of the klystron modulator, ensuring the interlocking of the HV-supply and beam switch-off, when the accelerator equipment enters into a hazardous state of operation, or when its operation becomes dangerous for service personnel. This unit controls about 60 digital and analog signals.

The HS unit controls the operation of two cooling and three precise temperature controlled water circulating systems, monitors the temperatures in the accelerator main components and the water flows in the cooling systems, using thermocouples and sensors. The TT subunit controls the magnetic elements of the accelerator and the target transportation system.

SOFTWARE AND PROGRAMMING

The program package CSL (Control System Linac 1.2), written in C, provides for the CS operation in three stages:

- linac switch-on and -off,
- parameter variation and control,
- automatic control, identification of any deviations from the assigned operation parameters and monitoring of the irradiation process.

Information on the system operation and the electron beam parameters is fed to monitors on the local control panels at each of the subunits TS, KAM, SU, TT, and to the color graphics three-screen display at the main control console. Control of the accelerator operation can be carried out either by the operator at the main console PC keyboard, or from the local control panels. For each of the 16 main accelerator systems there is a corresponding overlay module in the CSL, called by the basic monitoring software. These program modules allow the operator to monitor single or multiple systems parameters, to give commands, to save and view archive files, to control the radiation dose, to control the transport system for the articles being irradiated, etc. Parameters of several systems can be monitored simultaneously, but only one system can be regulated at a time.

CURRENT STATUS AND PROSPECTS

At the present time different versions of the system described above are working at two accelerator complexes. Two new KYT-2 accelerators which are now being manufactured will be supplied with similar systems.

REFERENCE

1. V.N.Boriskin et al. Control system for a linear resonance accelerator of intense electron beams / Nucl. Instr. and Meth. in Phys. Res A 352 (1994) 61-62.

Controls in the Past Year of ELETTRA Operation

D. Bulfone, L. Barbina, C. Bortolotto, M. Lonza, R. Marizza, P. Michelini, F. Potepan,
F. Radovicic, C. Scafuri
Sincrotrone Trieste, Padriciano 99, 34012 Trieste, Italy

After a successful commissioning phase, ELETTRA, the Italian third-generation synchrotron light source facility in Trieste, is operational. The paper describes the experience gained during the past year with the control system of ELETTRA. Emphasis is given to the operational aspects and to the major system upgrades developed to improve the machine performance.

1. INTRODUCTION

ELETTRA smoothly moved from commissioning to regular operation during 1994 [1] [2]. Since the beginning of 1995 the percentage of beam time dedicated to experiments is about 80 %, which corresponds to a total of 3800 hours in the past year. Figure 1 shows the cumulative operating hours for machine studies and user experiments from the beginning of the commissioning in October 1993. The 1995 efficiency of the machine, which is defined as the ratio between the delivered and the scheduled beam time, exceeds 90 %.

Five insertion devices (ten undulator sections, three wiggler sections) are presently installed. Four beamlines are fully operational and open to external users, another five will be completed by July 1996.

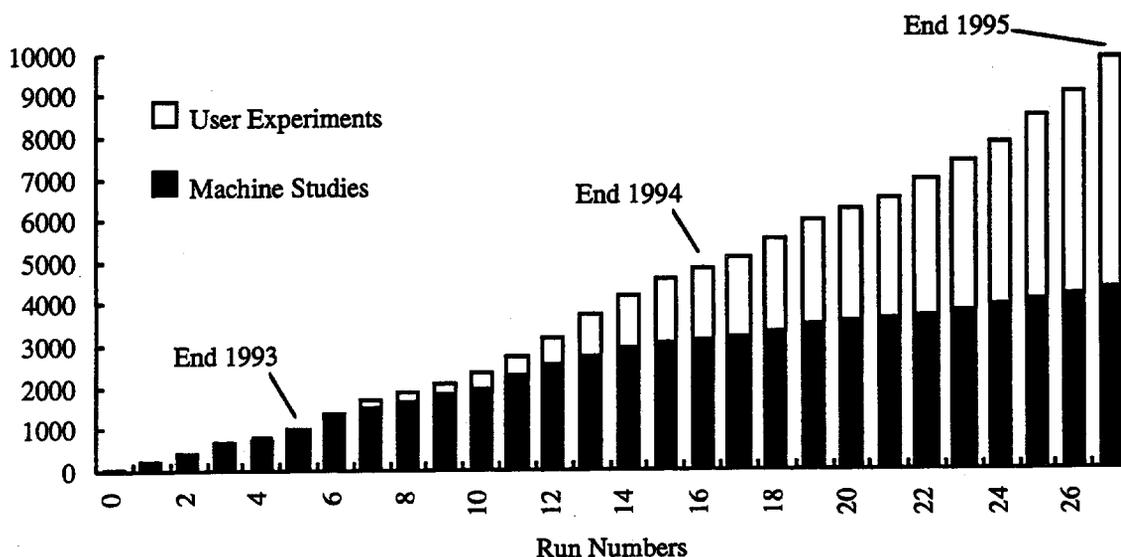


Figure 1. Cumulative operating hours for machine studies and user experiments from the beginning of the ELETTRA commissioning.

2. OPERATIONS

A 1.0 GeV electron beam produced by a linac is transferred via an underground transfer line to the storage ring where the synchrotron light is produced. A ramping mechanism, which will be discussed later, has been installed to provide the higher-energy electron beams which are requested by the users and this allows flexible operation of the facility in a wide energy range.

Two kinds of shift are scheduled. Machine physics studies are performed in the so called "machine shifts" where the ring is operated at different currents, filling patterns, energies and optics. During the "user shifts" the

ring is 80% filled with an initial current of 250 mA at an energy of 2.0 GeV and synchrotron light is delivered to the experiments. Injection is performed once a day.

The linac, which was bought as a turn-key machine, is presently controlled by its proprietary control system from a local console whereas the transfer line and the storage ring are fully controlled by the ELETTRA Control System from the storage ring control room.

Preparing for injection during a user shift, the control room operator has first to dump the stored beam, open the gaps in the insertion devices (ID), close the front-end shutters and valves, cycle the storage ring magnets and load the machine file [3]. After filling the ring with to requested current value, he has to execute a ramp to the final energy, close the ID gaps, correct the orbit to zero position and slope at the centre of the IDs and open the front-end shutters and valves to provide light to the users.

As the machine is more and more dedicated to user experiments, a number of control system improvements have been designed and/or installed in order to optimize the operation of the facility during the user shifts. The following sections describe such developments focusing on automatization and speeding up of machine preparation, ID and linac control.

3. IMPROVEMENTS

3.1 Ramping

The ramping process at ELETTRA consists of synchronously setting the currents of the storage ring magnet power supplies (46 main magnet (bending, quadrupole, sextupole) and 164 corrector power supplies) in order to bring the machine to a higher energy state with arbitrary (usually constant) optics. The definition of the path in the optics space and the specification of the DAC setting values (called "steps") are completely decoupled. This allows a free choice of the parameters for the first whereas the power supply hardware characteristics can be dealt with by the latter. The system is based on a general purpose mechanism which uses MIL-1553B broadcast packets for the synchronization of the software processes executed by the different interface computers (Equipment Interface Units, EIU). The measured jitter of the DAC setting times is kept between 10 and 300 μ s. The ramping process does not affect the control system operation and is transparent to the control room operator.

The earliest implementation and results of ramping at ELETTRA have been presented in [4] [5]. The system also featured the so called "file ramping" which enabled a smooth transition to any machine file. The first successful energy ramping from 1.1 to 2.0 GeV was performed in January 1994, after only one hour of ramping tests. Saturation effects in the combined-function magnets led to variations of the betatron tunes above 1.6 GeV. During simple energy or file ramping the tunes were stabilised with the use of a feedback system.

Electron beam energies from 600 MeV up to 2.31 GeV were easily achieved. The ramping speed was however limited by the tune feedback repetition rate and by the power supplies which were set to follow a linear ramp whose maximum slope was fixed to 1 DAC bit/10 ms step. Ramping from 1.0 to 2.0 GeV during a user shift, including large safety margins, took about 12 minutes.

A further improvement called "multiple-file ramping" which was installed in May 1995 overcomes the limitations above. The ramping is now performed through a set of machine files which are optimized and measured in advance on the machine, taking into account tune shifts, optics and closed-orbit distortion. Such a procedure avoids the use of the tune feedback. The setting of the power supply currents with time is done using a $1 - \cos(\omega t)$ function which gives zero derivative at the beginning and end of the ramp with the maximum in between, therefore minimizing the total time. The ramping speed is changed by varying the duration of each step from a minimum of 1 ms to the desired maximum value. The minimum time achieved for a 1.0 to 2.0 GeV ramp is 100 s. The typical time during user shifts is about 3.5 minutes.

Multiple-file ramping has also proven to be a flexible tool during machine shifts, allowing synchronized transitions between different machine files.

3.2 Insertion Device Loops

A closed-loop system on each ID compensates for the residual closed orbit distortions during gap closure [6]. The correction system for each undulator consists of a set of correction coils which act in the horizontal plane and is powered by one four-channel power supply. Twelve (four per section) permanent magnet rotating blocks moved by stepping motors are used for the wiggler.

The currents (positions) which minimize the orbit distortion are first measured at different intermediate gap values and a look up table is generated. When the gap is changed the software loop sets the currents (positions) accordingly by interpolating the measured values. A tracking scheme based on a predicting algorithm is

implemented for the wiggler. The closed loop takes into account the speed of the motors and the delays caused by the "slow" serial lines which interface the stepping motor and the ID gap controllers to the control system EIU.

3.3 One Button Machine

The progressive knowledge of the behaviour of the machine, together with its increasing operational stability led to the development of application programs that summarize several repetitive and disjointed actions into one single command. A small library of such programs is normally collected at control room level and used by the operators on shift.

The one button machine (1bm) application has been designed taking into account the following issues:

- keep the already developed and well tested programs
- define a logic flow control of the various tasks
- provide extended step-by-step help for each task action
- avoid operator intervention during flow execution
- provide the choice between manual or automatic error recovery in case of task failure
- use a robust Motif interface

The one button machine (1bm) is a simple task-spawner that uses conventional UNIX routines for process execution and communication. Once tasks are identified with letters, 1bm can understand basic Boolean strings such as "AB+Cd" which is interpreted as: "execute the next task if task A and task B were correctly executed, or if task C was executed and task D was in error or was not run at all". At each step, i.e. at each task termination, the control logic is checked: if the condition is true, one or more tasks can start concurrently.

The exit condition is used to validate task termination and to establish a simple error reporting convention. The standard output is redirected towards the spawner for tracking purposes and special characters are used to isolate automatic error recovery strings.

Processes jump over a basic state diagram in which manual intervention may execute, stop, abort or freeze any selected task, without invalidating the control flow dependencies.

If a foreseen task is not implemented, the extended help boxes guide the operator through the necessary steps and wait for a final "Done" acceptance that allows the automatic flow to take control again.

3.4 Linac Control

The linac proprietary control system does not provide the requested level of reliability and the design of a new system based on the ELETTRA Control System has started. Both software and interface hardware have to be replaced. A modular installation is being considered in order to minimize the effect on the facility running schedule. The new system will allow complete control of the linac from the control room workstations where a number of essential analog signals can already be displayed.

As a short term solution, a remote procedure call server has been installed on top of the original linac software. It allows monitoring and control of the main linac parameters from the control room.

3.5 Other Developments

Two different RF signal generators provide the 500 MHz driving signal to the storage ring RF plants and the linac. They can be phase locked or decoupled in order to perform synchronous or asynchronous injection and to independently change the storage ring RF frequency [7]. An automatic procedure searches for the phase value between the linac and the RF plants and optimizes the synchronous injection rate by acting on a phase shifter installed after the RF plant signal generator.

As the storage ring magnets have to be cycled before each injection, the procedure has been speeded up by a new dedicated server at the processing level which uses an S-like function for the setting of the power supply currents with time. The requested time has been reduced from twelve to about three minutes.

A service has been developed to share information and commands between the ELETTRA and the Beamline Control System [8]. It is installed on the ELETTRA general purpose server computer and is accessed by a set of remote procedure calls. The service has a multiprocess architecture with a common data area. For each active client there is a corresponding server process handling the data requests and the data is stored in a memory area shared by all the server processes. Typical information consists of general machine parameters (i.e. beam energy, current and lifetime) and ID and front-end status (i.e. valves, photon shutters and vacuum values). The possibility of sharing commands will allow each beamline user to control the corresponding ID.

General real-time machine status information is available through our World Wide Web server whose URL address is <http://waxa.elettra.trieste.it:8080/ELETTRA.html>.

4. PERFORMANCES AND REMARKS

The ELETTRA Control System has been comprehensively presented in previous publications and details can be found in [9] [10] [11].

The system architecture and the effective distribution of the control resources at each level provide a very high degree of flexibility which allows easy expansion or addition of new schemes to the original configuration.

Hardware and software modularization is fundamental. System changes and upgrades can be transparently installed to face specific needs using state-of-the-art technology. Taking advantage of the modular multi-master architecture, an upgraded Local Process Computer (LPC) equipped with Motorola 68040 microprocessors has been set up in few hours.

Fibre-optic Ethernet and its associated hardware did not give any trouble. The CERN LEP-NC rpc [12] worked very well on both the UNIX and OS-9 platforms. Measured Ethernet loading during machine working time is below 4% and no bottleneck situation has been experienced in two years of operation.

The MIL-1553B hardware confirmed its robustness and the long branches (about 500 m) covering the whole ring can work at 1 Mbit/s without problems. The installation of the OS-9/NET tools (e.g., remote login and distribution of the file system over the network) on top of the basic MIL-1553B communication software gives the system a powerful debugging environment at the lower layers.

The overall behaviour of the VME hardware, which has been almost completely purchased from the market, is good. The special DAC and ADC boards for the control of the magnet power supplies, which have been subcontracted, present outstanding temperature stability. Some of the serial lines which are used for interfacing intelligent controllers presented difficulties in terms of response time and protocol reliability.

The Man Machine Interface (MMI) provides the operator with an intuitive working environment and machine operation requires no special knowledge of the control system details. About forty non-professional operators alternate on shifts and use the control system. The scalable MMI organization leads to a coherent enhancement of the available tools. Higher level applications merging the functionality of different pieces of equipment are located within the existing logical frame and complement the "old" control panels.

The home-made Control Panel Editor (CPE) and its associated library boosted the production of Motif panels and high level software programs [13]. Non-expert programmers can take advantage of the resultant simple C language skeleton which hides the Motif technicalities. Also, a consistent style is also preserved amongst the different applications.

Developing ORACLE applications can be quite time consuming. The Motif implementation of the FORM is not satisfactory and third-party products are being tested to provide a better graphical user interface to the database. A number of improvements, such as integrity checking of the input data and permission control, are implemented with the last release of ORACLE (version 7).

Interlock Programmable Logic Controllers give reliable hardware and software flexibility, but they do not offer a comfortable program development environment. Both the interlock and the alarm system were subcontracted to external companies. They worked satisfactorily from the beginning of the commissioning but gave some problems with maintenance and the acquisition of the associated know-how took some time.

5. CONCLUSIONS

The ELETTRA Control System has effectively provided all the resources needed for the commissioning and operation of the machine. Its operation is reliable and no limitation in performance has been experienced so far. It can be expanded following the "natural" evolution of ELETTRA and permits major system upgrades, for example ramping to be carried out.

The complete installation of the tools aimed at optimizing the machine preparation during user shifts will reduce the time from beam dump to user readiness from 45 to about 20 minutes.

Beside the design and installation of the linac control system, a series of new projects are foreseen for the future. Among these are: a systematic monitoring and archiving of the machine parameters featuring various statistics on equipment operation; a new alarm server with extended capabilities, used as a pilot project in C++; the development of Digital Signal Processing techniques for machine feedbacks; and the control of FERMI [14], an infrared Free Electron Laser.

ACKNOWLEDGEMENTS

I would like to thank C. J. Bocchetta for the detailed statistics on machine operation, our WWW master G. Surace and all the ELETTRA staff who supported the development of the Control System project.

REFERENCES

- [1] A. Wrulich, ELETTRA Status Report, Proc. 4th European Particle Accelerator Conference, World Scientific (1994) p. 57.
- [2] C. J. Bocchetta et al., One and a Half Years of Experience with the Operation of the Synchrotron Light Source ELETTRA, Proc. 1995 US Particle Accelerator Conference, to be published.
- [3] C. Bortolotto and P. Michelini, The ELETTRA Control System Database, in Ref. 1, p. 1770.
- [4] D. Bulfone et al., The Design of the Energy Ramping System for ELETTRA, in Ref. 1, p. 1809.
- [5] R. Nagaoka et al., Energy Ramping in ELETTRA, in Ref. 1, p. 1812.
- [6] R. P. Walker and B. Diviacco, Review Scientific Instruments, vol. 66 (1995) p. 2708.
- [7] A. Fabris et al., Operational Performances and Future Upgrades for the ELETTRA RF System, Proc. 1995 US Particle Accelerator Conference, to be published.
- [8] R. Pugliese and R. Poboni, Optimization of the Synchrotron Beam Alignment Using a Linguistic Control Scheme, these proceedings.
- [9] D. Bulfone, Status and Prospects of the ELETTRA Control System, Nucl. Instr. and Meth. A 352 (1994) p. 63.
- [10] F. Potepan et al., An Integrated Set of UNIX Based System Tools at Control Room Level, Nucl. Instr. and Meth. A 352 (1994) p. 342.
- [11] D. Bulfone and F. Potepan, An Original Approach to Commissioning with the ELETTRA Control System, in Ref. 1, p. 1767.
- [12] K. Kostro, private communication.
- [13] M. Plesko et al., The High Level Software of ELETTRA, in Ref. 1, p. 1776.
- [14] FERMI Conceptual Design Report, Sincrotrone Trieste, April 1995.

The New Controls Infrastructure for the SPS

P. Charrue and M. J. Clayton

ABSTRACT

A completely new control infrastructure has been installed in the SPS machine and experimental areas, replacing the old control system based on NORD computers that dated back to the 1970s. The new system uses Unix workstations and X-terminals to replace the old console computers and PC and VME chassis running LynxOS to replace the low-level interface computers.

This paper will present the old method of equipment access and then describe the transitional phase when the two systems were run in parallel followed by the final complete transition to the new system and the removal of the NORD computers.

A great effort was made to recreate the old programming environment in the new system in order to preserve the enormous investment in application programs. The equipment access and the NORD console simulator are two examples of this effort. Finally the paper will present the results of the first few months of operation of the SPS and its experimental areas with this new control infrastructure.

OLD CONTROL ARCHITECTURE

The SPS Machine

The original control system of the SPS dates from 1976, and has been extensively described elsewhere [1]. Here we will briefly summarise its main characteristics.

It was based on NORD 100 computers connected together in a star network designed specially for the project. In the control room, dedicated computers drove specialist consoles which provided a graphical user interface with trackballs and knobs, while computers installed around the machine were connected to the equipment either by CAMAC or by a fieldbus designed for the project, the Multiplex systems [MPX]. All the applications software was written in an interpretive language, NODAL [2] which had instructions that allowed commands to be executed on any machine on the network. The computers distributed around the machine, to which equipment was connected, contained special functions written in the NORD assembly language which allowed the NODAL programmer easy control of the equipment. It is interesting to note that these functions, called Data Modules, contained their own private data areas and within the limited resources of the time implemented many of the concepts now used in object-oriented languages.

Although many of the computers in the field were connected to equipment belonging to different groups and acted as general service machines, some computers were dedicated to special tasks and contained a lot of special system code. An example of this was the PS computer that controlled the magnet currents in a real time feedback loop.

The general layout of the hardware connections is shown in Figure 1. The NORD computers were connected to the CAMAC crates directly via their I/O bus, the CAMAC crates being controlled by special controllers that appeared as NORD devices on the bus. Although some equipment was controlled by CAMAC modules, most was controlled by MPX modules which were installed in crates attached to the MPX fieldbus, a serial bus with sufficient range to allow control throughout an auxiliary building.

