

Control System of the SPring-8 Storage Ring

R. Tanaka, S. Fujiwara, T. Fukui, T. Masuda,
A. Taketani, A. Yamashita, T. Wada and W. Xu

SPring-8, Kamigori, Ako-gun, Hyogo 678-12, Japan

The SPring-8 storage ring is under construction and will be commissioned in February 1997. The design for the control system of the storage ring adopted the so-called Standard Model concept. VMEbus is used as the front-end control system and is widely distributed around the 1436m circumference of the storage ring. The VMEbus is controlled by a single-board-computer with a RISC processor. The VME crate is connected to the FDDI backbone network through an FDDI-Ethernet switching hub. The CPU boards use the LynxOS-based HP-RT real-time operating system. The remote I/O system (RIO), interconnected by optical fibers, is used as the field bus. The operator console system consists of UNIX workstations with a human-oriented interface built using a Motif-based GUI builder. The control software design is based on the event-driven client-server scheme. The remote procedure call (RPC) is used for communication between machine control applications over the network. Accelerator operation history and equipment parameters are stored in a standard database, using an RDBMS.

1. Introduction

The SPring-8 project is to construct a large scale advanced synchrotron radiation facility and to promote fundamental science in the field of synchrotron radiation research. The facility being constructed at Harima Science Garden City in western Japan consists of a 1 GeV linac, an 8 GeV booster synchrotron and an 8 GeV low emittance storage ring. The storage ring has 38 beamlines from insertion devices and 23 from bending magnets, including eight beamlines 300m long and three of 1000m. Long beamlines are available for medical experiments planned for the future. Furthermore, SPring-8 is reserving the space of four long straight sections in the ring for future options. The construction of the storage ring is proceeding towards commissioning, which is scheduled for February 1997 [1].

2. Storage ring control

2.1 Concept

The design concepts of the SPring-8 control system are as follows:

- (1) distributed processors are linked with each other by a high-speed network,
- (2) sub-system (linac, synchrotron, and storage ring) control computers are loosely coupled, due to the different accelerator construction schedules and for the convenience of independent operation,
- (3) all the accelerators are operated from one control room by a small number of operators,
- (4) the system is built using industry standard hardware and software as much as possible,
- (5) for high productivity of application programs, computer-aided program development tools are used.

2.2 Architecture

We designed the SPring-8 control system based on the above concept. Figure 1 shows the control system architecture. It consists of UNIX engineering workstations (EWS), VMEbus systems, and optical fiber network (FDDI) systems [2]. In the storage-ring control system the network structure is designed to be flat. The remote I/O systems (RIO) are used mainly as the field bus system [3]. We use HP 9000/700 series EWS with the HP-UX operating system as operator consoles and HP J200 for the database servers. The CPU board of the VME system is HP9000/743rt with a RISC processor. The real-time OS HP-RT was chosen for the CPU board. The control software is designed based on the event-driven client-server scheme. Machine control programs are all written in the C language. The remote procedure call (ONC/RPC) is used for communication between machine control applications over the network. Accelerator operation history and equipment parameters are stored, using Sybase as the RDBMS. A detailed description is given in the following sections.

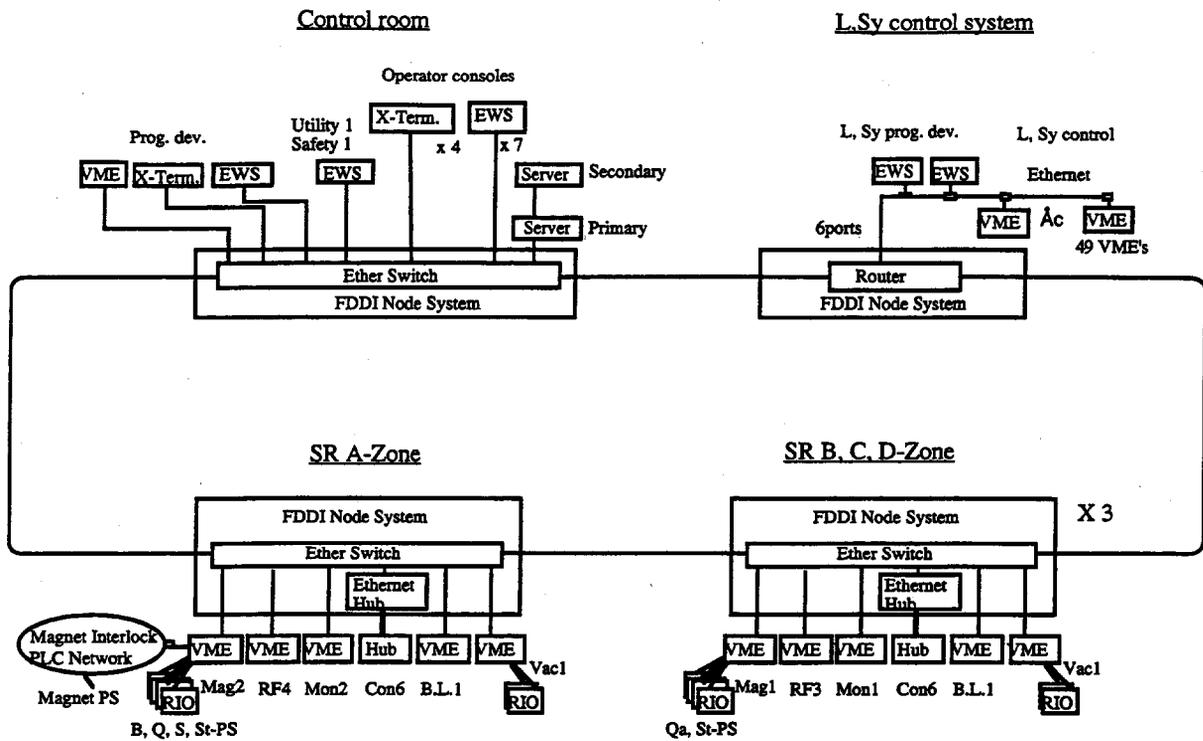


Fig. 1. The architecture of the control system.