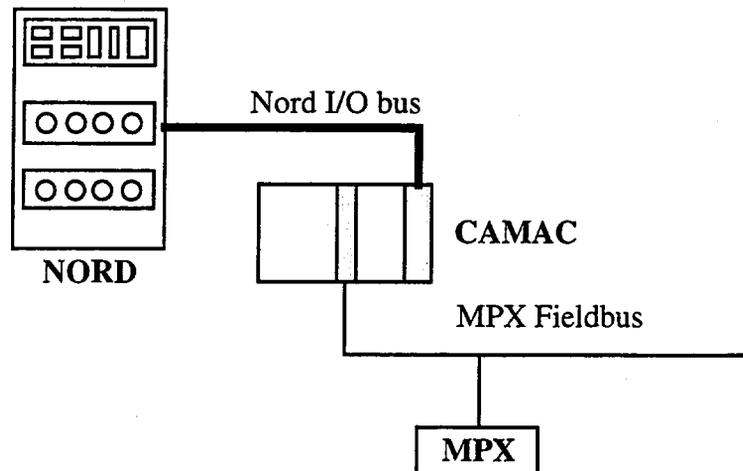


Figure 1

The SPS Experimental Areas

The SPS has two large Experimental Areas, the North and the West, and the Experimental Areas group was responsible for the control of the beamlines between the targets and the experiments. The control system was basically the same as that of the machine: it used the same NORD computers, network, programming languages and concepts. It differed in two ways from the control system of the machine: it used Serial CAMAC as the interface to the equipment and it was multi-user as it provided terminal access for each experiment. Another difference between the Experimental Areas and the Machine had a profound effect on the design and evolution of the control system and that was the fact that the Areas were constantly changing, even during operation, as experiments came and went. The system therefore contained extensive facilities for the easy installation of new equipment even during operation.

The general layout of the hardware connections is shown in Figure 2. The layout was later complicated to allow for the connection of several computers to the Serial CAMAC, but the general principle remained the same.

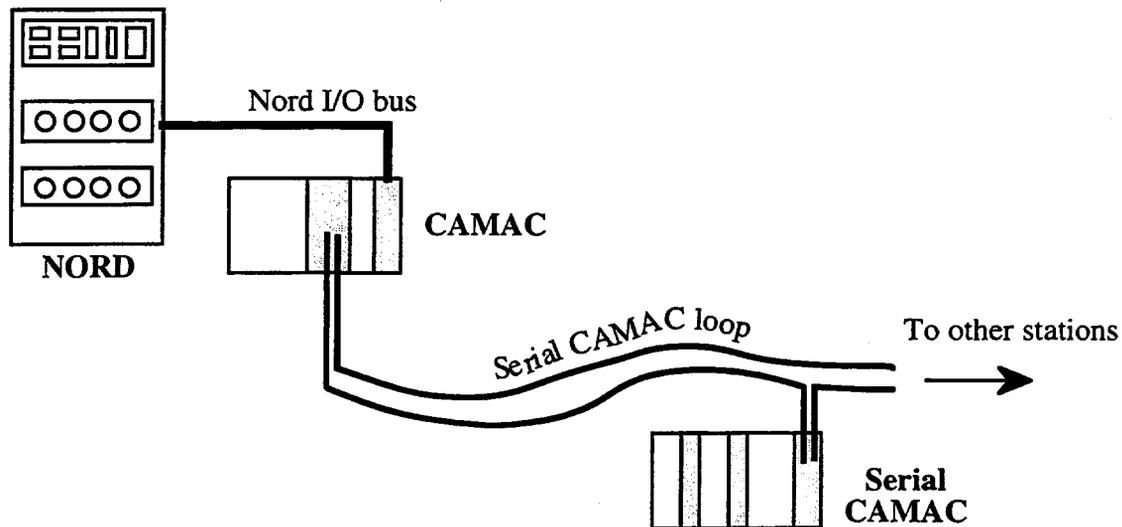


Figure 2

There were several Serial CAMAC loops in each zone, there being three in the largest one, the North Experimental Area. This zone also contained the longest loop which was about 2km long.

NEW CONTROL ARCHITECTURE

Although a great success, the NORD-based control system became old and difficult to maintain, and the restrictions imposed by its limited address space and non-standard operating system became more and more of a problem. The new LEP control system [3] provided a model for the integration of modern operating systems and computers in the accelerator control environment towards which the SPS control system could evolve.

The general outline of the new control architecture is shown in Figure 3. It has three levels:

- **The Control Room Layer.** This consists of X-terminals and UNIX servers which provide the Graphical User Interface to the operators, the computing power to run the application programs and specialist services like file servers, databases and alarm servers.
- **The Front End Computing Layer.** This is made up of front end process computers installed in the field and controlling clusters of local equipment. These front end computers are PCs or VME crates running a real-time UNIX (LynxOS) and tasks are assigned to them on a geographical or functional basis.
- **The Equipment Control Layer.** This layer consists of Equipment Control Assemblies (ECA) and industrial Programmable Logic Controllers (PLC) which are connected to the front end process computers by various fieldbuses or RS232 links.

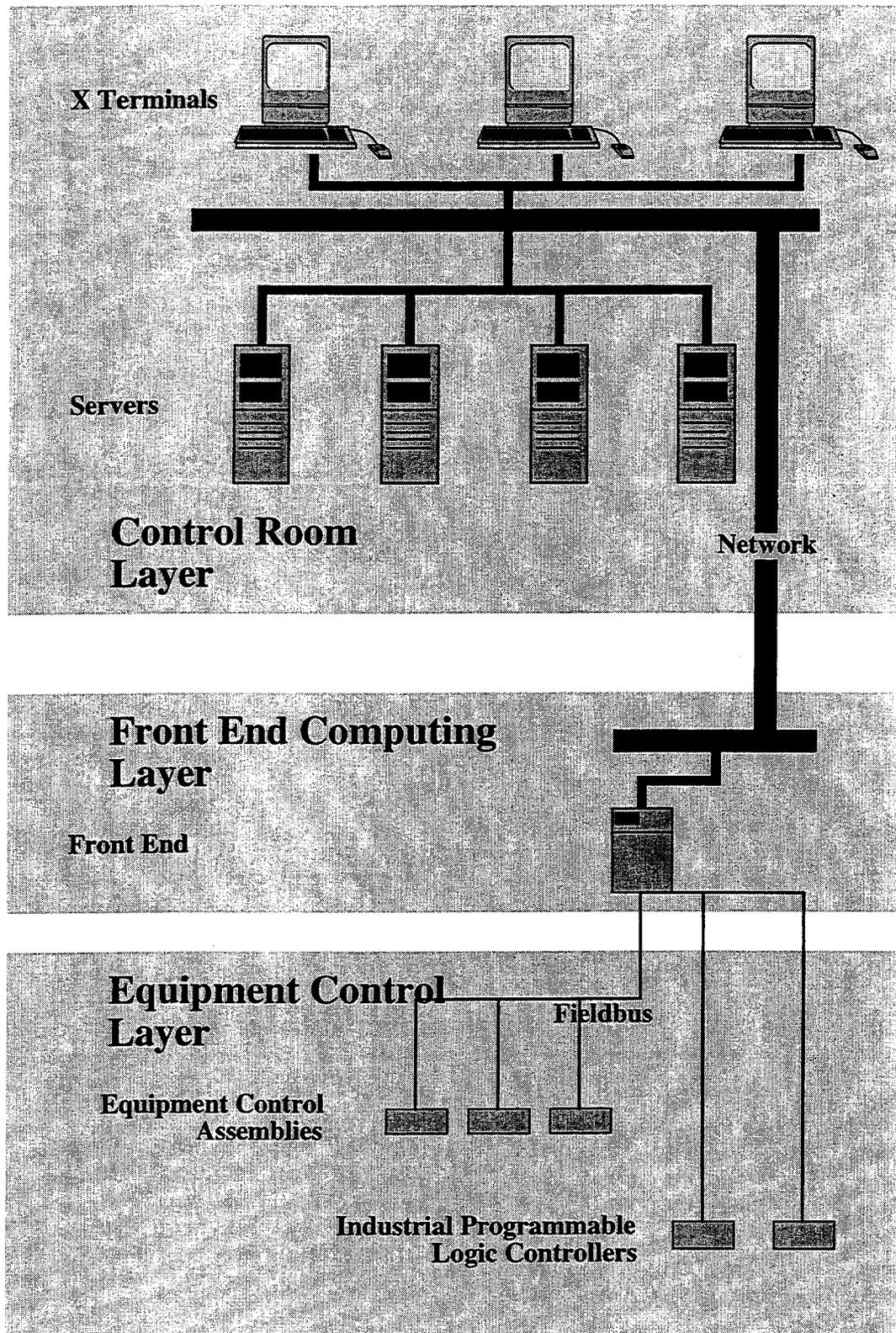


Figure 3

The network between the control room layer and the front end computing layer is a TDM/Token ring backbone. This is connected to Ethernet in the surface buildings and the Experimental Areas via IP routers and to a variety of transmission media in the control rooms via a central intelligent MMAC HUB [5] [6].

The Control Room Layer consists of X terminals, HP-UX Unix servers and some Apollo workstations. The software running at this level communicates with the Front End computers either by RPC calls to specialist servers written for the particular application, by the standard SPS/LEP EQUIP call [7] or by the network communications facilities written into the NODAL language. The SPS/LEP EQUIP call mechanism is the most important standardisation of interprocess communications that has been introduced in the system, so a brief description is given here. An equipment database is maintained in the system, which stores all equipment by name and holds for any piece of equipment both addressing information and the method of access. A standard equipment access library is provided for C applications programmers and NODAL is in this instance another C applications program. The library translates the standard equipment access call into the particular protocol used by the actual equipment, thus removing this level of detail from the application programs. This method of access is used to communicate with the data modules in the Front End computers.

The Front End computers are 386-based PCs running LynxOS. The fieldbus uses MIL STD-1553 which is controlled by cards installed in the PC or by modules installed in the VME crates. The detailed layout is shown in Figure 4 below

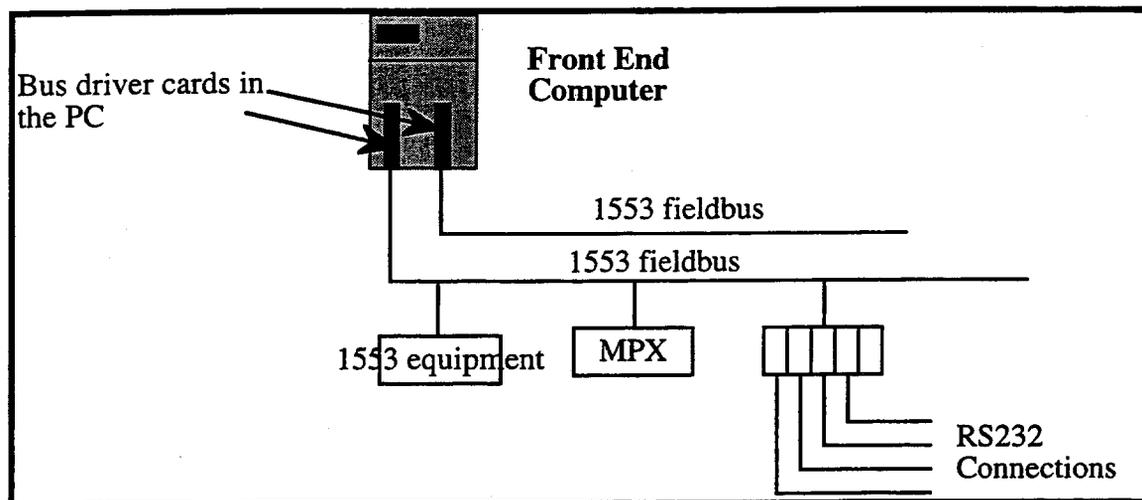


Figure 4

The software in the front end computers consists of

- Drivers for the bus driver cards.
- Translations of the original NORD data modules into C for the new environment.
- A new NODAL interpreter to execute the original NODAL code from the NORD environment.
- A message handler to route EQUIP calls.
- Any specialised equipment servers required by the local equipment.

The data modules were translated from the original NORD assembler code using a strict template which greatly speeded up the process of translation, assured a homogenous functionality of the data modules in the new system and will simplify maintenance in the future.

Some intelligent equipment had been developed for the SPS following the LEP standards, and changes had been made to the old control system to permit access to this type of equipment. Such equipment fitted naturally into the new system as it was immediately accessible via the Equip call mechanism.

The SPS Machine

The old SPS control system had many programs specifically written to take advantage of the particular environment which provided trackballs, knobs and several computer screens. The programs manipulated this environment not only by calling specialist functions but also by sending extensive sequences of control characters to the devices. Many of these programs had been written by specialists long departed and their translation to a modern environment could not be envisaged on a reasonable timescale.

The problem was solved by writing a piece of software that emulated the behaviour of the trackball and knobs and mapped the different screens of the old system into windows on a X terminal. The old NODAL programs could run as before, communicating with the operator via the emulator and with the equipment by NODAL network commands and the new data modules installed in the Front Ends.

The console emulator solved the problem of the old application programs written in NODAL. There were, and still are, a large number of application programs written in C and running on the Apollo workstations. These are the result of a first attempt to rewrite the control system in a more modern environment, but they use the proprietary graphics standards of the Apollo computer and they must be replaced by programs using the X windows graphics standards. They continue to run because they access the hardware either by calls to RPC servers that exist in the new system, or by emulating the NODAL access calls.

The original SPS machine had a very large MPX installation with about 1000 crates and at its peak over 5000 modules. Although this installation was gradually reduced as equipment groups slowly converted their equipment to be directly connected to the 1553 fieldbus (or in some cases directly to the Ethernet), a large installation still remained. This was connected to the new system by allowing MPX crates to be connected directly to the 1553 fieldbus with a new controller module, and software was written to allow access to the equipment in exactly the same way as before.

The SPS Experimental Areas

Although the old Experimental Areas control system had two notable upgrades during its lifetime, the introduction of microprocessor switch crates to share the load between several Nord computers, and the introduction of the faster Nord-100 computers in place of the old Nord-10s, it kept the basic structure of the original 1976 control system and ran with no major changes after 1980. Additions and changes were made to the large pool of applications programs.

The problem of the upgrade of the Experimental Areas was examined by a working group, which proposed the solution [4] which is now implemented. Two major problems dictated the form of the new control system for the zones: the large investment in hardware at the level of the CAMAC crates and below and the large and in some cases little understood suite of applications programs written in NODAL over the preceding 15 years by many engineers and technicians, most of whom had moved to other projects.

The old and the new control systems are shown in Figure 5.

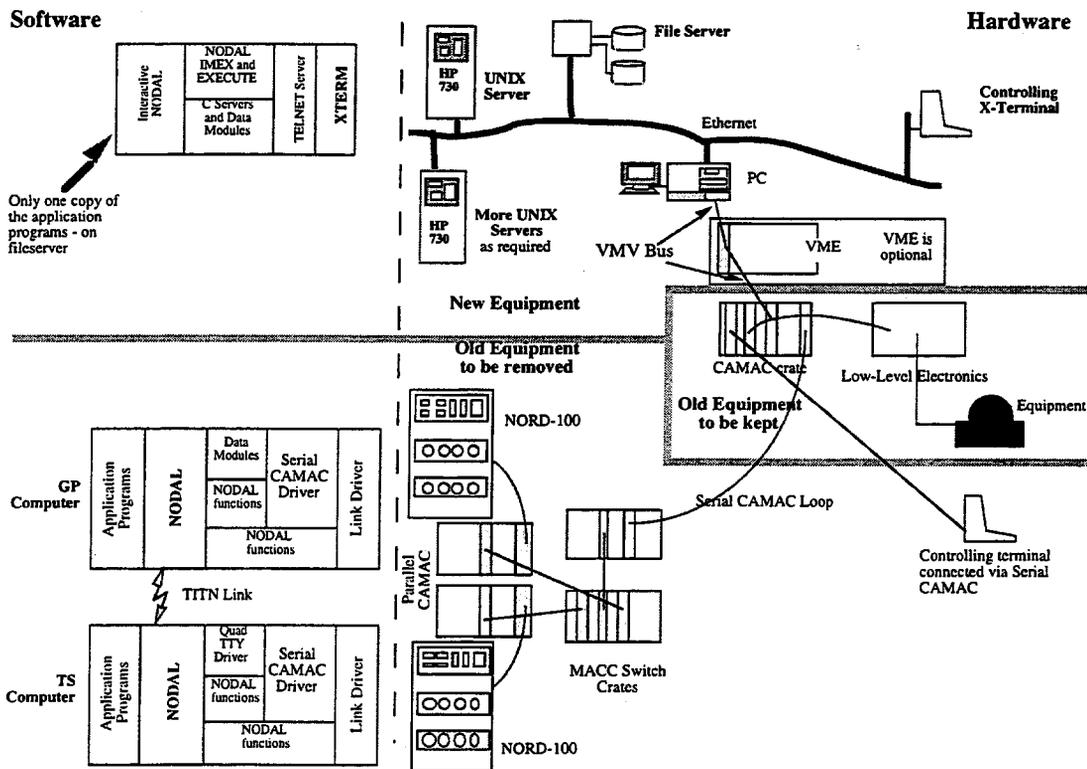


Figure 5

At the bottom of the diagram is the equipment to be removed: the Nord computers and the local parallel CAMAC, the Serial CAMAC loop and the old simple terminals. In the middle of the diagram is the equipment to be kept: the Serial CAMAC crates, the CAMAC modules installed in them and the interface electronics that was connected to these modules. This represents a very large installation as shown by the table below:

Zone	# CAMAC Crates	# NIM Crates	# Patch Panels	# Modules	# Connections
North	28	81	179	1488	8981
West	17	49	108	903	5453
EA Lab	10	29	64	531	3208

It should be noted that the EA Lab mentioned in the table was a third small installation used for test and development. It was the very large investment in material and also in its installation as shown by the large number of connections that made the preservation of this part of the installation unavoidable.

The equipment in the SPS Experimental Areas is installed large experimental halls; the largest, EHN1 is 290 metres by 50 metres, so the equipment is actually installed in small barracks which are inside the halls. This causes the equipment to be concentrated naturally into clusters called stations consisting of one or more CAMAC crates and the dependent lower level electronics. A station controls the equipment geographically close to it.

The upper part of the diagram shows the new installation. It follows the structure of the new architecture with the front-end computers in the zones and HP-UX server machines at the control room layer level. These machines are actually installed in the control room area for ease of maintenance, but they provide a terminal service all over the zones via the Ethernet. The main departure from the software structure of the machine is in the fact that the data modules run in the HP UX machines and communicate with the equipment via small specialist servers in the front end machines. This departure from the normal structure was done because it maintained the overall software structure of the old system, with a single equipment access machine for a single experimental area: an arrangement that was deeply built into the structure of the programs.

The CAMAC crates were connected to the front end PCAs by the VICbus, a parallel bus originally conceived to allow extension of the VME address space to several crates. The CAMAC crates were connected to the bus by commercially-available controllers, and it was hoped that this scheme would allow the easy mixing of CAMAC and VME and lead to a graceful future extension of the system. In practice, it has been found easier to connect VME crates directly to the Ethernet and the VICbus has not turned out to be commercially successful.

THE TRANSITIONAL PERIOD

The transitional period was a critical one, where parallel access to the equipment from the new and the old systems had to be supported as much as possible to avoid the risks and disruption of a "big bang" approach, where an old system is removed and a new system installed and started from scratch. The parallel access was successfully carried out for the machine and the Experimental Areas, although each had to have its own solution.

The SPS Machine

In the case of the machine, the original hardware layout is shown in Figure 1. This was replaced by an arrangement which allowed both the old NORDs and the new front-end machines to be connected to the fieldbus at the same time, as shown in Figure 6.

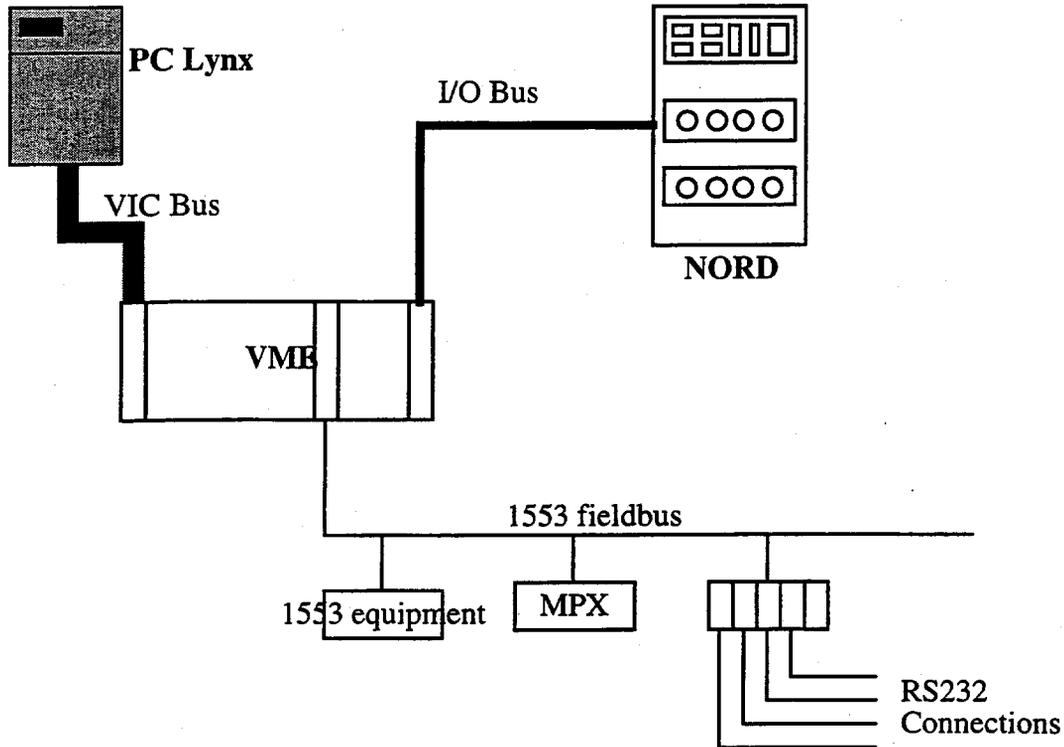


Figure 6

With this arrangement, the translated data modules could be installed and tested in the new Lynx front-end machines while the old data modules continued to operate in the NORDs. When the new data modules became operational, changes were made in the NORD systems to allow them to be called from programs running in the NORDs. As the NORD systems were essentially frozen by this time, this was accomplished by patching the NORD systems so that the equipment appeared to be intelligent LEP-type equipment, for which drivers already existed in the system. The calls to this equipment were intercepted in the VME crate and routed to the new data module in the front end Lynx machine which performed the required action.

A lot of time and effort was required to tune the console emulator so that the old programs could run. This was particularly true in the case of the programs for the SPS machine as the users had over the years exploited many undocumented features of the old system and a decision had to be made each time as to whether to change the program or whether to change the emulator. Usually the decision that required the least work was taken, so that facilities that were used in many programs were reproduced in the emulator, but in some cases changes to the programs were made.

The SPS Experimental Areas

The same principle of graceful transition was applied in the Experimental Areas, but the different hardware arrangements dictated a different solution. The VICbus controller could be configured to act as an auxiliary crate controller and it was used in this position in the transitional stage as shown in Figure 7 below.

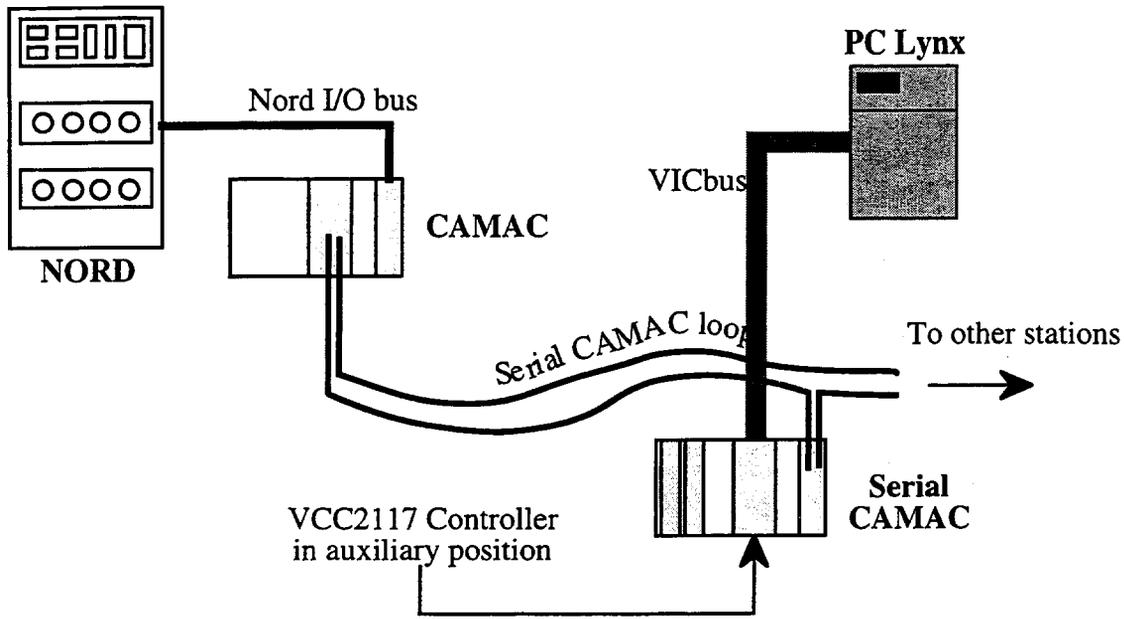


Figure 7

With this arrangement it was possible to test the new data modules while the old system continued to run. It was hoped to run the two systems completely in parallel to allow a painless transition between the old and the new systems, but this was prevented by the problem of the LAMs, the CAMAC interrupts. The CAMAC interrupt mechanism had been fully exploited in the old system, both for the connection of terminals to the system and for the operation of slow devices where the old system could not poll for the response. This mechanism was not used in the new system as the dedicated front end computers could poll without affecting the performance of the machines in the control room layer. However, when both systems were operational the interrupt was taken by whichever one saw it first, usually the NORD as it was not polling. This meant that when the old system was operating, the new system would work only partially work. It could be made to work fully by stopping the old system or by disabling the interrupts in the Serial CAMAC crate controllers and it was the latter course of action that was usually followed. This disabling could be done very quickly by a program and just as quickly reversed. It was thus easy to run test sessions during machine development periods.

There were some problems with the NODAL interpreter, but fewer than those experienced with the Console Emulator as the programs were written to run on simple alphanumeric terminals and did not exploit the idiosyncrasies of the old system as much as the machine programs did.

OPERATIONAL EXPERIENCE

The new control system was put into operation for the 1994 start up of the machine, and for the 1995 start up of the Experimental Areas.

The main problem experienced during the machine start up was associated with access to the data from the target secondary emission monitors. These monitors were intelligent LEP-type equipment, and worked well during tests, but became very difficult to access during actual operation, causing programs to block while waiting for data, and finally crashing the front-end computer. The problems were traced to time-out problems in the equipment, which was unable to respond, or responded with corrupt data during a brief time after an acquisition. This problem was finally solved by queuing the requests for this equipment in the message handler and imposing a suitable delay between requests. This problem shows that no matter how extensively the equipment is tested beforehand, it is impossible to foresee the access patterns that will occur during actual operation.

The Experimental Areas, with the benefit of the experience of the commissioning of the machine, started with no real problems and beam was available for the experiments several days ahead of schedule. The remaining problems are mostly concerned with very long term stability: the correction of memory leaks and the slow accumulation of useless processes.

ACKNOWLEDGEMENTS

The authors would like to thank the SPS Operations group, the Physicists in the Experimental Areas group, the operators in the Power Converter group and all their other colleagues whose hard work in testing their programs, and in exercising the new system in both test and operational conditions was invaluable to the success of the new controls infrastructure.

REFERENCES

- [1] **The Design of the Control System for the SPS**, M. C. Crowley-Milling, CERN 75-20
- [2] **The NODAL System for the SPS**, M. C. Crowley-Milling and G. C. Shering, CERN 78-07 (1978).
- [3] **The LEP Control System**, P. G. Innocenti, Nuclear Instruments and Methods in Physics Research, ICALEPS Vancouver 1989.
- [4] **Report of the Working Group on the Upgrading of the SPS Experimental Area Controls**, H. Atherton, G. Baribaud, O. Berrig, P. Charrue, H-P Christiansen, M. J. Clayton (Chairman), W. Heinze, K. D. Lohmann, D Thomas, SL/CO/Note/91-27
- [5] **The SPS and LEP Control Network Architecture**, P. Lienard et al, Nuclear Instruments and Methods in Physics Research, ICALEPS Vancouver 1989.
- [6] **Evolution of the SPS and LEP Communication Control Network for the SPS and LEP Accelerators**, P. Lienard, Nuclear Instruments and Methods in Physics Research, ICALEPS Berlin 1994.
- [7] **The equipment access software for a distributed UNIX-based accelerator control system**, Nikolai Trofimov, Serguei Zelepoukine, Eugeny Zarkov, Pierre Charrue, Claire Gareyte, Hervé Poirier, Nuclear Instruments and Methods in Physics Research, ICALEPS Berlin 1994.

A Versatile Modular Radiation Monitoring System

Don Dale, Daryl Bishop, Tyler Ewert, Dan Harrison, Jack Lam, Michael LeRoss
TRIUMF, 4004 Wesbrook Mall, Vancouver, B.C., Canada V6T 2A3

ABSTRACT

A modular microprocessor-based Radiation Monitoring System has been developed for the detection of neutron, gamma and beta fields. The system consists of distributed local monitoring stations located close to the radiation detectors. A local monitoring station houses up to two Universal Detector Modules and a power supply in a standard Eurocard cage. The Universal Detector Module supports Geiger-Mueller tubes and photomultiplier detectors. The module provides local display of averaged radiation fields and high radiation level trip contacts. A local station can operate stand-alone or communicate with a supervisory processor on an RS-422 highway. Experience with an installed system is described.

1 INTRODUCTION

TRIUMF and Ebco Technologies Inc. of Vancouver entered a Technology Transfer Agreement to design and build small cyclotrons for medical isotope production. A TR30/15 model, which can deliver 500 μ a of protons of 30MeV or deuterons of 15MeV energy was installed at the Institute of Nuclear Energy Research (INER) near Taipei, Taiwan. The operation of the cyclotron required a radiation monitoring system, which is an integral part of the safety procedures around particle accelerators. Monitoring equipment used at TRIUMF consists of radiation detectors, a High Voltage (HV) power supply and pulse discriminators in NIM electronics connected to CAMAC scalars that are read by a computer. Detectors typically consist of gamma monitors (Geiger Mueller (GM) tubes), Neutron monitors (BF₃ tubes) and stack air monitors (an air sampling system with a scintillation/photomultiplier tube). The installation at INER required twenty detectors at ten different locations within the cyclotron complex. The NIM/CAMAC based solution used at TRIUMF was considered too expensive within the INER infrastructure. A new design with a distributed modular approach was chosen to provide a more viable solution.

The Radiation Monitoring System was designed to monitor areas normally accessible by staff and provide radiation level information that forms part of a safety system. This system was not intended to replace radiation surveys undertaken by a trained radiation surveyor.

2 DESIGN GOALS

Our goal was to design a flexible Radiation Monitoring System (RMS) that could be accessed remotely by a host computer, yet could also function stand-alone. The system had to provide the necessary high voltages and accept three types of detectors: Geiger-Mueller tubes (GM), BF₃ tubes and photomultiplier tubes. To function as a stand-alone instrument, a front panel switch and display were needed. Other essential requirements were that calibration parameters could be modified and modules replaced with no manual reconfiguration. The system had to also convert pulse counts to standard units of radiation dose-rate-equivalent (DRE). An interlock output had to be provided when radiation levels exceed predetermined setpoints.

3 SYSTEM DESIGN

The radiation monitoring system designed and built for the TR30/15 is distributed and modular. It consists of local monitoring stations close to each detector or detector group. A local station consists of several microprocessor-based Universal Detector Modules (UDM) housed in a Eurocard cage. Each detector module is auto-configured from the backplane according to its location in the crate. The module contains no adjustable devices. A local station can operate stand-alone or in communication with a supervisory processor on an RS-422 highway.

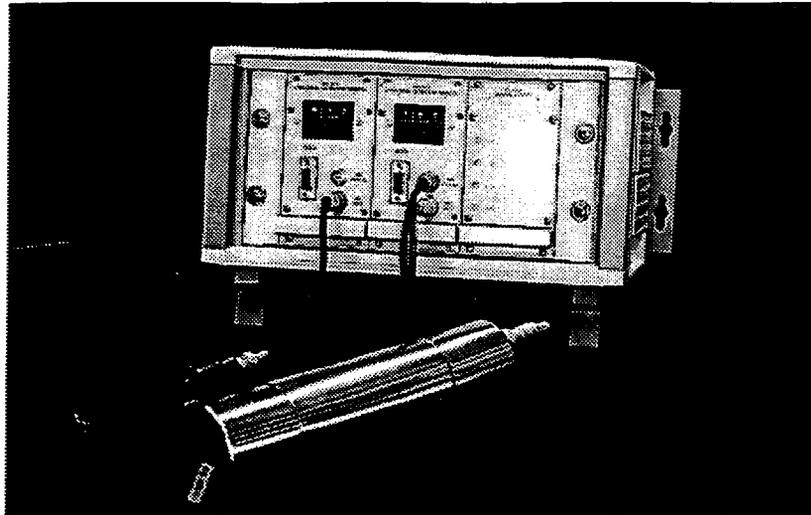


Figure 1. A station crate with two UDMs connected to a Geiger-Mueller tube and a photomultiplier tube

3.1 HARDWARE

Full-width (six slots) or half-width (three slots) station crates are possible. The logic power supply resides in the farthest right-hand slot and the remaining slots may be filled by UDMs. An 8-bit address is set during module configuration.

Station Crate

A Station Crate consists of a standard Eurocard card cage. The card cage is 4U high and has either the standard 19" width of 84HP, or a half width of 42HP. Configuration information is stored in an EEPROM located on the crate backplane. If a defective module is replaced, the new module will configure itself for the same high voltage, calibration constant and other parameters in use by the original module. This reduces the spare parts required.

Logic Power Supply

The logic power supply consists of a commercial unit built into a 3U by 14HP module and provides +5, +/-12 and +24 VDC. The power supply may be powered by either 90-132 or 175-264 VAC at 47-63Hz.

Universal Detector Module

The module is 3U high and 14HP wide. It will accept either a photomultiplier tube, a GM tube or a neutron detector. The UDM uses a Signetics 80C552 microcontroller as CPU.

The UDM can operate in stand-alone or supervised mode. A block diagram is shown in figure 2.

High Voltage Power Supply

The high voltage power supply is a switching one and converts +24VDC from the backplane to a maximum of 2500V. The three nominal voltage ranges to be used are: 1850V for the neutron detectors, 1200V for the photomultiplier tube and 900V for the gamma detectors. These nominal voltages are the default values. The manufacturer's specified tube voltage is entered during module configuration. The microcontroller sets the high voltage with a pulse width modulated output converted to a DC voltage and monitors it through a resistor divider fed into a 10 bit ADC. The voltage can be set to one part in 256 for a range of 0 to 2500 volts.

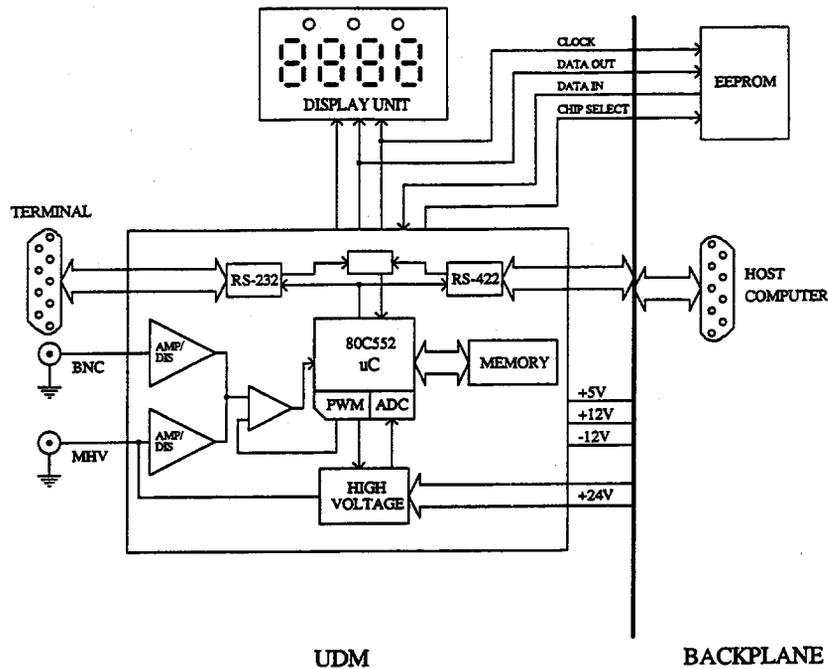


Figure 2. Block diagram of the UDM

Amplifier Section

Two independent amplifier and discriminator sections have been implemented in the UDM, one for the gamma, residual and neutron detectors and the other for the photomultiplier detector. The two amplifier stages are necessary because of the difference in pulse amplitude between the GM tubes and the photomultiplier tube. The gain of the amplifiers is fixed. The threshold for the discriminator section is set by the microcontroller to one part in 256. The microcontroller selects the amplifier stage according to the configuration information.

Pulse Counting

After the pulses have been amplified and discriminated they are fed directly into the microcontroller's count register. This accumulated count is sampled every second. An average is taken of the last four samples (thirty for the neutron detector) then the DREt value is calculated and displayed.

Front Panel

A display on the UDM front panel provides anyone in the area with an indication of the radiation DRE in both supervised and stand-alone operation. All indicators are designed to be readable from a distance of two metres.

The front panel (see Fig. 1) has:

- a four-digit LED display to show the radiation DRE and HV status
- three individual LED lamps to indicate the detector configuration, green for the gamma detector (GM tube), red for the stack air (photomultiplier tube) and yellow for the neutron detector
- one MHV connector for high voltage; for the GM and neutron tubes this connector is also used for the pulse input
- one BNC connector for the pulse from the photomultiplier tube
- one subminiature "D" nine pin connector for a terminal connection (RS-232)
- a momentary switch to enable/disable the high voltage during stand-alone operation

The four digit LED display shows:

- "OFF" if the high voltage is disabled
- the radiation DRE in microsieverts/hr for the gamma and neutron detectors
- the radiation flux in counts/minute for the stack air monitors
- "con" when a terminal is plugged in for configuration of the UDM
- "rrr" when the module is acquiring data, but no valid average has been established

- "FAIL" if the module detects a difference of greater than 100 volts between HV setpoint and HV readback
- "___" if the module detects a bad configuration on power up.
- Flashing "9999" when the display is over ranged.

Communications

The UDM communicates with the host computer's serial port via an RS-422 link at 19,200 baud. This link consists of two twisted pairs, which allow bidirectional, full duplex data communications.

Power Up and AC Fail Recovery

On AC power up the module will start automatically. It first reads its configuration data from the non-volatile memory on the backplane and then starts depending on the module configuration. There are three options for enabling the high voltage after a power up: a) immediately, b) after a host computer "on" command or c) by the front panel switch in stand-alone mode or as a backup if the host computer fails.

Protection from ElectroMagnetic Interference (EMI)

We have taken several steps to prevent and/or to recover from EMI. The UDM is enclosed in an all metal housing. An EMI resistant spray coating was applied to the inside of the plastic rack enclosure and a hardware watchdog timer resets the microcontroller if necessary. Communication with the host computer uses a differential RS-422 shielded cable. In addition the communication protocol includes a parity bit for each character and a checksum per message. All accesses to the backplane EEPROM are verified using a checksum.

3.2 SOFTWARE/OPERATION

The UDM firmware is written in C. The application program consists of a main loop which distinguishes between normal operation mode and configuration mode.

In normal operation mode, the main loop is interrupted every 50 milliseconds by the microcontroller's on-board timer. Every twentieth interrupt (one second) the microcontroller's pulse accumulator is read. This value is placed into a ring buffer of 30 elements for a neutron reading and four for a gamma or photomultiplier reading. The main loop averages the complete buffer to obtain a counts/second value. This value is then passed to the appropriate conversion routine to calculate the radiation DRE in microsieverts/hour using the parameters entered during configuration. The result is output to the display and sent to the host computer when polled. The host computer's poll generates a serial port interrupt, and the microcontroller responds with the current data during the interrupt routine.

Configuration mode is entered when a terminal is connected to the serial port on the front panel of the UDM and the main loop detects Data Terminal Ready. In configuration mode the HV is switched off and the operational parameters may be modified. Preprogrammed into the UDM is a menu system to guide the user through the configuration process. The user will first select the module address and the detector type then enter individual parameters or select the defaults for the detector. Interlock output levels may also be entered. The configuration is saved to the EEPROM on the backplane. When the terminal is disconnected, the microcontroller resets the UDM and returns to normal operation mode

If the UDM is supervised by a host computer, the host is the master using a command-response protocol. Messages are ASCII strings. The detailed message formats are as specified in fig 3.

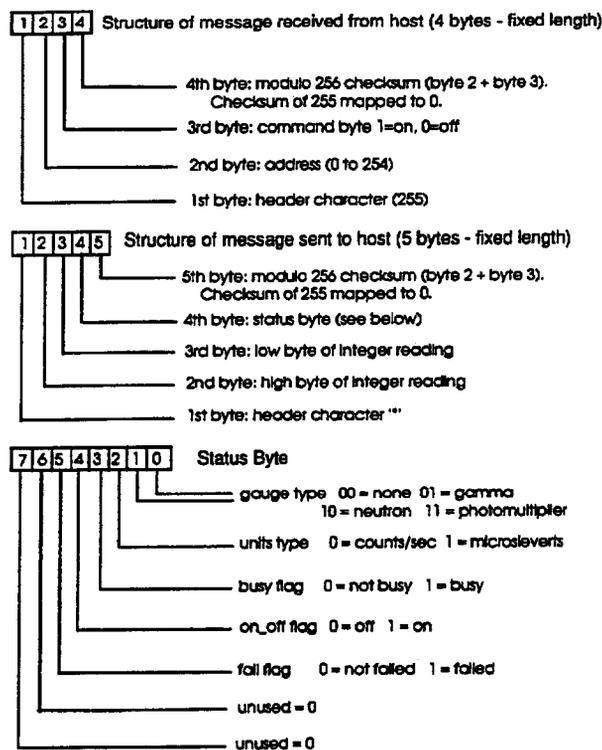


Figure 3. Communication protocol

4 SUMMARY

A versatile radiation monitoring system consisting of a Universal Detector Module and Power Supply housed in a standard Eurocard case has been developed. A system utilizing six GM tubes for residual field measurement inside the vaults, six GM tubes for hallway measurement, six neutron detectors for hallway measurement and two photomultiplier tubes for stack air measurements is currently in operation on the EBCO Technologies Inc. TR30/15 cyclotron at the INER in Taiwan. The ten stations communicate with the cyclotron safety system PLC (Allen-Bradley PLC5/40L). A similar setup was installed for the TR13 Cyclotron at the National University hospital in Seoul, Korea.

Experience at INER suggests two possible improvements to the system: a preamplifier for the neutron detectors, to allow the detector to be moved farther from the module and a higher powered HV supply. The system at INER has been running without interruption since early 1993.

5 ACKNOWLEDGMENTS

The authors thank Richard Cline of EBCO Technologies for his assistance. We would also like to thank the TRIUMF Safety Group for their assistance, especially Tom Moskven and John Drozdoff.

6 REFERENCES

- [1] Glenn F. Knoll, *Radiation Detection and Measurement*, John Wiley & Sons, Inc. 1979
- [2] C Development Documentation, Archimedes Software Inc., 2159 Union Street, San Francisco, CA
- [3] 80C51-Based 8-Bit Microcontrollers Data Handbook, Philips Semiconductors-Signetics Company, 811 East Arques Avenue, Sunnyvale, CA

Implementation of PCs in the HERA Control System

P. Duval (DESY-MKI, 22607 Hamburg FRG; duval@pktr.desy.de)

Abstract

Progress in controlling the HERA machine at DESY with PCs is described, with emphasis on the use of producer-consumer communication links. To date the quench protection system for the superconducting proton ring, the proton collimators, and major portions of the beam diagnostic control and RF controls are incorporated in a control system using PCs running MS-DOS and WINDOWS-based software in a NOVELL network environment. Although dominated by PCs communicating with IPX protocols, the system also supports IP-based communications for links to UNIX machines and VME CPUs. HERA Magnet control is still performed by NORD minicomputers, but the coming upgrade is also discussed in the context of the PC magnet control system planned for the HERA pre-accelerators and peripheral machines, which is now being tested in the PETRA and DORIS rings.

Introduction

The HERA (Hadron-Elektron Ring Anlage) accelerator at DESY consists of a 6.3 km electron storage ring (30 GeV) straddling a 6.3 km superconducting proton storage ring (820 GeV). Prior to injection into HERA, electrons and protons undergo similar sequences of acceleration starting with LINACs, passing on to small synchrotrons, and the PETRA ring. The beams are then accelerated in the HERA rings, stored at full energy, and brought into collision. The design luminosity of $1.6E31$ requires 210 bunches in each ring. Collisions are observed by the H1 and ZEUS experiments, each consisting of a large superconducting solenoid surrounding the beam pipe at one of the interaction regions, and associated detectors. The electron beam can also be polarized and used to study nucleon spin structure, while the protons diffusing out of the beam are used to produce (among other things) B mesons to study CP violation.

Since the smaller accelerators and the experiments have their own control systems, we can speak of the HERA control system as referring only to the HERA machine. However, we should keep in mind that each system needs vital information from the others, so there has to be a mechanism for fast data exchange. Of necessity, each sub-system originally followed its own frenzied development during commissioning, and each has its own view of the "Standard Model," so establishing such a data-exchange mechanism is in some cases no small task.

In this report we shall focus primarily on the HERA machine, but nonetheless present a few details concerning data-exchange among the sub-systems.

The HERA Machine

Before we begin discussing the relative merits of PCs versus something else, let us first take a quick look at what we have to control. Beam storage and steering is achieved using magnets controlled by approximately 1200 independently powered circuits in the two rings. Setting and reading these magnet currents is the primary task of the control system. Acceleration is achieved via coordinated ramping of the RF system frequency and voltage (consisting of 6 cavities and transmitters for the proton ring and over 80 for the electron ring) and the magnet currents. Backgrounds are controlled by scraping off the beam halos with 3 proton and 12 electron collimators. There are independent proton and electron vacuum controls. As the proton magnets are superconducting, there is a separate cryogenics control for these elements, and an associated quench protection system. Monitoring the beam is of prime importance and to this end there are ~300 proton and ~300 electron Beam Position Monitors (BPMs), and approximately the same number of proton and electron Beam Loss Monitors (BLMs) and associated Alarm modules. As the proton BPMs have an external trigger, there is also a separate HERA Integrated Timing system with corresponding trigger and delay modules for each BPM. There are several integrating beam current monitors for both electrons and protons, as well as fast single and multi-bunch current monitors. Likewise, there are several beam profile monitors, including wire scanners and residual gas monitors. The proton and electron tunes are also monitored, as are various state parameters of the machine in general, such as tunnel temperature.

From the preceding paragraph, a dichotomy of controllable objects is clearly seen, namely those objects related to diagnostics and those which actually control the beam. In the case of diagnostics, under most circumstances devices of a given type, say BPMs, can be read, controlled and coordinated from a single CPU for the entire ring. Furthermore, if such a CPU needs to be rebooted (for whatever reason), it does not affect the machine to any great extent. This is not necessarily true for a machine control object such as a magnet. It might not be possible to control all such objects from a single CPU, necessitating distributed control in some form or another, and if one of the controlling CPUs needs to be rebooted it could easily lead to loss of control and in the worst case a beam abort.

PCs in HERA

1. What is a PC?

A relevant question these days is: When I say PC what do I mean? Do I insist on a hardware definition centered around the INTEL CPU and inexpensively priced end-units, or do I stick with an operating system definition centered around Microsoft, or both? Although, when people run LINUX on a PENTIUM they still refer to the machine as a PC, and when the same people run NT on an ALPHA they don't, in this report we'll generally abide by the third (i.e. both) definition, and go a step further and state that a HERA PC is an INTEL machine which runs either MS-DOS or MS-WINDOWS 3.1.

2. Why PCs?

Looking at hardware, arguments concerning costs are compelling. Granted, an extra high-end PC may come in around the same price as a run-of-the-mill SPARC, but one seldom needs a high-end PC. In fact, since old "forgotten PCs" can frequently do the controls job we're interested in, hardware costs are sometimes negligible. Keeping in mind the age-old adage: "If we just want to go shopping, do we need to drive a Mercedes?" we should always ask ourselves if we are just going shopping.

Looking at software, it's not hard to beat MS-DOS and/or MS-WINDOWS as an operating system. The major shortcoming here is that the OS is not protected, and risks exposure to every vicious C program that tries to run on it. One should reiterate though, that neither MS-DOS nor MS-WINDOWS crashes by itself, and healthy programs will run on it indefinitely (although development can be agonizing). Anyway, that's the down side. The up side is that Microsoft has around 80 percent of the software market and is in no danger of disappearing anytime soon. There is an enormous culture base built around MS-WINDOWS, and an impressive array of commercial software which already covers most of our needs. The overwhelming market share and user-feedback that Microsoft enjoys has essentially defined what is user friendly and what isn't.

3. When and Where to use PCs:

In truth there might be some times when we really do need a Mercedes. It's therefore important to 1) plan for an integrated control system and 2) use the right tool for the right job. So where is a PC the right tool for the right job?

There is generally no disagreement that a PC can be effectively used at the CONSOLE side of the control system (there might be some grumbling, but there is no major disagreement). This is true since the CONSOLEs are only displaying data and passing commands to the Front Ends. They are not directly steering control system hardware. Since real-time is never an issue here, MS-WINDOWS is fine. Furthermore if a rogue program gets loose on the CONSOLE and MS-WINDOWS crashes, you lose the display, but nothing more serious than that.

When can PCs be used at the Front End? This is a more contentious issue. Even the most ardent VxWorks enthusiasts will nonetheless concede that there are simple control tasks, which don't need Real Time by a long shot, and can be quite trivially controlled by a PC. For example, reading an ADC, which monitors the ambient tunnel temperature every few minutes. Such a task can be realized by a 286 PC running MS-DOS (cost: ~100 \$). If the machine crashes (this will be a program bug, MS-DOS doesn't crash spontaneously), you are in no danger of losing the beam. Let's take that a step further, and claim that a good many diagnostic tasks can live quite happily on an MS-DOS machine with (for example) a

GPIB card. Such dedicated diagnostic Front Ends, whose temporary disappearance (although a nuisance) is not fatal, we shall refer to as simple FECs (Front End Computer). Simple FECs are excellent candidates for PCs. By the way, this "temporary disappearance" might occur on any platform. Whether a bug causes the process to "core dump" or the machine to hang is immaterial. You'll still have to have a way of dealing with it.

Beyond Simple FECs are Complex FECs, i.e. Front Ends with multiple tasks, and/or Real Time requirements, and/or crucial importance. At HERA, we have many PC FECs meeting all of these criteria (Note: there are several Real Time Kernels for MS-DOS commercially available). Nonetheless, we prefer to leave the case of Complex FECs an open issue, and assume that we have a multi-platform control system. Indeed, as we shall shortly see, we incorporate VxWorks FECs, and FECs running as servers on a variety of UNIX and VMS machines.

4. Networked PCs

Following the "Standard Model", the CONSOLES and FECs at HERA reside on an ETHERNET. In addition, the PC FECs and CONSOLES are attached to a NOVELL file server. Furthermore, all control system code is located on the file server (and not on the PC!), meaning that at boot time a PC logs in and mounts network drives to the file server, and then loads and runs its code. In the case of a CONSOLE or an FEC running WINDOWS (most FECs run DOS), WINDOWS itself is also loaded from the file server. As WINDOWS generally has several open files and does considerable swapping, this does introduce a weakness in that the connection to the file server is of paramount importance, since if the connection is lost the workstation is almost guaranteed to crash. On the other hand, all backup strategies can be entirely focused on the file server. The weakness can be (and will be this shutdown) patched by mirroring all relevant software locally at login time, and always running from a local operating system.

Control System Model

So far, what we have described follows the so-called "Standard Model" of a control system, in that hardware-near FECs communicate with user-near CONSOLES over the ETHERNET. We have not insisted, however, that FECs are VME CPUs or that anyone is running UNIX. On the contrary, we strive for a multi-platform control system, where CONSOLES and FECs communicate with each other without concern as to what operating system is running at the other end. With that in mind, we make the following ansatz:

- Σ An FEC should present an integrated device to potential clients, i.e. a client should see an object representation of the device to which it is speaking, be it a magnet, oscilloscope, BPM or whatever. A client should never deal with raw data, but instead receive data ready for display, and issue commands via mnemonic properties.

This goes hand in hand with saying that any change of state made by a CONSOLE should be seen by all other CONSOLES.

Also important is that there should be no distributed control over the ETHERNET, requiring one FEC to be crucially dependent upon knowing the state of another FEC. To be sure, an FEC can be (and frequently is) a client of another FEC. However, the possibility that information might not come in on time or might not come in at all should be anticipated.

It remains to discuss specific communication models CONSOLE to FEC, and at this juncture we should mention our indebtedness to the ISOLDE project at CERN/PS and its developers. The ISOLDE concept was our starting point in 1991, and in some respects our system still bears a strong resemblance to ISOLDE although our needs and development began to diverge rapidly from the original concept in late 1992. Initially the communication model was pure client-server, in that a CONSOLE always asked an FEC synchronously for data. Polled data were requested from a timer on the CONSOLE. Also to be noted is that the original concept was a PC-only one and used only IPX-based protocols. In 1993, IP-based protocols were included, opening the door to non-PC communication partners. The fundamental model remained, however, "client-server". FECs were servers, and did not supply data unless specifically asked to.

Both IPX and IP communication channels were based on socket libraries, and specifically not commercial RPC products. This was largely owing to the fact that such products did not encompass both the IPX and IP worlds at the same time, whereas (working with sockets) the application layer and network/transport layers were easy to keep separate, facilitating development.

It soon became clear that certain data channels were of vital interest to a large number of CONSOLES and FECs. These were items such as the energy and current of the proton and electron beams. And rather than having N clients request the present values of these parameters from one server (where N is a large number!) it was decided to make such values available via broadcast. To be efficient, the value is broadcast upon change (according to an appropriate tolerance) or at the system heartbeat of once per minute. In this way, no one has to ask, one only has to listen. Here, we are slipping over to a "producer-consumer" model of data exchange, where an FEC is seen as a data "producer," who is simply providing data to any "consumer" who is listening. An FEC should be rightfully regarded as a "producer" anyway, in that it should be in background constantly reading hardware and preparing data for display. Requests for data from the client side should in most cases end up fetching data from RAM and not initiating a hardware read.

This pure "producer-consumer" model was only applied to the most important machine parameters, however. Nevertheless, a crucial step in the "producer-consumer" direction was taken in 1994, by adopting a registered consumer model in which data links were now registered at the Front End instead of being polled from the CONSOLE. In other words, an FEC would keep track of a "mailing list" of consumers of pertinent data. A consumer could request values at a specific polling rate, or to be refreshed only upon change. The data would then arrive entirely asynchronously. The reduction in wasted CPU time, not to mention network traffic, in such a model is considerable! It is not uncommon, for instance, to have a popular control application, which might be getting a data update at 1 Hz, run on 15 to 20 different client workstations. Rather than having all 20 stations individually ask a server-FEC for data (one packet to the FEC, plus a function call) and receive data (one packet from the FEC) at 1 Hz (times 20), the producer-FEC keeps track of all 20 clients, makes one and only one function call at 1 Hz, and sends the data out to its mailing list (one packet from the FEC times 20). Furthermore, as the operation is connectionless and asynchronous, there is never any waiting on the part of producer or consumer.

Acknowledgments are required under only one set of circumstances. If a client has requested to be refreshed only upon change of data, and the data have indeed changed, then the producer-FEC asks to be acknowledged upon receipt of data. Otherwise, a consumer-CONSOLE knows very well if its requested polling rate has been met or not, or if the system heartbeat time of one minute has been exceeded. If warranted, it will sound an alarm and make an effort to relocate the FEC. Similarly, when a consumer-CONSOLE registers with an FEC, it subscribes for a certain quantity of data updates. When the subscription runs down, it must renew its subscription, or it will be removed from the FEC's mailing list. In this way, there can be no dangling consumers. Maximum efficiency regarding network traffic is maintained by packing together in the same ETHERNET packet all subscriptions destined for the same consumer at the same time.

Command-based requests of course follow a client-server approach. The client has the choice of sending commands synchronously or asynchronously. As the communication is connectionless, the turn around time for request and reply is typically 2 to 3 milliseconds (IPX is marginally faster than UDP) plus any time the FEC might need to read hardware.

These types of detail are of course hidden from the control system application programmer. What he sees is an API which tells him how to link data from a Front End into his control program, in the case of a CONSOLE application, or how to offer data for linking, in the case of an FEC application. Important is only that the Front End device shows up as a tag name in the API. The tag name is resolved at initialization into an FEC address and an equipment function living on the FEC, both invisible to the API. The FEC address will be either an IPX address, if both partners speak IPX, or an IP address.

At present, name resolution begins with a database file, which matches tag name to FEC name and equipment function name. This will be replaced by a name resolver during the 1996 winter shutdown. An FEC address is then established by the following: If the client is not a PC attached to the control system file server, then the local host table is scanned for the IP address of an entry whose alias matches the FEC name. If the client is a PC attached to the control file server, then the user list for the file server is scanned for a user matching the FEC name. If a match is found then an IPX address is established. If no match is

found then the local host table is scanned for an IP entry. All CONSOLES speak both IPX and IP. All PC FECs speak IPX, and speak IP only if necessary. And all non PC-FECs speak IP. We should also mention that there are non PC-clients to certain control system elements as well. These are principally machines outside the HERA control room, and serve to integrate the various sub-systems of the HERA machine and experiments.

An important element to the control system structure is the concept of the Data Server. This is a machine designed to acquire all important machine parameters at a sufficient rate so as to be able to act as a data gateway to clients outside the immediate control sub-system. It is both a client and a server with a large number of channels, and is in general very busy. This is nonetheless a much more efficient model for N clients to obtain machine data, than for the same N clients to form individual links to potentially many different front ends (largely owing to the data packing mechanism described earlier). The most important subset of machine parameters is broadcast from this machine as described above. Furthermore, as this machine always has an up-to-date record of the machine parameters, it also serves as the control system archiver.

As to archiving, machine state variables are regularly archived (with appropriate filters) allowing data retrieval and correlation throughout the year. Similarly, critical events (such as a magnet quench) can initiate archive dumps of the state of a particular subsystem at the time of the event. This information is of course of vital interest to the systems engineers.

The control system is also designed to be open, in that office client machines also have access to control system data and applications, but with certain restrictions. Namely, the FEC server itself always knows the identity of the caller and can pre-specify a list of users with WRITE access. READ access is generally allowed since data READs do not change the state of the hardware. Likewise, client applications can also choose to hide options from under-privileged users.

HERA: Current Status

So where are the PCs in HERA? As of this writing, CONSOLES in the control room are either PCs running WINDOWS or NORD mini-computers running SINTRAN, in approximately equal numbers, with the NORDs living on borrowed time. The current generation of PCs in the control room are 486 33 MHz or 50 MHz machines. These will likely be replaced by PENTIUMs in the coming year.

Most components of the proton beam diagnostic controls, as well as some electron diagnostic controls, are incorporated on PC FECs running MS-DOS. A small minority run MS-WINDOWS, in cases where a user-friendly GUI is important at the front end. And there are also a number of non-PC FECs which are a fully integrated part of the HERA control system. These are summarized in Table 1.

As the data exchange mechanism described above has been ported to most platforms seen at DESY, a number of control sub-systems provide data server gateways and/or are clients to the HERA data server. The Proton Vacuum and Cryogenics systems, for instance, both of which are autonomous control systems in themselves, are persistent clients to the HERA data server.

HERA: Next Year

One of the last and most important steps in upgrading the existing HERA control system involves magnet control. As yet the 1200 electron and proton magnet controllers are driven by 4 NORD minicomputers. There are two promising approaches currently being tested at DESY. One involves scaling the all-PC PETRA control system to HERA, and the other involves using Symmetric Multi-Processing (SMP) on a multi-CPU SUN workstation (see Herb and Wu, "Accelerator Magnet Control from an SMP Workstation", poster session this conference). In any case, the 1200 individual channels necessitate a multi-CPU approach. The SMP solution would control all magnets from one computer whereas the PC solution would span more than one. The PC solution appears to work well in PETRA. The number of magnets there is considerably smaller, and all magnets can be controlled from one PC. In HERA the horizontal and vertical correctors could conceivably be controlled from separate PCs.

Table 1. Front End Computers at HERA following the PKTR Control System Model.

FecName	OS	Description	Hardware
ADDA	DOS	Hera Tune Control	i386, GPIB
BEAMSCOP	DOS	Hera P X-channel/Y-channel Beam Scope Monitor	i386, GPIB
BPM	DOS	Hera P BPMs,BLMs,MTMs,Delay Modules,Alarm Loop Modules	i486, 4x SEDAC, V24
CMFL	DOS	Hera P Fast Current (Lopez) monitor	i386, GPIB
DATASERV	DOS	Hera P Data Server	i486, Watchdog
BEAMCURR	DOS	Hera P and E beam current monitors	i386, GPIB, SEDAC, Watchdog
FECSIM	DOS	Hera P development FEC	i386, Watchdog
HERA208	DOS	Hera RF (208 MHz)	i486, SEDAC
HERA52	DOS	Hera RF (52 MHz)	i486, SEDAC
HERAQ	DOS	Hera P X and Y tunes	i386, 2 DSPs
HIT	DOS	Hera Integrated Timing	i386, GPIB
PETRA52	DOS	Petra RF (52 MHz)	i486, SEDAC
SCRAPERS	DOS	Hera P Collimator Steering	i386, SEDAC, Watchdog
OSZIS	DOS	Oscilloscope Steering	i386, GPIB, V24
IPS104	UNIX	Orbit Correction	HP
MKI101	UNIX	Orbit Correction, Magnet Data	HP
SUN1	UNIX	Development	SUN SPARC
SUN2	UNIX	NORD Gateway	SUN SPARC
EMIT_PP	DOS	Petra P Residual Gas Monitor	i386, SEDAC
BLME	DOS	Hera E BLMs	i486, 4x SEDAC
EMIT_HP	DOS	Hera P Residual Gas Monitor	i386, SEDAC, ADC
ZLUM01	UNIX	Zeus Lumi Workstation	SGI
DORCAV	DOS	Doris Cavity	i486, SEDAC
BUNCHCUR	DOS	Hera-P Bunch Current	i386, GPIB
WIRE	DOS	Hera-B Wire Target	i486, SEDAC
DORQ1	DOS	Doris Q1 Sender	i486, SEDAC
HPPET_R	DOS	Petra Sender	i486, SEDAC
DORQ4	DOS	Doris Q Sender	i486, SEDAC
VWMASTER	UNIX	VxWorks (Development)	VME, CAN
VWSLAVE	UNIX	VxWorks (Development)	VME, CAN
PETRAQ	DOS	Petra Tune	i286, 3 DSPs
PETRAI	DOS	Petra Beam Current	i386, SEDAC
HPPET_L	DOS	Petra Sender	i486, SEDAC
TIC	DOS	Hera Integrated Timing	i486, GPIB
TUNNEL_O	DOS	Tunnel Temperatures	i386, SEDAC
TUNNEL_W	DOS	Tunnel Temperatures	i386, SEDAC
DORFB	DOS	Doris Q2 Sender	i486, SEDAC
BCURRE	DOS	E Beam Current	i386, GPIB
LPSVAX	VMS	ZEUS Roman Pot Positions	VAX
PROXY	DOS	Middleman FEC: Quench Proxy, Netmex Proxy	i386, Watchdog
BLMDSY	DOS	DESY BLM FEC	i386, SEDAC
DESYGAS	DOS	DESY Gas Monitor	i486, GPIB
TUNEMOD	DOS	Hera Tune Modulator	i486, GPIB
LPS_H1	WINDOWS	H1 Roman Pot Positions	i486, SEDAC
WINFEC	WINDOWS	Windows FEC simulator	i486
DESYWS	DOS	DESY III Wire Scanner	i486, SEDAC
HWEST	DOS	HERA Sender (Development)	i486, SEDAC
HF_HE_SL	DOS	HERA Sender	i486, SEDAC
HF_HE_SR	DOS	HERA Sender	i486, SEDAC
HF_HE_OL	DOS	HERA Sender	i486, SEDAC

Table 1. Front End Computers at HERA ... (continued).

FecName	OS	Description	Hardware
HF_HE_OR	DOS	HERA Sender	i486, SEDAC
HF_HE_NL	DOS	HERA Sender	i486, SEDAC
HF_HE_NR	DOS	HERA Sender	i486, SEDAC
HF_HW_FB	DOS	HERA Sender	i486, SEDAC
MKI102	UNIX	Orbit Corrections	HP
IPS109	UNIX	Orbit Corrections	HP
VWWEST1	VxWorks	SPS Master	VME, CAN
VWNORD1	VxWorks	SPS Master	VME, CAN
VWEAST1	VxWorks	SPS Master	VME, CAN
VWSUED1	VxWorks	SPS Master	VME, CAN
VWWEST0	VxWorks	Transient Recorders	VME, CAN
VWNORD0	VxWorks	CPU 0	VME, CAN
VWEAST0	VxWorks	CPU 0	VME, CAN
VWSUED0	VxWorks	CPU 0	VME, CAN
VWWEST2	VxWorks	SPS Slave	VME, CAN
VWNORD2	VxWorks	SPS Slave	VME, CAN
VWEAST2	VxWorks	SPS Slave	VME, CAN
VWSUED2	VxWorks	SPS Slave	VME, CAN
ZIZI	UNIX	Zeus Lumi Data	SGI
EMIT_HPS	DOS	HERA Gas Monitor	i486, GPIB
DESYSCOP	WINDOWS	DESY Scope Gas Monitor	i486, GPIB

The PETRA model is worth mentioning here. For one thing, it is a much more homogeneous model where all of the players (with very few exceptions) are PCs running MS-WINDOWS. This in itself has merit, as we shall shortly see. Furthermore, the control system ETHERNET segment is kept isolated, with only a gateway interface to the outside. Most of the traffic on that segment is restricted to pure producer-consumer traffic, or rather "producer traffic." Data are flushed from the front ends onto the net via broadcast at a cycle frequency of 1 or perhaps 2 Hz. The consumers only listen; they don't introduce any extra traffic. Of course allowing commands from the console necessarily implies that client-server traffic also appears from time to time, but not in sufficient quantity to upset the overall loading.

Which, if either, of these solutions will be adopted remains to be seen. However, we should note that if the SMP approach is adopted, this will add yet another make of 'Mercedes' on the control system. Is this bad, good, or irrelevant?

There is much to be said for keeping a system as homogeneous as possible. The expertise developed in dealing with a specific platform is then shared by many, and no one person ever becomes indispensable. On the other hand, there is also a danger in putting all of our eggs in one basket (however mighty Microsoft or Wind River appears at the time). In this regard it is perhaps more prudent to relax our definition of homogeneous a bit. In the end, "How we got there" is more important than "Where we went." In other words, if I can take the same code and compile and run it on another platform with minimal effort, then my learning curve on the new platform is not so steep. On the contrary, I am likely to feel at home in my new environment, and my general knowledge will increase steeply. I then also have the ability to "plug and play" at the front end, where (as long as I adhere to our initial ansatz of the control system model) I can completely alter the machine and reconnect it to the control system, without any of the other participants knowing. Rather, the application layer will know that something has changed and figure out how to deal with it, but the applications programmer still sees the same object on the other side of the API.

As to the data exchange mechanism, the homogeneity of the HERA system lies in the API, which is an identical C interface across all of the previously mentioned platforms. Reading hardware might of course require a completely different expertise on one system than on another, and in this light there remains a

strong bias in sticking to the PC environment. Nonetheless, as UNIX offers a fairly standard working environment across many platforms, UNIX solutions to controls problems are not to be avoided, on the general principle of hedging our bets. Being able to easily incorporate a special UNIX solution to a controls problem and at the same time to allow an engineer who has tested and developed his hardware with a PC to trivially include his work in the control system proper are both good capabilities to have.

The waters will undoubtedly be muddied even further in 1996 as the control system API will be ported to WINDOWS-NT, i.e. to the WIN32 API.

C or C++ is the natural language of choice when interfacing with the control system API. The notable exception is under WINDOWS where, since it offers such an outstanding GUI, Visual Basic is used. Visual Basic itself fills a development tools niche so well that there has been over the past two years a flurry of activity from several vendors either to offer something similar (or better) or to port Visual Basic to non-PC platforms. The ease with which one can learn and program something useful with Visual Basic is astounding. This point should not be taken lightly, as a good many of our own CONSOLE applications have been written in this language by machine physicists whose last programming experience had been FORTRAN IV several years ago. This in turn enables the software engineers to devote more time at lower systems levels. Future incarnations of Visual Basic will almost certainly offer inheritance and polymorphism to its GUI objects, making it a legitimate object oriented language.

Acknowledgments

I would like to express my indebtedness to A. Pace and I. Deloose at CERN/PS for many of the thoughts and ideas presented here. I would also like to explicitly thank S. Herb, K.H. Mess, H.G. Wu, and F. Peters for many useful discussions and rounds of constructive bickering.

Using A Public Domain Real-Time Kernel On A VME/Ethernet Based Control System

David E. EISERT

Synchrotron Radiation Center, University of Wisconsin-Madison, 3731 Schneider Drive, Stoughton, WI 53589-3097, USA

There are many options available when choosing a real-time kernel from full-featured operating systems to writing your own kernel. The cost of these systems also covers a large range from tens of thousands to only a few hundred dollars. Recently there has been added yet another option, well documented real-time kernels that have been submitted to the public domain, these include Chimera (Carnegie Mellon University), RTEMS (U.S. Military), and μ C/OS (Jean J. Labrosse). We selected the μ C/OS kernel because it is small and has the required capabilities. Although the kernel was ported to the system in a matter of days, several months of development were required to write device drivers for the processor board. This paper will discuss why we selected the μ C/OS kernel, the effort required to implement this kernel and our experience with the completed system.

1. INTRODUCTION

The Synchrotron Radiation Center (SRC) designed and built the Aladdin storage ring in the late 1970s. The original Multibus-based controllers were replaced with VME/Ethernet-based systems in 1986 [1]. The VME CPU boards used in that upgrade were based on a 12.5 MHz Motorola 68000 with a total on-board memory of 512 KB. A second VME board was needed for the ethernet interface and was based on the AMD LANCE chip set. The software for the systems was written in assembly language and was made to imitate the original Multibus systems. We wanted to migrate to a high-level language but the compilers available at the time produced inefficient code and consumed a large amount of system resources. Software from the VME CPU board vendor consisted of a monitor ROM and other commercial real-time kernel vendors lacked support for ethernet networking. We had no other choice but to write all the drivers ourselves. Much has changed since then, with both VME CPU boards and commercial software support for networking in real-time kernels. As a result we decided to replace our VME CPU boards sometime in the 1992-1995 budget period, but unfortunately the budget only allowed about \$25K for the ten 6U VME systems used on the ring.

In order to understand our choice of a real-time kernel, some background into the selection of a CPU board is necessary. In early 1992 the SRC Optics group was working on a new beamline that required a dedicated control computer to implement a feedback loop for precise positioning of an optical component [2]. The SRC Controls group was called upon to assemble a VME system for this beamline. We decided not to use the older MC68000 VME CPU boards but to evaluate other boards that could later be used on the storage ring. We selected the Heurikon V3D with a MC68030 processor, an optional floating point coprocessor, a 32-bit ethernet coprocessor and up to 16 MB of parity protected RAM. The vendor offers two commercial real-time kernels with network support but the cost of the development system for these kernels exceeded our available budget. Software support was not a consideration in our case due to lack of funds to purchase a real-time kernel and the availability of GNU C++ for software development. In addition, we were purchasing only a single VME CPU board for this project with some prospect of purchasing more boards at a later date. It was very hard to justify the cost of a real-time development system for only a single board.

2. THE KERNEL

Since we had already decided not to purchase a real-time kernel, our original plan was to copy much of the assembly language kernel from the older VME boards. The older kernel simply remained suspended waiting for interrupts from periodic timer events, ethernet messages or other I/O device requests. The interrupt handlers would then process as much as they needed and place the address of a routine that would do further processing into a FIFO queue. After the interrupt returned, the routine would be pulled off the queue and executed. This method lacked many desirable features available in a pre-emptive multitasking kernel but proved adequate for the tasks assigned to the VME systems at that time.

2.1 The μ C/OS Kernel

As the device drivers for the on-board hardware were being written, *Embedded Systems Journal* published a series of articles describing the μ C/OS real-time kernel [3,4]. μ C/OS is a preemptive multitasking kernel with

semaphores, message mailboxes and dynamic task priorities. This kernel was originally developed for 8-bit microcontrollers and as a result the code was optimized for size and efficiency. The author also tried to make the kernel as portable as possible and reduced the amount of assemble language code to a minimum. μ C/OS does have some limitations including a maximum of 63 tasks, each task must have a unique priority and it does not include any of the utility tools normally included with a commercial kernel development package. μ C/OS is technically not in the public domain. The license agreement allows distribution of the object code but not the source code.

The process of porting the kernel proved quite simple. The approximately 600 lines of C source code compiled with only a few changes to remove PC style memory declarations. The less than 100 lines of Intel 80186 assembly language were only used during context switches. The context switches occur only after an interrupt. A software interrupt is used to perform a context switch after a new task is added or a task has changed priority. An assembly language interrupt wrapper had already been written so that interrupts could call interrupt handlers written in C. The interrupt wrapper saves the registers, looks up the address of the interrupt handler for the given interrupt vector and calls the handler. When the handler returns, the interrupt wrapper restores the registers and issues a return from interrupt instruction. To handle the context switch a few lines had to be added to the interrupt wrapper. On entry to the interrupt wrapper, registers are saved onto the current task stack and the stack pointer is switched to the interrupt stack. As the interrupt handler executes, calls to μ C/OS routines made from the interrupt handler may change the highest priority task ready to execute. When the interrupt handler returns to the interrupt wrapper, the highest priority task is found and the stack pointer is moved to that task's stack. When the interrupt wrapper issues the return from interrupt instruction, the new task starts to execute. The whole process of porting the kernel required only a few days.

2.2 Utility Routines

Many real-time kernels supply utility routines that include memory allocation, string handling, and software downloading. Unfortunately μ C/OS lacks these routines but source code to many of these routines is readily available in the public domain. In fact, the Free Software Foundation supplies a very complete set of the standard C library and a C++ library. This library was considered, but since the library calls many kernel routines a complete port of this library would have been difficult.

Three basic functions were desired from the standard C library. The string handling functions such as *strcpy* and *strlen*, the string formatting functions *printf* and *scanf* and the memory allocation functions. String handling and formatting functions were obtained from a public domain C library that was not as complete as the Free Software Foundation's C library. These routines compiled easily and did not contain any kernel calls. The memory allocation routines were acquired from an article published in the *Embedded Systems Journal* [5]. The routines presented there are well documented and integrated easily with the existing kernel and library routines.

The final utility routines consisted of downloading and debug monitor routines. In our case we had been using the ethernet for software downloading for many years and software downloading would be handled later in the network software. The debug routines that include examining memory and other system operations proved simple to write once the standard library routines for string handling and formatting were available.

3. WRITING DRIVERS

We had been using the network interface for downloading code into the older VME systems almost from the start of the 1986 upgrade. Thus one of the early goals was to use our ethernet based monitor, written for the older boards, to download code into the new boards. The serial port still remains a vital tool for debugging purposes and the serial port driver was implemented after the ethernet driver. A small version of the serial driver was used to dump out messages during the development of the ethernet driver. The last system driver needed was a timer driver. The timer driver is used for preemptive multitasking and for precise delays required by some tasks.

3.1 Ethernet Driver

The ethernet coprocessor used on the Heurikon V3D is the Intel 82596. We had previous experience with the Advanced Micro Devices LANCE ethernet chip set so many of the functions were similar. The main drawback to the Intel coprocessor is that it was designed primarily for little-endian processors with enhancements for big-endian processors. The big-endian mode handles only some of the data ordering problems and close attention to byte ordering is critical during driver development. One advantage of the coprocessor is that it can be programmed to ignore ethernet broadcast messages. At present our VME systems are still connected to the campus wide network through several bridges that eliminate all unnecessary traffic except for ethernet multicasts and broadcasts. The internet protocol uses a great deal of broadcast messages during normal operation and very infrequently "broadcast

storms" on the campus wide network would swamp the older VME systems. We did not use a standard network protocol at the time this driver was being developed and only recently have we started to use the internet protocol.

The ethernet driver is composed of three main parts, the initialization of the coprocessor, the interrupt handler and the packet transmission routines. During the initialization phase the ethernet coprocessor is given its ethernet address, any multicast addresses it may accept and a link list of memory buffers to place received packet data. The coprocessor has many different memory organizations available for received packet data, the simplest of which is a link list of buffers. The interrupt handler deals with processing the received packets, cleaning up data structures after a packet has been transmitted and handling any errors generated by the coprocessor. When the interrupt handler is notified that a new packet has been received, it first checks the packet destination address to make sure it is either addressed directly to this system or it is a multicast packet that this system has been enabled to receive. The coprocessor does not completely filter all multicast addresses and other multicast packets can be received by the coprocessor. Next, the header is checked for the proper format and improperly formatted messages are discarded. Finally the packet is placed in a FIFO queue to be processed by a network command handler task. The last set of functions performed by the driver allow other tasks to get a properly formatted network message buffer and then pass it to the coprocessor to be sent out.

3.2 Serial Port Driver

The main task of the serial port driver is to stream bytes into and out of the serial port chip. This is implemented by having two circular queues per serial port, one queue for the received characters and one for the transmitted characters. Counting semaphores are used for both queues with the receive semaphore set to zero and the transmit semaphore set to the transmit queue size. When a character is received, the driver increments the receive semaphore. Thus when a task asks for a character string from the serial port with a call to *gets*, the task is suspended waiting for a non-zero count in the receive semaphore. When a character is received this task is made ready and the *gets* routine places the new character in the task's buffer. This process continues until the end of line character is received. When a task wishes to send a character string out of the serial port, the characters are placed in the transmit queue and the transmit semaphore is decremented by one for each character in the string. If the transmit queue fills before all the characters can be placed into the transmit queue, the task is suspended waiting for some characters to be sent and space to be made available in the transmit queue.

3.3 Timer Driver

The timer driver is designed to have very fast response and as a result bypasses most of the kernel. The timer driver builds a list of time events ordered from the shortest time delays to the longest. The driver adds a new timing event into the queue by subtracting all the time from events that occur before the new timing event in the queue. When the time has elapsed for the time event in the head of the list, the timer driver calls the routine associated with the time event at the timer interrupt level. This means that the routine must function very quickly or signal a task to complete the work outside of the interrupt level. One of the main timer events is the call to the kernel routine *OSTimeTick*. This routine decrements the delay counter of any task that voluntarily suspends itself a number of kernel time ticks. When the task's delay counter goes to zero, the task is made available to execute. When the timer interrupt returns, the context switch is made to the highest priority task available to execute. Another timer event is used for ramping digital to analog converters (DAC). The fast response of the timer driver allows for a time delay resolution in the tens of microseconds. With this resolution the DACs can be ramped very smoothly.

The timer driver uses a timer chip that derives its clock from a crystal oscillator. Crystal oscillators are reasonably good for short time periods but they display long term drifts. In addition, the timer driver may have difficulty obtaining the exact elapsed time between removing the time event from the head of the queue to starting the next time event in the queue. Therefore the timer driver makes small adjustments to the kernel time tick delay by using a battery-backed real-time clock. Every 30 seconds the real-time clock is read and compared to the elapsed time from the time tick. If a discrepancy exists, an adjustment is made to realign the clocks over the next 30 seconds. The small adjustments do not effect the timing of any other timing event but long term times that are obtained from the time tick are accurate to about 2 seconds per day.

4. SYSTEM TASKS

All systems contain a few common tasks needed for normal operation. These tasks include processing network messages, serial port messages and periodic I/O driver tasks. The network task parses the received ethernet packets and performs the operations detailed in those messages. The monitor task waits for any commands typed into a serial port and executes them. Finally, the driver task calls all the I/O drivers at a periodic rate so the I/O drivers can perform any operations off this event without creating its own timer event.

4.1 Network Task

The network task suspends waiting on the ethernet driver's receive packet semaphore. When a packet is received by the ethernet driver it places the packet into a FIFO queue and sets the receive packet semaphore. The network task removes the packet and inspects the header of the packet for a proper network protocol. Two protocols are currently supported, a private protocol developed in 1986 and the internet protocol. The network driver then removes the first data word in the packet. This data word is used in a *switch* statement to call a routine to process the rest of the information in the packet. The routine may load software, read or write information from a driver, or perform some other task.

We used our own packet format during the 1986 upgrade because we found a simple method on our VAX/VMS systems to send and receive raw ethernet packets. VAX/VMS has supported this raw packet I/O for many years and even after several VAX/VMS operating systems revisions it has remained unchanged. Back at that time we were considering DECnet as the network protocol but we never implemented DECnet because only minor software maintenance was required to support the raw packet I/O. We did not consider TCP/IP because there was only third party support for TCP/IP under VAX/VMS.

Recently it has become obvious that the internet protocol is the primary network protocol in use and we have implemented a subset of that protocol on our VME systems. Although formal implementations of TCP/IP require a minimum set of functions, it is possible to implement a much smaller set of functions. The minimum subset of TCP/IP needed for communication are the internet protocol (IP), user data protocol (UDP) and the address resolution protocol (ARP). The IP protocol is actually just 20 bytes of information inserted at the beginning of the network packet. Eight of those bytes are the internet addresses of the sending and receiving systems, two bytes are used for the total length of the packet and another two bytes are used for a checksum of the IP part of the packet. The other bytes select various options for packet fragmenting, data protocol and routing. These options can normally be set to a fixed value for all transmitted packets. The UDP protocol is inserted after the IP section and consists of an eight byte header plus the actual user supplied data. Four of those bytes specify the source and destination UDP ports, two are used for the length of the UDP part of the packet and the last two bytes are used for an optional checksum of the UDP part of the packet. The actual implementation of these two protocols required less than 50 lines of code in the VME systems. The ARP protocol defines a method for ethernet based TCP/IP systems to obtain the physical

ethernet address of another system on the same ethernet segment. Basically a system will broadcast a simple packet that contains the IP address of the system for which it wants to find the physical ethernet address. Then all systems on the ethernet segment inspect the packet and compare their IP address to the address given in the packet. If their IP address matches the one in the packet, that system replies to the requesting system with a packet containing its physical ethernet address. Although the ARP protocol could be implemented very easily, we did not want to inspect all the broadcast packets on the ethernet segment. We avoided this problem by using a VAX/VMS system to handle the ARP requests for the VME systems. The VMS TCP/IP software can be set up to watch for other system's IP addresses on the ethernet segment and reply with the other system's physical ethernet address.

4.2 Monitor Task

The monitor task is used mainly for debugging software over the serial port. This task waits for character strings to be entered through the serial port, then it parses and executes the commands. The monitor task is not a complete debugger because it lacks features such as break points and watch points. Its main features are to examine/modify memory, display task status and display memory allocation status. All of these features are also available with the ethernet-based monitor software that executes on remote networked PC systems.

4.3 Driver Task

The driver task is used to periodically call all the I/O drivers on the system. When the driver is called, it can perform basic tasks that are relative to the kernel time tick. An example would be for a driver to read all the channels of a multiplexed ADC and log those readings over time. After the driver task has called all the I/O drivers on the system it voluntarily suspends itself for one kernel time tick.

5. FINAL INSTALLATION

As noted earlier, the initial implementation of the kernel and CPU board drivers was on a single storage ring beamline. The beamline system had several types of VME I/O boards but the storage ring uses different I/O boards. Thus drivers for the storage ring VME I/O boards had to be completed before the new CPU boards could be installed on the ring.

5.1 Hardware Drivers

Development of the storage ring VME I/O board drivers proceeded quickly due to many years of familiarity with the storage ring I/O boards. The first CPU board was installed on the ring within two months after they arrived. The rest of the CPU boards were installed over an eight-month interval to allow for normal operation of the storage ring.

5.2 Maintenance

Even though the new kernel had been running in the beamline system for over a year we still discovered two bugs in the ethernet driver. The first problem appeared when one system would generate a Spurious Interrupt after several weeks of operation. It turned out that the ethernet coprocessor would signal an interrupt request but when the processor issued an interrupt acknowledge the coprocessor did not respond. The board was sent back to the manufacturer but they did not find any problems. As more CPU boards were installed on the storage ring we noticed that some boards never demonstrated this problem and others would generate the error almost weekly. The problem was resolved by trapping the Spurious Interrupt in the ethernet driver and checking the status of the ethernet coprocessor. If the coprocessor status word indicated that it was requesting service, the ethernet interrupt service routine would be executed. The second problem was discovered recently that causes an ethernet coprocessor fault from insufficient memory buffers for received packets. Apparently a burst of unfiltered multicast traffic on the ethernet segment arrived at such a rate that the processor could not discard the packets quickly enough. The ethernet driver should have reset and reinitialized the coprocessor but the code did not clean up some memory structures properly. The conditions that generated this fault are almost impossible to reproduce. They occurred on several occasions during a few month time period and then seemed to have stopped. Code that corrected the problem was placed in a few systems and those systems have shown the ability to recover from one such event.

Another indication of the quality of the kernel can be observed from the ability to modify the system software as hardware changes are needed. Recently we upgraded some of our analog converters from 12-bit converters to 16-bit converters. After the new 16-bit ADC boards were selected and ordered, the driver was written for the boards several weeks prior to the expected delivery. The driver was fairly complex requiring an interrupt service routine to handle the hardware scanning of the ADC. Coding of the driver required 400 lines of C and was completed in a single day.

When the board arrived, the driver functioned without a single problem. The first board was placed into operation on the storage ring two days after the boards had arrived.

6. CONCLUSION

Although the kernel only required a few days to port, the device drivers for the CPU board required more than a month of work. Many more months of work were required for the complete set of VME I/O drivers. Many believe that commercial real-time kernels are well worth their purchase price. One would clearly purchase a commercial kernel for large projects involving many programmers. For smaller facilities such as SRC the decision is not as clear. Three costs have to be considered when writing a kernel, the development cost, the project delay cost and the maintenance cost. We did not have any problems with project delays and maintenance costs have been minimal. The development costs were reduced substantially by using public domain code in the kernel.

This work was supported by the National Science Foundation. The current contract number is DMR-9212658.

References

- [1] J. P. Stott and D. E. Eisert, *Nucl. Instr. and Meth.* **A293**, 107 (1990).
- [2] D. E. Eisert, M. Bissen, M. Fisher, R. Reininger, J. P. Stott and H. Höchst, *Rev. Sci. Instrum.* **66**, 1671 (1995).
- [3] Jean J. Labrosse, *Embedded Systems Programming* Vol. 5 No. 5-6, (1992).
- [4] Jean J. Labrosse, *μC/OS The Real-Time Kernel*, R&D Publications, (1992).
- [5] Leslie Aldridge, *Embedded Systems Programming* Vol. 2 No. 7, 28 (1989).

NLS Beamline Control and Data Acquisition Computer-System Upgrade

Shuchen Kate Feng, and D. Peter Siddons

National Synchrotron Light Source

John Skinner and Robert M. Sweet

Biology Department

Brookhaven National Laboratory, Upton, NY 11973, USA

The NLS beamline computer systems have gone through two major updates: (1) a graphical user interface (GUI) was incorporated into the original beamline-control and data-acquisition program ACE to form the new composite program GrACE, and (2) we updated device drivers to run GrACE on a Pentium PC running the UNIX-like operating system LINUX. With a proper PC CMOS setup and a PCI local-bus graphics card, the LINUX X server on a Pentium is dramatically accelerated to run GUI applications. With in-house-designed CAMAC and MCA device drivers and a public-domain GPIB device driver, we can run the original beamline-control and data-acquisition program without changing the computer-interface hardware. The updates produce powerful, low-cost, flexible, and user-friendly computer systems.

1. Introduction

The original NLS beamline-control and data-acquisition program ACE [1] had enough functions to serve its purpose. However, we recognized that a graphical user interface on top of ACE is necessary to provide a friendlier user interface. GUI applications had tremendously negative impact on program execution speed due to the poor architecture of the software and hardware of the original 486 PC-UNIX platform[2]. LINUX [3] supports a wide range of software, from X-windows to the GNU C/C++ compiler to TCP/IP. It's a versatile, *bona fide* implementation of UNIX, freely distributed by the terms of GNU General Public License. After Intel marketed the Pentium, some developers wrote device drivers for PCI local-bus graphics cards to take the advantage of the local bus system capable of moving 32 bits of data at 33 MHz which accelerates GUI applications dramatically. We updated the original system with low-cost computer systems and in-house-designed software to achieve powerful, reliable, and user-friendly computer systems.

2. Hardware configuration

The PCI local bus greatly improves I/O performance, especially graphics. The PCI bus can transfer data between the processor and the peripherals at up to 132 MB/s, far faster than the ISA bus rate of 5 MB/s. A full-featured PCI-compliant VGA card, with at least 1 to 2 MB of video RAM, will accelerate graphics performance further. We bought several 66MHz, 100MHz, and 120MHz Pentium P5 [4] computers, each equipped with a PCI EIDE hard disk controller, thirty-two megabytes of RAM, a 256 kB cache, an ATI MACH 64 with 2MB of VRAM, and a 1GB hard disk drive. We chose PCI local-bus Ethernet cards [5] for network communication.

Table 1 and table 2 shows the CPU and system benchmark among various computer systems we studied during evaluation. We only compared systems under \$12K in price and picked the best values.

TABLE 1. CPU benchmarks [6]

	Pentium 66MHZ	Pentium 120MHZ	HP 715/50	HP 715/80	SUN/SPARC/ 20/50	SUN/SPARC/ 20/61
Cache size	256 KB	512 KB	256 KB	256 KB	256 KB	256 KB
SPECint92	78.0	133.7	49.2	83.5	69.2	88.9
SPECfp92	63.3	99.5	78.8	120.9	78.3	102.8

TABLE 2. System performance using BYTE Benchmarks [7]

	Pentium P5	Pentium P5	Pentium P5	Sun Sparc 10
Kernel	LINUX 1.2.8	LINUX 1.2.8	LINUX 1.2.8	SUN OS
Clock	90 MHZ	100 MHZ	100 MHZ	50 MHZ
Hard disk controller	EIDE	PCI EIDE	PCI SCSI-II	
RAM	32 MB	16 MB	32 MB	128 MB
Average Index	6.6	11.6	11.6	7.9

The system performance was an average index of arithmetic tests, file copy, pipe-based context switching tests, shell scripts and so on.

3. Software development

Some of the source codes of the following software development is sharable [8]. The user manual for ACE is on the World Wide Web [9].

3.1 Device driver development

We wrote a device driver for the AT-bus PC004 PC-CAMAC interface card for a 6001/6002 CAMAC crate controller, available from DSP Inc., and a device driver for the AT-bus PCAII multi-channel analyzer from Nucleus Inc.

We obtained the GPIB device driver for LINUX from a public domain site [10]. One has to modify the kernel to use DMA if the PC has more than sixteen megabytes of RAM. One could leave the kernel untouched by disabling the DMA manager. On the GPIB board [11] we used, the DMA operation is unimportant when a chunk of data less than 32 bytes is transferred and most effective when a chunk of data more than 512 bytes is transferred.

In LINUX, these device drivers can be installed on the fly without modifying the kernel or rebooting the system.

3.2 GUI development

We wanted to add a GUI on top of ACE with minimum changes in the original structure of the ACE program. We developed three GUI processes to communicate with ACE through pipes. It not only facilitates debugging

but also preserves flexibility because each process can run independently. Furthermore, this kind of structure will facilitate further development into a client/server application. Figure 1 shows the current structure of GUI/ACE.

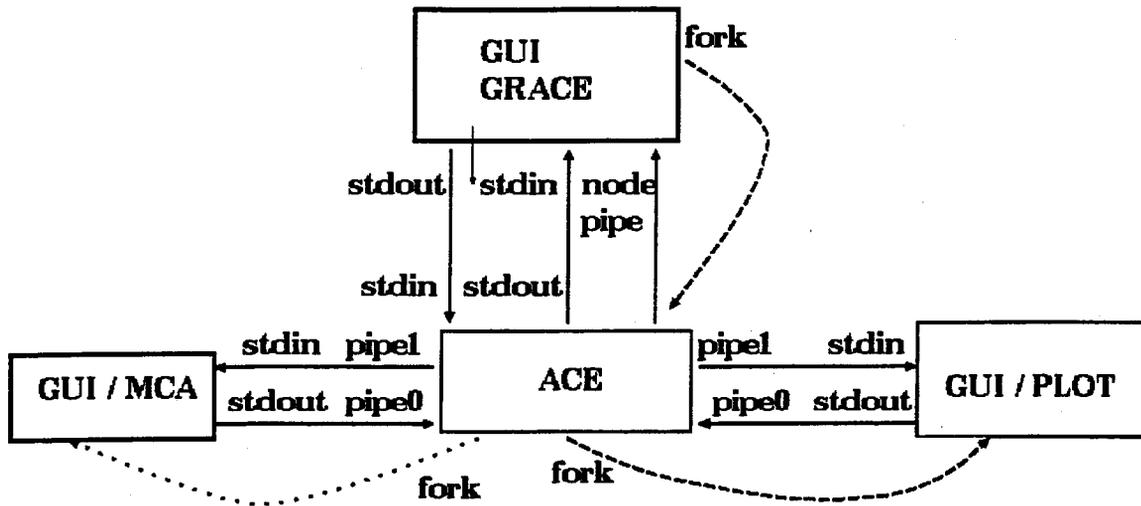


Figure 1: GUI/ACE

3.2.1 GUI/GrACE process

Figure 2 shows one example of the GUI/GrACE which maintains a scrollable window to print out the original screen message from ACE (on the lower left corner). In addition, it shows the GUI for the motor database editing, align/scan and counting. Another innovative design is to use Tcl/Tk to implement a GUI builder to build a user's own macro definitions. GrACE sends strings of commands to the ACE server whenever users click on a GUI as if those commands were typed out by them[12]. The Builder Xcessory[13] was used to build the GUI.

3.2.2 GUI/MCA and GUI/PLOT processes

Figure 3 shows one example of the GUI/MCA and GUI/PLOT called by ACE during a scan. Both processes, capable of running alone, will display the X and Y positions of the cursor in the plotting window interactively. Initially, the Builder Xcessory was used to layout the GUI. Thereafter, a plotting and user-interface library using Motif and X-windows graphics were written in-house to develop GUI processes.

4. Discussion

To integrate the LINUX OS on a Pentium PC demands skillful system administration. Although it is not quite well supported commercially[14], we can fulfill most beamline applications to achieve a powerful and user-friendly computer system.

Acknowledgments

This work was supported by the U.S. Department of Energy under Contract DE-AC02-76CH00016 with Associated Universities, Inc.

ACE : MOTOR DATABASE

Motor Code Motor Name Active Motion Disable Backdash Scan Feedback

Motor Group Beamline Hutch Other

Driver Type MMC E500 DAC0 DAC1 DAC2 DAC3 DAC4 DAC5

Driver Modifier _x25m _x19m none Slot Number Motor Address

Unit Name Scale Factor Units Position Pulses Position

Min Speed Max Speed Acceler. Rate Backdash

Left Align Right Align Scan Start Scan End

Scan/Align Step Size Soft Limits Lower Limit Upper Limit

Data Source Calibration No. (monochromator)

ACE **ALIGN / SCAN**

File	Motors	Counters	Beamli	Code	Align Left	Align Right	Scan Start	Scan End	Step	Current Position	Allow Move
Command				fth	0.100deg	0.050deg	1.000deg	20.000deg	0.001deg	17.200deg	.
				fth	1800.000asec	1800.000asec	0.000asec	5.000asec	180.000asec	-2390.553asec	.
											.

14: Count low limit (beam monitor) : 100
 15: Counter Preset(preset counter) : -2
 18: Counter Time (preset counter) : 1
 17: Counter Base (preset counter) : 1
 18: Counter Delay (preset,seconds) : 0
 26: Plotter?(0,1: Epson;2-13:Laser) : 4
 30: Real time clock slot? (for Kinetic counter): 0
 40: Update ALL database settings.
 41: Update ANOTHER counter
 42: Quit. (Default)
 Please select the setup number. (42)
 ome: 2nd mono: 28.0000deg, mth: mono theta: 17.2000deg,
 mono: mono theta: 704.4828arcsec, tth: anal
 2the:-2390.5532asec,
 th: anal theta:-212.5784asec, mbt: btm mono slit: -2.0000mm,
 wtp: white top slit: 1.0000mm, wbt: white bottom slit: 1.0000mm,
 mtp: top mono slit: 20.0000mm, d0: dummy: 31.0000point,
 INPUT (h for HELP)

GUI Builder for ACE Macros

Macro Name:

Number of Parameters:

Counter Monitor

Counter Channels:

Count Time

Figure 2: GUI/GrACE calls ACE server

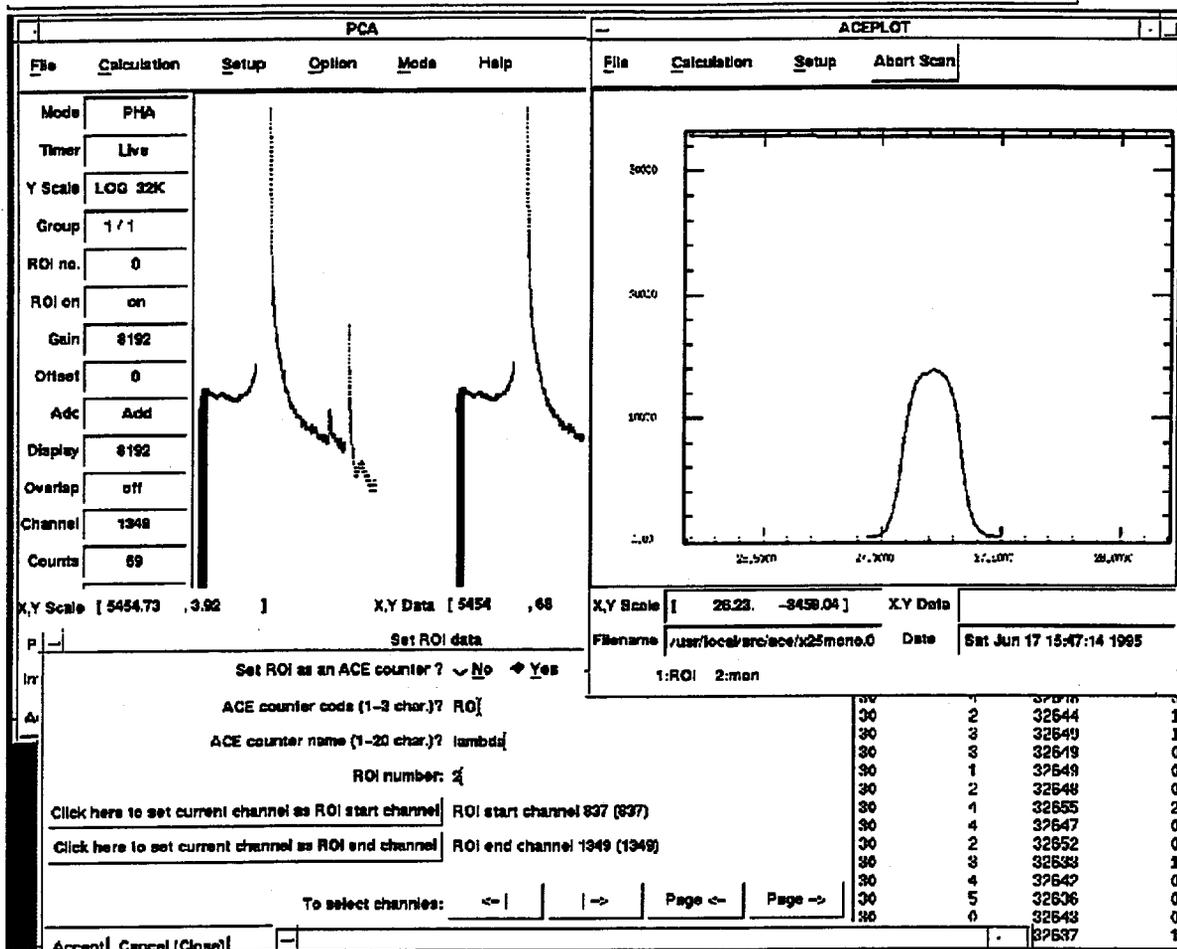


Figure 3: GUI/MCA and GUI/PLOT called by ACE during a scan

References

- [1] S. Kate Feng et al., "NSLS Beam Line Data Acquisition and Analysis Computer System". Nuclear Instruments and Methods in Physics Research A347 (1994) 603-606.
- [2] INTERACTIVE UNIX, available from SunSoft, (800) 227-9227.
- [3] Anonymous FTP sunsite.unc.edu:/pub/Linux/distributions, ftp.linux.org:/pub/mirrors/sunsite or other mirror sites.
- [4] Gateway 2000 Pentium P5, available from Gateway 2000, North Sioux City, SD, USA.
- [5] SMC8432BT ether power PCI ethernet adapter, available from SMC distributor.
- [6] Pentium benchmark was obtained from <http://web.jf.intel.com/procs/perf/bench/spec.html>, HP benchmark was obtained from HP9000 Series 700 Configuration Guide, SUN benchmark was obtained from anonymous FTP ftp.nosc.mil in /pub/aburto.
- [7] obtained from <http://www.silkroad.com/linux-bm.html>.
- [8] sharable form anonymous FTP ftp.nsls.bnl.gov in /nsls/pub/beamline.

- [9] <http://www.nsls.bnl.gov/BeamRD/beamlines/ACEman/man.html> for ACE, and <http://lsx12e.nsls.bnl.gov/x12c/grace.html> for GrACE.
- [10] obtained from anonymous FTP enif.astro.indiana.edu in /pub/linux/LINUX-LAB/IEEE488.
- [11] AT-GPIB/TNT controller, available from National Instruments, U.S.A., (800) 433-3488.
- [12] "Development of a GUI for an Existing Command-line Driven Program", J.M. Skinner, J.W. Pflugrath, and R.M. Sweet, Proceedings of SHARE 80, Winter Meeting, San Francisco, California.
- [13] Builder Xcessory, Integrated Computer Solutions Incorporated, U.S.A., (617) 621-0060.
- [14] S.K. Feng et al., "Application of a TMS320C31 chip for DSP/Embedded System", these proceedings.

Integration Feasibility of the Existing Linac Control System and Ring EPICS System at KEKB

Kazuro Furukawa*, Norihiko Kamikubota, Kazuo Nakahara, Isamu Abe and Akihiro Shirakawa
National Laboratory for High Energy Physics (KEK)
Oho 1 - 1, Tsukuba, Ibaraki 305, Japan

Abstract

In the KEKB project both the ring and the linac accelerators are to be upgraded from the TRISTAN project. It was recently decided to employ EPICS control software for the KEKB ring controls. It is reasonable to use such mature collaboration-based software when starting to build a new control system. However, on the linac side, which will be upgraded from 2.5 GeV to 8 GeV, we have been operating a new separate control system, employing international and de-facto standards with an object-oriented design for the last two years.

We are thus searching for a scheme to join these two control systems together, since it is important to have a tight control coordination between the linac and ring in order to achieve a higher luminosity in physics experiments. In this report, several schemes for the integration between the ring and the linac are discussed.

I. Introduction

Modification of the KEK electron/positron linac for the B-Physics (KEKB) project with an improved beam current and energy started in 1994 [1]. We will upgrade our linac energy from 2.5 GeV to 8 GeV by 1998. A control system is also being prepared for it employing international and de-facto standard systems with object-oriented design [2]. It will improve both functionality and reliability [3].

In the project the linac should provide beams of better quality, stability and availability in order to achieve a higher luminosity in the ring. Thus, cooperation is indispensable between the control systems at the KEKB ring and linac. Extensive studies must be carried out in order to understand the correlation between the operational parameters in both accelerators with common database and accelerator-analysis codes.

Recently it was decided to employ EPICS (Experimental Physics and Industrial Control System) [4], for the KEKB ring controls and the control software is being reconstructed while maintaining most of hardware resources from the TRISTAN project.

On the other hand, we have been gradually rejuvenating the linac system since we must continue beam injection to the Photon Factory (PF) ring during the improvement. It is reasonable to employ EPICS as a result of the international software-sharing effort when a new control system is designed. However, the issue of integration with the existing linac control system remains.

In the former TRISTAN project, accelerator operation was performed separately at the linac and ring control rooms. Although some control information could be exchanged concerning the last part of the linac, it was not much used during normal operation. In the KEKB project, it is necessary to merge at least the beam handling console.

It is very desirable to have a common control architecture. Fortunately, we use a similar hardware architecture in both of the new control systems. There should be possible a scheme to shift over toward common EPICS controls.

In this paper we consider the technical aspects of the integration feasibility of the existing linac system and the ring EPICS system. Operational aspects will be considered elsewhere.

II. Control System Cooperation Between the Linac and Ring

In order to achieve system cooperation between the KEKB linac and ring, the operational procedure and use of the control-room are under discussion. From a technical point of view, there are several possibilities. The use of EPICS at the linac is of course one of them. Below we describe four major possibilities.

A. Employment of EPICS at Linac

In order to integrate two control systems the best solution would be to use EPICS at the linac control system replacing the existing one, since common resources could be shared between the two control systems without much effort. This scheme would be taken if both control systems were to be designed again.

*Also at CERN/PS during 1994. furukawa@kek.jp.

However, because of the need for PF ring injection we cannot shut down the linac accelerator for a long period during the KEKB upgrade and much of our manpower will be necessary to follow a gradual accelerator improvement program. It is thus quite difficult to replace software during this period.

Although the equipment access method over the network was standardized at the linac, we have used three different methods for the graphical operator interface: MS-Windows GUI applications, DOS-based touch panel systems and Tk [5] based X-Window GUI applications. Because of such a variety it would take some time to re-design the graphical operator interface under EPICS.

Also, the equipment-control software based on object-oriented design at the linac does not match the channel-oriented EPICS software, so it would be necessary to redesign the structure of most application software for controlling the linac.

In order to accommodate the differences between equipment access methods it is possible to develop special EPICS records, such as magnets and klystrons. Although we may emulate our current access method on EPICS IOC (Input Output Controller), those pieces of software would become large, and we would have to modify not only the client applications, but also the EPICS source tree.

If we change to an EPICS environment, it would be attractive to implement such records to fulfill the equipment-oriented design at the IOC level. However, during the construction time, such a complication should be avoided and a more simple scheme should be implemented.

B. EPICS Capability in the Current Linac Control System

The current linac-control system comprises VME front-end systems, Unix systems and operator interface subsystems. If it would be possible to implement the EPICS IOC capability into the current VME systems, both the current linac software and EPICS client software could be executed and a gradual transition to EPICS could be accomplished.

One of the authors (k.f) studied such a feasibility [7]. As a result, the idea came up that if we could emulate some VxWorks system calls, which EPICS IOC utilizes, in a thread environment, we should be able to execute EPICS IOC codes, although dynamic symbol manipulation is technically not easy to implement.

On the linac VME, while the OS9 operating system is currently employed, most software is written for both the OS9 and LynxOS operating systems. We had a plan to move to LynxOS, which is currently suspended. With LynxOS we could use the POSIX thread (pthread) environment, which is one of the standard thread specifications. Though the code is not yet finished, an implementation has been attempted on LynxOS.

Therefore, such transition is possible. However the software development required is too much compared to the amount of sharable resources. Since the transition from OS9 to LynxOS has been suspended for certain reasons, this scheme is not feasible now.

It would be interesting to adopt the POSIX real-time standards to the EPICS IOC, since it would expand the EPICS application area. It should be discussed not only for KEKB integration, but also for more general areas.

C. Communication Protocol Conversion

In the linac control system we implemented our own control protocols both between Unix and VME and between Unix and the operator interfaces. On the other hand, in EPICS the Channel Access protocol is used between the IOC and OPI. If we could convert one of these protocols into the other, the two control systems could communicate with each other.

A server for EPICS Channel Access, called Portable CA server [8], is being developed in the EPICS collaboration, in order to enable access to external controls from EPICS Channel Access clients. Since several sites use it without any obvious problems, the Portable CA server could be installed at the linac in order to allow EPICS clients to reach the linac controls.

After writing protocol conversion routines for the linac equipment, EPICS Channel Access clients could see the linac accelerator as one large IOC. Although it is asymmetric as shown in Figure 1 and we could not share low-level controls between the linac and ring, such a solution would be very simple to implement.

In such a scheme, if it is necessary to reach an IOC in the ring side, from the linac, Channel Access library routines could be called from inside the linac operator interface. At least at the beginning, some kind of security checking should be made between the linac and ring controls.

D. Common Upper Level Control Protocol

A Channel Access server described in the previous section is a practical solution in a one-to-one conversion. However, we must soon think about the Photon Factory ring as one of the downstream clients for the linac. Also, controls for physics experimental groups and other facilities may be loosely combined.

In such a multi-architecture situation it is natural to design a common upper-level protocol and several groups are working to implement it. The CICERO project is a development collaboration based on CORBA (Common Object

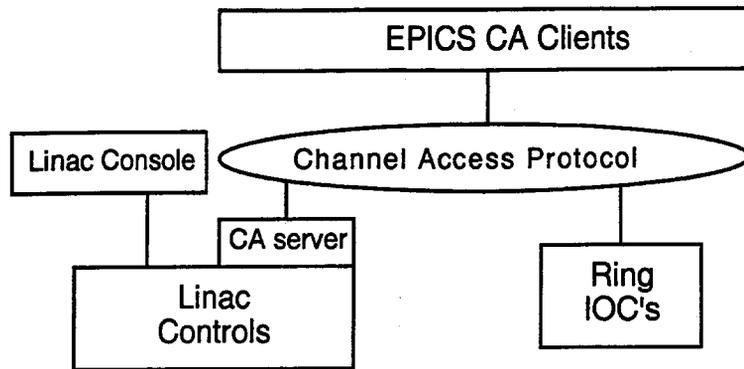


Figure. 1. CA server implementation at Linac to enable common applications.

Request Broker Architecture). In the EPICS collaboration cdev (control device) [9] is proposed as the common API and implementation is made available.

Since cdev is designed to be object-oriented, it should be easy to write an interface to the linac control protocol. Although the link at the top level with cdev is not efficient in speed, software design will be clear.

Although cdev over the distributed network is not yet defined well, we expect that ACE (Adaptive Communication Environment) or CORBA will be soon applied to cdev. Figure 2 shows such a common environment.

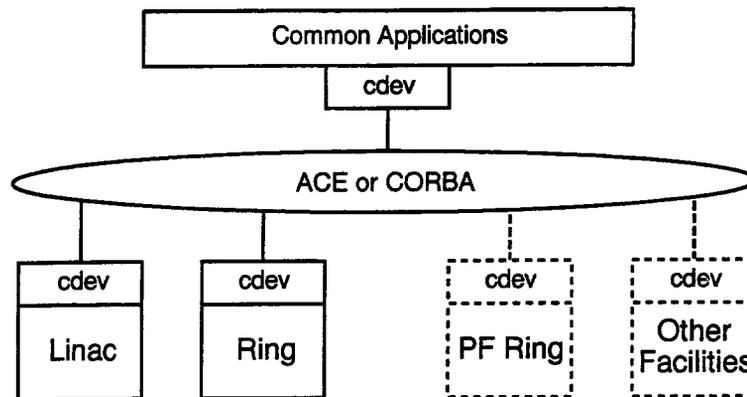


Figure. 2. Possible integration of control systems at KEKB with cdev and ACE or CORBA. Common Applications may communicate with any control system.

Although this scheme seems very suitable, and would provide a common application environment, the number of existing general application programs is small. Most of EPICS clients use only the CA protocol, but we hope that more application programs will be developed on cdev. In the long run this scheme is considered to be the most promising.

III. Other Services

Other services, such as computer resources, computer networks and relational databases, as well as their management, can be combined more easily than the control protocols. Since our manpower is very limited, we should pay attention to merge such services.

IV. Conclusions

The above discussion shows two schemes which are feasible for merging the existing linac control system and the EPICS ring control system. One is to employ the CA server at the top of linac controls for a short-term solution.

The other is to implement a cdev environment over the top of the two control systems as a long-range plan. We need to design a network-communication interface and a set of common control application programs in the EPICS collaboration.

If the adoption of cdev is successful, we may skip the use of EPICS on the linac side. Of course, in order to save manpower, we may design new sub-systems with EPICS. We would not have any problems with such a mixture of architectures, since cdev can manipulate all communication between them.

As a result of these considerations we hope to achieve successful results with the KEKB project.

V. Acknowledgment

One of the authors (K.F.) is deeply grateful to Dr. B. Kuiper for supporting this work during his stay at CERN, and to Mr. R. Dalesio for his EPICS support via Internet. The authors would like to thank the linac staff and control-group members of the KEKB ring for valuable discussions.

References

- [1] A. Enomoto et al., "Re-formation of the PF 2.5 GeV Linac to 8 GeV", proceedings of the 1994 International Linac Conference, Tsukuba, Japan, 1994, p. 184.
- [2] K. Furukawa et al., "Upgrade Plan for the Control System of the KEK *e.e.* Linac", Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems, Tsukuba, Japan, 1991, p. 89.
- [3] N. Kamikubota et al., "Improvements to Realize Higher Reliability on the KEK Linac Control System", these proceedings.
- [4] Leo R. Dalesio et al., "The experimental physics and industrial control system architecture: past, present, and future", Nucl. Instr. and Meth. A 352, 1994, p.179.
- [5] K. Furukawa et al., "Control System for a Bunch/Profile Monitor at the KEK *e.e.* Linac", Proceedings of the 1994 International Linac Conference, Tsukuba, Japan, 1994, p. 819.
- [6] T. Katoh et al., "Proposed Control Systems for the KEKB Rings", these proceedings.
- [7] Feasibility study of EPICS on LynxOS was done by K. Furukawa at CERN/PS.
- [8] J. Hill, "Portable CA server", private communication.
- [9] J. Chen et al., "An Object-Oriented Class Library for Developing Device Control Application", these Proceedings.
C. Watson et al., "Introduction to Cdev", 1995.

A CONTROL SYSTEM UPGRADE OF THE SPEAR SYNCHROTRON AND INJECTOR*

R. Garrett, S. Howry, C. Wermelskirchen, J. Yang

*Stanford Synchrotron Radiation Laboratory
Stanford Linear Accelerator Center
2575 Sand Hill Road
Menlo Park, CA 94025*

ABSTRACT

The SPEAR electron synchrotron is an old and venerable facility with a history of great physics. When this storage ring was converted to serve as a full-time synchrotron light source, it was evident that the time was due for an overhaul of the control system. Outdated hardware interfaces, custom operator interfaces and the control computer itself were replaced with off-the-shelf distributed intelligent controllers and networked X-workstations. However, almost all applications and control functions were retained by simply rewriting the layer of software closest to each new device. The success of this upgrade prompted us to do a similar upgrade of our Injector system. Although the Injector was already running an X-windows-based control system, it was not networked and Q-bus-based. By using the same Ethernet-based controllers that were used at SPEAR, we were able to integrate the two systems into one that resembles the 'standard model' for control systems, and at the same time preserve the applications software that has been developed over the years on both systems.

INTRODUCTION

SPEAR was originally commissioned in the early 1970's as an electron-positron collider, and, when the PEP ring was built at SLAC, the control system at SPEAR was upgraded by installing a version of the PEP system [1]. The PEP system used custom-built CAMAC hardware which had a parallel crate architecture. This system used a Grinnell graphics processor to provide the operations displays and the background display for the touch-panel controls. This processor provided primitive graphics and text with color capability. The PEP system also used Modcomp computers to enable distributed control to the CAMAC crates due to the physical separation of the control room from the various parts of the PEP ring. The Modcomp system was not required for SPEAR because of its more compact physical layout. The port of the PEP system to SPEAR was done in the early 1980s and ran the SPEAR ring for more than ten years using VAX 11/750 and 11/780 computers. In 1990, the SPEAR ring became a dedicated synchrotron radiation source operated by the Stanford Synchrotron Radiation Laboratory.

The SPEAR Injector was built in 1989 and used a control system that was transferred from a system developed at the ELSA ring in Bonn, Germany [2].

REQUIREMENTS AND OPTIONS

Our motivation for doing the upgrade of SPEAR (and then using that experience to upgrade the Injector as well), was to replace a couple of very old systems which were either already obsolete and unsupported, or soon would be. The two systems that were of most concern were the VAX 11/780 that ran the control system software and provided the UNIBUS interface to the CAMAC hardware and the Grinnell graphics display processor.

The 11/780 was already obsolete and very expensive (\$15k/year) to maintain and was an obvious candidate for replacement. The problem was that the current interface to the CAMAC hardware was via a proprietary UNIBUS-based controller called the VAX CAMAC Channel (VCC) [3], a SLAC built device. Since none of the modern computer systems currently available supported the UNIBUS, it was going to be necessary to replace the VCC as part of the upgrade.

* Supported in part by the Department of Energy, Office of Basic Energy Sciences, High Energy and Nuclear Physics under Department of Energy contract DE-AC03-76SF0015

The Grinnell graphics processor drove the operator displays and the touch-panels. It provided only primitive graphics capability and the manufacturer of the hardware had gone out of business many years previously, making maintenance difficult as spare parts became scarce.

Our basic requirements for replacing the existing system were to:

- Provide CAMAC performance no worse than the current system which operated at about 3 Hz.
- Provide an X-windows-based graphical user interface (GUI).
- Continue to provide a capability to knob operator-selected parameters.
- Eliminate the VAX 11/780 and Grinnell processor.

Several proposals were submitted on how to meet these requirements and a design review was held in November 1992. The proposals submitted were to:

- 1) Obtain a VAXBI based computer and use a specially built fiber optic interface (which was similar to units being built for the SLC project at SLAC) to connect to the CAMAC crates.
- 2) Expand and install the control system used at our Injector to SPEAR.
- 3) Replace the current VCC and crate controllers with VCC emulation software and MicroVAX crate controllers from Kinetic Systems.

All proposals involved replacing the Grinnell system by converting the Grinnell graphics calls to X-windows equivalents and replacing the old touch-panels and color monitors with X-terminals.

The result of the design review was to choose the MicroVAX controller option.

THE UPGRADES

Hardware -- Crate controllers

We replaced the outdated SPEAR VCC interface and aging parallel crate controller with an Ethernet-based distributed intelligent controller. The advantage was that the Ethernet solution leaves the SPEAR control system open to the option of non-CAMAC hardware such as VME.

The Kinetic Systems model 3968 was adopted for both SPEAR and the Injector. The controller features a 2.7 VUP rtVAX 300 processor including both floating-point and advanced Ethernet co-processors, as well as 8 megabytes of RAM memory, two terminal ports, a CAMAC interface, and 8 kilobytes of dual-ported memory.

Hardware -- Knob boxes

The existing knob boxes at SPEAR were home-built and old, and the person responsible for maintaining them was retiring. They connected to the existing VAX system via an RS-232 link. After some searching, only one commercial unit was found that could provide similar capability and this was a knob box manufactured by Hytec.

The Hytec unit has only two knobs, as opposed to the four knobs on the old boxes, but is programmable in such a manner as to actually provide the capability of controlling twice as many parameters from one box as compared to the old knob boxes. These new units also use an RS-232 link, but are connected to an Ethernet based terminal server instead of directly to a VAX.

Hardware -- Computers and X-terminals

Once the decision to go over to an Ethernet-based control system, it allowed us to choose virtually any computer system that could run VMS to replace the VAX 11/780. Given our budget constraints, it was decided to get a VAX station 4000-90 system with two 1 GB hard disks and 128 MB of memory. This system provides more than 30 times the raw CPU power of the 11/780 and gives us the ability to do more compute-intensive activities locally, such as machine modeling, that would have been extremely slow or impossible to do previously.

It was also decided to provide several operator stations in the control room area using Tektronix X-terminals. The main operator station has an X-terminal with dual display capability, allowing the operator to display much more information in one place at one time. These units are all Ethernet-based.

As mentioned previously, the original SPEAR control system was built on top of VCC hardware that was interfaced to the CAMAC crates via a parallel data link and was connected to the VAX UNIBUS via the VCC (see figure 1).

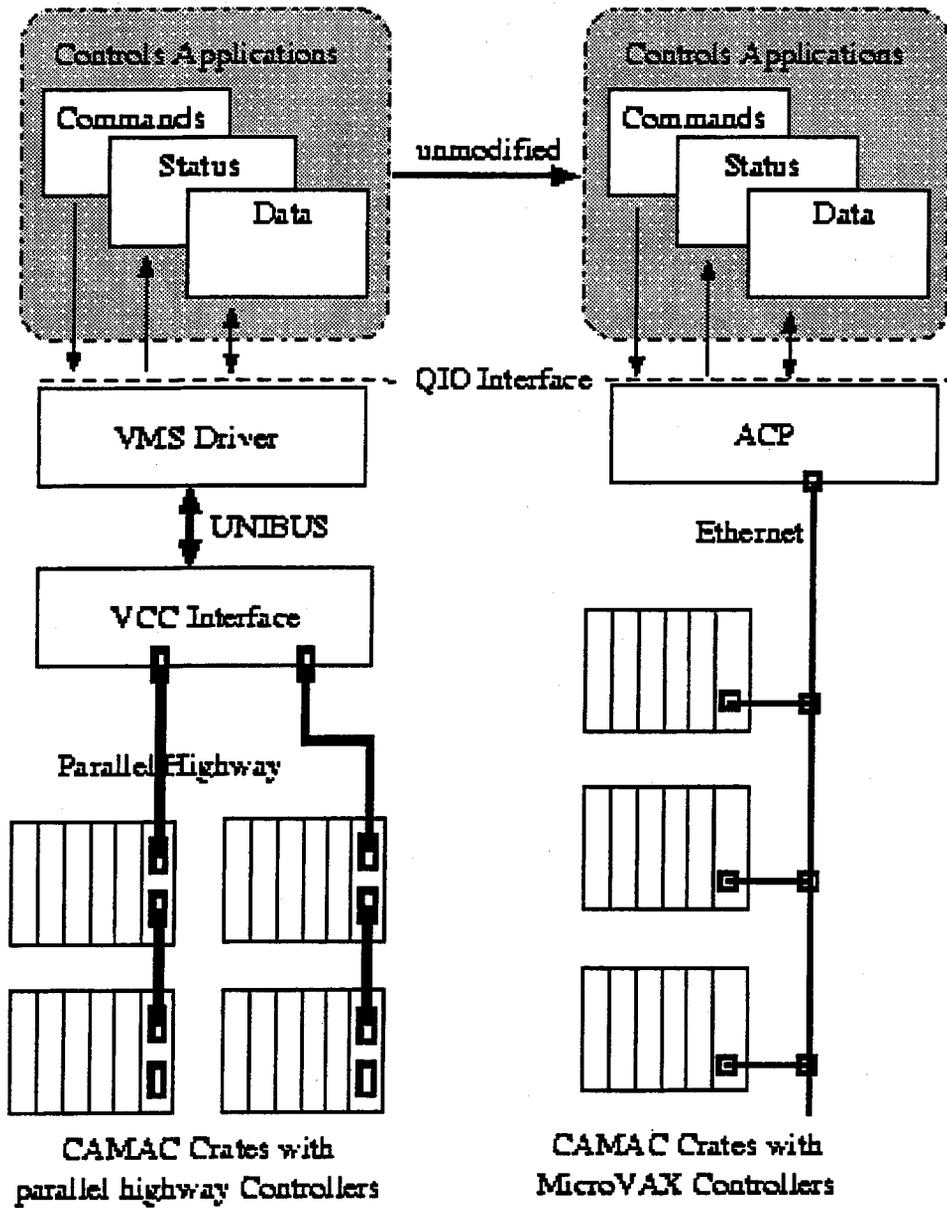


Figure 1
Schematic of Old and New CAMAC Systems

A VMS driver provided the operating system interface for the application software. All programs used this QIO device interface to execute CAMAC operations. The parallel data highway was organized in several branches (0 to 7), each branch connecting up to 8 crates (0 to 7). Each CAMAC crate was addressed by using the branch and crate number. An important decision for the upgrade project from the beginning was that the new CAMAC interconnect should keep the existing control application programs as they were, without any modification to the existing code. This was possible with the use of the well-defined QIO interface and provision of an identical device interface. For many years, SSRL ran application programs for the beam-line controls that used the capability of VMS to have an Ancillary Control Process (ACP). Such a process provides a QIO interface for application programs and handles input / output requests for these programs within a regular program [4]. This interface was chosen for the upgrade project because it yielded the flexibility for the new software to emulate the old CAMAC functions on the main host. The additional software overhead was tolerable as the new SPEAR control host (VAX station 4000-90) is about a factor of 30 faster than the old VAX 11/780.

The ACP on the central host now establishes network connections to all crate controllers and thus executes the CAMAC instructions. In more detail the CAMAC system now works as follows:

- An (untouched) application program generates a list of CAMAC commands, provides a data buffer for write and read data as well as a status buffer. This is done in exactly the same format as for the old VCC system.
- The application issues a QIO and passed these buffers to the operating system.
- The ACP gets these buffers.
- The ACP generates independent lists of CAMAC commands and write data for each crate addressed.
- If not done already, the ACP establishes a DECnet connection to each crate used.
- The ACP sends a packet containing commands and write data to each crate.
- The MicroVAX crate controllers receive the command and data packets.
- The crate controllers (in parallel) execute the given CAMAC commands, using the given write data.
- The crate controllers each build a list of status information and read data.
- Each crate controller sends a status and data packet back to the ACP.
- The ACP receives the status and data packets from the crate controllers.
- The ACP returns the status information and data to the application program's buffers.
- The QIO operation is complete.

As described, this new system fulfills the requirement of replacing the old hardware system without any modification to the existing application software. The critical aspect of this project was the real-time performance of the new system. This was difficult to estimate as the number of CAMAC commands per CAMAC list varies from a single command to up to 100 commands. At the same time, the commands in a CAMAC list can address an individual CAMAC crate or all crates at once. When first implemented, the ACP sent the command packet to the first involved CAMAC crate, waited for the response and only then sent the command packet to the next involved crate. As expected, the performance was dramatically improved after a revision, so that now all packets are sent and then the status and data packets are received. A measurement of the worst-case scenario, i.e., a single CAMAC command in a list, shows a total round-trip execution time of about 7 milliseconds. This is the minimum response time, but additional commands in the same list increase this time by very little. In the running system the ACP now collects statistical information about the data throughput to each CAMAC crate. This indicates that the average execution time for a SPEAR CAMAC operation is about 9 milliseconds. The performance of the Ethernet was also taken into account. To be independent of the global Ethernet traffic, the control system segment was isolated from the rest of the network by a LAN bridge. The current SPEAR machine operation requires that the injected and stored electron beam be accelerated from 2.3 to 3.0 GeV. This is done over a time of about 3 minutes and it requires that all the main and corrector power supplies are ramped in a very synchronized procedure. In the old VCC hardware system this was guaranteed by the VCC hardware. In the new system, the commands for the different CAMAC crates are handled more independently by the ACP and there exists a potential for variations in the millisecond range. However, this problem has not been experienced.

Displays on the old control system were driven by a 1970's era video controller from Grinnell. This device accepted 16-bit commands to set position, draw pixels, lines, rectangles, and to manage a color lookup table that depended on the division of available planes among the different screens. Ours was configured to two black and white touch-panel screens and three 4-plane color display screens. A 'switch channel' command allowed programmers to send commands to selected screens. The device was shared, allowing (in principle) any application to write to any or all screens.

The replacement of this device was greatly facilitated by the modular, object-oriented flavor of the original control system software. In this design, all applications were event driven, quickly-executed 'methods' organized into

concurrently executing VAX/VMS processes (APs). Each received event always contained the screen number(s) in its auxiliary data, so applications never had to make this decision. Events are scheduled at selected frequencies, or by operator touch-panel button pushes, or by something happening in the storage ring.

Further, each application communicates to the displays through a 'graphics layer'. This layer generates 16-bit commands from requests to draw standard (but limited) graphics items. The protocol is:

- initialize (screen#, buffer);
- draw_item#1(data, buffer);
- ...
- draw_item#2(data, buffer);
- ...
- flush(buffer);
- return.

At 'flush' time, the buffer of commands was sent to the device as a single I/O operation (the layer automatically generates extra I/O operations if the buffer becomes full). The graphics layer is linked to each AP containing an application that requires graphics.

To replace the video controller device:

- Each screen of the device was replaced by an X-window client running on an X-server. The client process, which can also reside at any network node, receives buffers of 16-bit commands and converts them into Motif requests. It also maintains backing structures to redraw the window when requested to do so by the server. The structures are necessarily 'AP semantics-independent', built up only from the packets of 16-bit commands received since the last 'clear_screen' command.
- A 'dispatch' layer of software was inserted at the device end (where the I/O operation used to be invoked) of the graphics layer described above. This layer arranges for each buffer of 16-bit commands to be forwarded to the client process of the buffer's screen. The layer is linked to each AP that needs the graphics layer.
- No modifications were made to any of the AP's (but of course they had to be re-linked).
- A possible problem was the proliferation of network sockets if each AP could send to each client. We decided to reduce this by requiring the AP's dispatch software to forward buffers to the 'master scheduler' process (the one that also manages the touch-panels; there can only be one of these at a node). This process, is the only one that talks over the network to the X-clients.
- Since things are executing asynchronously on many nodes, ring buffers had to be installed at several places. The most important spot is where buffers entered the clients, particularly if there are slow 'dumb X-terminals' installed as X-servers on the system. Another spot is where buffers from applications are received by the master-scheduler process.
- On the input side, physical touch-panels are not easily available, so button pushes are replaced with mouse (release) clicks. These mouse clicks from the X-server's screen are received by the client and sent (over the network if necessary) to the Touch-panel Manager, wrapped to look like button pushes from the old touchpanel screens.

The new system automatically provides the following enhancements:

All screens can have color. At present 16 colors are available and deemed sufficient. The screen windows can be resized by the Motif window manager. Fonts look much better since the old device didn't even support lower case(!).

The number of X-servers executing at one time is limited only by the number of network sockets allowed. A single server can show windows of different screens. Two servers at different locations can monitor the action of the same screen.

The number of logical screens can be increased. Applications can be dedicated to screens, or they can be dynamically allocated to screens by the operator (an almost redundant feature, but it comes free!).

Software -- Injector

The successful upgrade of the SPEAR control system by using MicroVAX CAMAC crate controllers defined the upgrade for the Injector control system. The Injector also got rid of the CAMAC interface that was connected to the proprietary bus (Q-bus) of the MicroVAX 3600 on which the control system ran. All CAMAC crates -- for SPEAR and the Injector -- are now controlled by the same type of crate controller, thereby minimizing maintenance activities. These crates are connected through Ethernet, which eliminates dependence on a single manufacturer.

The Injector control system is much younger than the SPEAR system and therefore does not have an extensive history in the application programs. The design of the control system incorporated the feature of putting machine parameter values into the control system database only when the parameter values had changed. This was implemented by a data acquisition process that collected all input data from the CAMAC modules and applied the parameter values after filtering, i.e., comparing with previous values. The upgraded Injector control system now generates a configuration for each CAMAC crate from the control system database. This configuration is loaded into the crate controller CPU. The processor now runs the data acquisition within the crate; it filters the data and only sends a data packet through the network to the central host for the parameters that have changed. This significantly reduces the network utilization.

As the CAMAC interconnection was already isolated from the application programs before the upgrade, the change in data management within the crate controllers did not affect the existing programs. By replacing the shareable library that addresses the CAMAC hardware, the control applications now "talk" to the crates through a network instead of the serial highway. The only other change made to the Injector was forced upon us when we ported the control system from the MicroVAX 3600 to a (somewhat) newer VAXstation 3138. The VAXstation was running a more recent version of VMS, necessitating a conversion of all the DECwindows routines to their equivalent Motif calls. The MicroVAX was showing signs of age and was becoming under-powered for the load we were putting on it.

POST-UPGRADE IMPROVEMENTS AND FUTURE

Alpha Port

As always, whenever you make new capabilities available in a system, people will always find a way to exploit those capabilities beyond the capacity of the system to support them. In an effort to stay ahead of the game and to remain in synchronism with computer technological trends, we plan to port the entire SPEAR control system to a DECstation 3600 AXP. Although the current SPEAR computer is more than adequate for our foreseeable needs, we also realize that it will take a significant amount of time to port all the various SPEAR applications to the Alpha architecture and wish to make that transition before we are forced to make it.

For most of the applications it will be mainly a matter of compiling and relinking the code. However, the CAMAC and database interface codes will most likely need more extensive rework in order to meet the requirements imposed by the 64-bit architecture and the larger page size.

Feedback / BPM Upgrade

In the previous SPEAR control system, the central host had to execute the very complex CAMAC command lists to perform the beam-position measurement (BPM). The BPM data acquisition was all done in a single CAMAC crate. This crate controlled RF multiplexers, which selected a single monitor button out of the 24×4 signals, started a BPM signal processor and digitized the signal. From the raw data values the central host then calculated calibrated beam positions. After the replacement of the old VCC CAMAC hardware, the process of the BPM data acquisition and data calibration was moved into the intelligent crate controller. The MicroVAX controller now handles the modules in the CAMAC crate by itself. The dedicated software performs essentially the same steps as the old central host, but it does this in a much faster manner. Beside generating the raw button data, it also generates calibrated beam positions. The calibration functions were actually linked into the MicroVAX crate controller code as they were before. This was made possible when existing VMS FORTRAN subroutines were linked with the new C code that runs under the VAXELN operating system in the crate controller. The migration of the BPM processing into the new MicroVAX crate controller resulted in the following improvements:

- The data acquisition time to measure a complete orbit was improved from one non-averaged measurement per minute to about 1 measurement per second, each averaged over more than 1000 orbit readings.
- The central host receives calibrated data much faster without any overhead.

SPEAR / Injector Merging

Both accelerators now have identical CAMAC crate controllers. The functionality, that is, the type of commands from the central hosts, is still different. For the SPEAR system, lists of commands are exchanged and executed about 3 times per second. For the Injector system, the acquisition configuration is loaded once and the crates send data when needed. Of course, changes of parameter settings are possible at any time.

The two different software packages for SPEAR and Injector were merged together. Each CAMAC crate controller can now be addressed by the SPEAR control system as well as by the Injector system. To date this feature is not fully utilized. Care has to be taken that control settings have a single-master system.

VME Integration

Recently, a VME crate was added to the SPEAR control system. This crate will eventually be used for a feedback system [5]. As a migration path for the future, this crate is now housing DAC modules that control some SPEAR power supplies. The VME crate also has a MicroVAX controller that is connected to the SPEAR network. The software that runs in this controller emulates a CAMAC crate for the SPEAR control system and thus completely hides the existence of a VME crate from the SPEAR system.

In the future, more applications will be moved into the front-end crates. This will allow the central control systems to concentrate on more global management tasks as well as the operator interface. In the long-term, the different control systems will merge, as different individual systems can be seen as parts of a global system that are all interconnected and exchange data with each other.

CDEV Compatibility

The CDEV application programming interface (API) being developed at CEBAF and APS appears to have the potential to make the sharing of accelerator programs between various facilities much simpler. We would like to take advantage of this capability by modifying our control system so that it can transmit and receive data at the CDEV layer. We believe this can be accomplished with only about a one to two man-months effort.

CONCLUSION

The new SPEAR control system has been running now for about 2 years and has been very reliable. We have seen no evidence of any network bottlenecks or other kinds of problems associated with Ethernet communications. We continue to make minor adjustments to the crate software to improve performance and to support new functions.

The Injector control upgrade is in the checkout stage at the time of writing, but is essentially fully operational and appears to be working quite well. We are using this system as we are starting up for our next user run, with the intention of operating with the new system for that run.

ACKNOWLEDGMENTS

Thanks to Max Cornacchia, Heinz-Dieter Nuhn, Jeff Corbett, Ed Guerra and all the other operations and support personnel at SSRL who helped to make this project a success.

References:

1. S. Howry et al., "A portable database driven control system for SPEAR," SLAC Publication 3618, April 1985
2. C. Wermelskirchen et al., "The SSRL injector control system," *Proceedings of the 1991 IEEE Particle Accelerator Conference*, 1991
3. D. Nelson, M. Briedenbach, et al., "The VAX CAMAC channel" *IEEE Transactions on Nuclear Science*, NS-28, 1981
4. A.D.Cox, "The SSRL beamline data acquisition systems", *Rev. Sci. Instrum.* 63(1), 854-855, 1992 (Special issue: *Proceedings of the 4th International Conference on Synchrotron Radiation Instrumentation*, 15-19 July 1991, Chester, UK)
5. R. Hettel et al., "Digital orbit feedback control for SPEAR," *Proceedings of the 1995 IEEE Particle Accelerator Conference*, 1995

Data Acquisition and Management in BEPC

X. Geng, J. Zhao, Y. Yan, Y. Yu, H. Luo, W. Liu

Institute of High Energy Physics, Chinese Academy of Sciences
P.O.Box 918-10, Beijing 100039, China
Internet: Suny@bec5.ihep.ac.cn

This paper describes the method of upgrading the BEPC control system, in which the dedicated adapter VAX-CAMAC-Channel (VCC) was replaced by a commercial adapter KSC2922/3922, Qbus CAMAC interface. All low level I/O driver routines have been changed without changing the whole CAMAC hardware system. The upgraded control system has a distributed architecture and several hierarchical databases are installed in the FEC computers, so the data flow should be controlled. Once raw data in any node have been refreshed, any changes will be transferred to the other nodes to maintain uniformity of data in those databases.

1. INTRODUCTION

The original BEPC control system has a centralized structure mainly composed of a console, a VAX750 computer, the intelligent channel VCC, the CAMAC system and hardware devices. The structure of the system is shown in figure 1. VCC is a dedicated product from SLAC, which is no longer produced and of which there is a shortage of spare parts. Now the VCC has been replaced by the commercial product KSC2922/3922, a Qbus-CAMAC adapter produced by Kinetic Systems Corporation, which serves as the data communication interface to the CAMAC hardware.

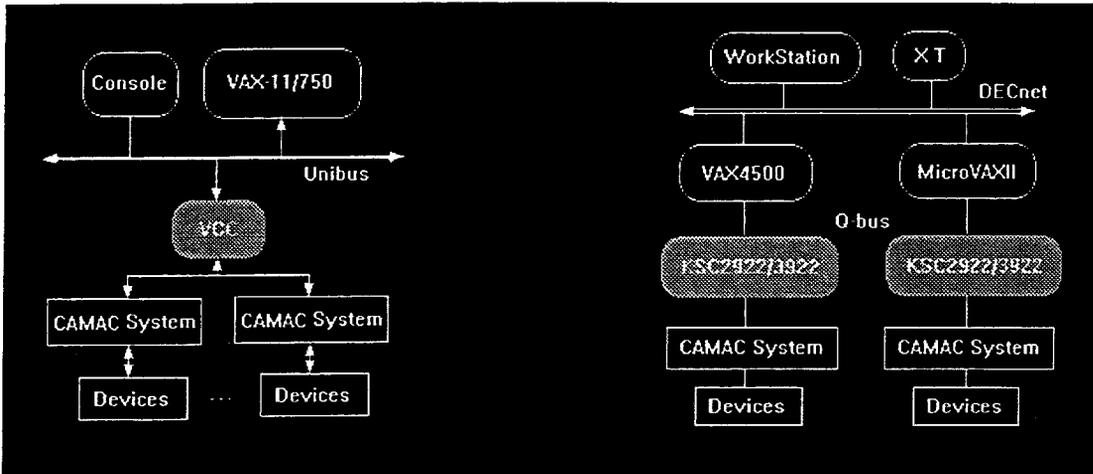


Figure 1 Original control system structure

Figure 2 Upgraded BEPC control system

The upgraded control system has a distributed architecture based on Ethernet and has been operational since Oct. 1994 (Figure 2). Because of the short time available for upgrading, the low level CAMAC system was retained.

2. HARDWARE STRUCTURE

The VAX-11/750 used the VCC as a Unibus - CAMAC interface, but the VAX4500 and MicroVAXII computers in the improved system use the Q-bus. The KSC 2922 Computer Bus Adapter provides an interface between the DEC Q-Bus and up to eight 3922 dedicated crate controllers through a byte-wide parallel bus. The 2922/3922 combination provides four DMA modes and a programmed transfer mode. All modes of operation are capable of transferring 16- or 24-bit CAMAC data words. DMA data rates up to 0.77 MB/s can be achieved.

3. SYSTEM SOFTWARE

Data acquisition software consists of a packet creation program PBZ, a data I/O program XCAMAC, a device on/off program DCOUT, the digital voltage acquisition program of the main B and Q magnet power supplies SPRDVM and the beam position monitor program BPM.

In the control system, there are nearly 7,000 signals which can be classed into 7 types. (see Table 1)

Table 1. The signals table

Input signals	DM (digital monitor)	1 bit digital input
	AM (analog monitor)	analog input
	DI (digital input)	16 bits digital input
	DV (digital voltage)	R*4 analog input
Output signals	DC (digital control)	1 bit digital output
	AC (analog control)	analog output
	DO (digital output)	16 bits digital output

For acquisition of the signals mentioned above, the following programs were rewritten:

The VCC packets are made by subroutine PBZ. PBZ takes the CAMAC I/O address of each signal from the database and assembles it to the control word and VCC packet. Then the control words and VCC packets are sent to the database to be used by XCAMAC and other processes. The sequence of the VCC packets in the database is: DMAMDI packets, AC_OUT packets, AC_IN packets, DCDO packets, IPSC packets and SAM packets. The BEPC system has about 1350 VCC packets.

The data acquisition process XCAMAC refreshes the database at a rate of twice a second and acquires about 4000 signals each time. The process also carries out the ramp operation of the magnet power supplies during particle acceleration in BEPC.

The subroutine SPRDVM acquires the digital voltage signals (DV) so that operators can monitor the present current status of the magnet power supplies. DVM3456A is connected to the VAX computer via a 3388 GPIB interface.

4. IMPROVEMENT

There was a hierarchical database in the original control system which had a static area and a dynamic area. In the static area, there was information about the CAMAC interface and machine parameters. The original data from the accelerator equipment was stored in the dynamic area which was refreshed at 2 HZ. For the distributed architecture of the new system, the original database had to be modified. First of all, we installed a database in each FEC computer with same data structure and created a 3922 packet area in the dynamic area of the databases. To keep the uniformity of the data records in these databases, a network communication program was developed to exchange the data between those databases and several new sections were inserted the dynamic area of each database to hold the raw data from other nodes which are refreshed once per second through the network. As shown in figure 3, when the database on node 1 receives the raw data from its input/output port, the network communication manager is notified by a event flag to fetch the data and send them to the database in node 2. In order to prevent alteration of the high level application program, the 3922 data area in the dynamic area is mapped onto the original VCC data area and the local index of the database is replaced by a new global index in each database at control system startup, so that the raw data from all of the accelerator devices can be read in each node.

The format of the data and command packets differs from that for the VCC; therefore the main work was in changing the packet chains from VCC format to 3922 format. Another difference is the data bit format. The VCC requires the 16 high bits to be valid, but the 2922 needs the 16 low bits.

A new packeting organization program QPBZ acquires the CAMAC I/O address of every signal from the database, assembles them to CAMAC control words by calling the 3922 driver subroutines such as Cainit, Caopen, Caclos, Canaf, Cainaf, Cablk, Cahalt, Caexew and Camsg, etc. Since block transfer operation is need for acquiring the analog signals by the SAM modules, we wrote a new program for the organization of SAM packets.