3. Hardware and software

3.1 VME system

Expandability and maintainability are the most important criteria for adopting industry standard hardware. Consequently the VMEbus system was chosen for the front-end control system, as shown in Fig. 2. The interspersed VME system is suitable for the widely distributed control environment of a large storage ring. The CPU board HP9000/743rt/64, powered by the PA-RISC 7100LC processor, provides quite satisfactory performance (77.7MIPS). It has 16MB of memory and a 20MB PCMCIA flash ROM card, which is used as a boot device for the operating system. VME I/O modules such as DI, DO, TTL DI/O, AI, and PTG are used for direct control of RF equipment. The remote I/O (RIO) system is widely used as the field bus for magnet power supplies, BPMs, and vacuum system controllers. The GPIB bus is used as a field bus as well. We will use 28 VME systems for the storage ring control.

3.2 Front-end operating system

The selection criteria for the OS on the VME system are mainly real-time performance and open system features, indicating compliance with the POSIX standard. From among such OSs HP-RT was chosen. HP-RT is the LynxOS migration to the PA-RISC and can be called a real-time UNIX, so it is easy to understand and manage. The main difference between LynxOS and HP-RT is the program development environment, since HP-RT is supported only in a cross-development environment. HP-RT is running on the particular CPU boards developed by HP, so although availability maybe limited, we do expect excellent support from the company. This is a crucial point for reducing development cost and time for the new OS. HP-RT device drivers for VME modules are under development by the control group [4].

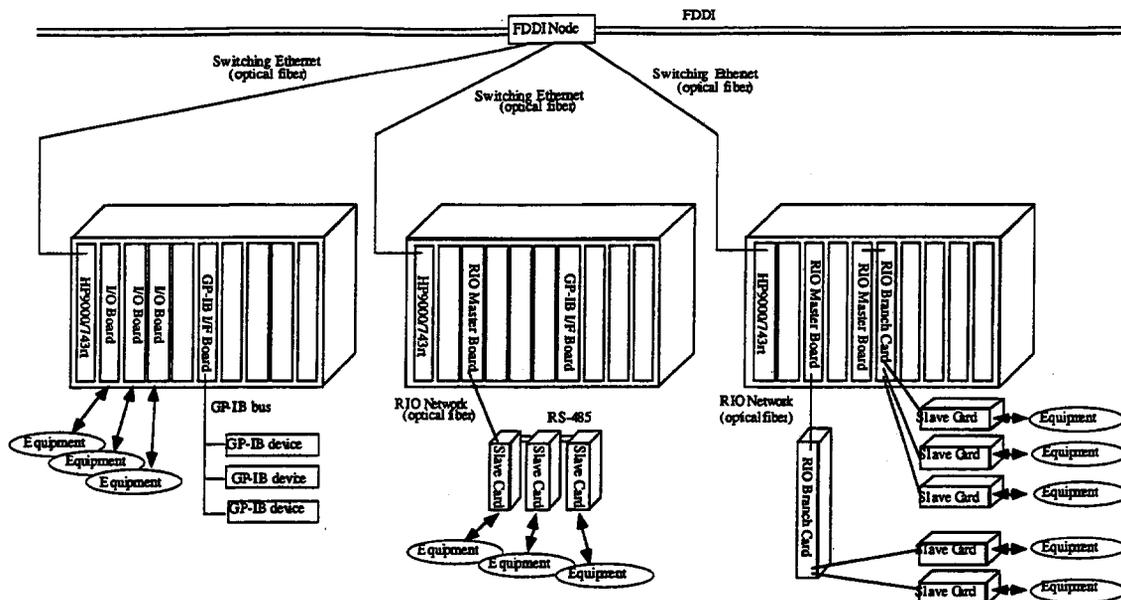


Fig. 2. The VMEbus control system.

3.3 Remote I/O bus

An optical fiber distributed remote I/O (RIO) system is used as the field bus for the VME system. The RIO system was initially developed for magnet power-supply control, and eventually will be applied to beam-position monitor and vacuum system control. The RIO system has good electrical isolation and is robust against noise. The system consists of; a) master module with dual port memory, b) several types of slave cards, c) 8-port branch card, and d) star connection with optical fiber cables between master module and slave cards. The features of RIO system are as follows,

- optical fiber connection,
- serial link communication,
- up to 62 slave cards can be connected to one master module,
- 1 Mbps transmission rate,
- up to 1 km transmission distance,
- HDLC protocol,
- can be connected by twisted pair cable (RS-485).

Six types of slave cards are available (type-A, B, D, E, F and G), depending on the type of interface signals of the equipment.

3.4 Operator consoles and servers

Operator consoles consist of seven UNIX workstations. These are HP9000/712/60 workstations running HP-UX with 128MB memory and 2GB disk. Four more consoles are planned to be used as the operator consoles for injectors and monitoring of human safety and utilities. Consoles are connected to the FDDI network node through an FDDI-Ethernet switching hub. The GUI based application programs run on each console using the X11 protocol. The connection of consoles has to be considered carefully because unnecessary transactions on the FDDI backbone may degrade its performance. In our scheme, switching hubs filter transactions efficiently without leaking unnecessary communication onto the backbone. The secondary database server is an HP J200 which has dual PA-RISC7200 CPUs, 256MB memory, DAT storage system, CD-ROM juke box and 13GB disk array with RAID mechanism. The primary server will be much faster than the secondary one.

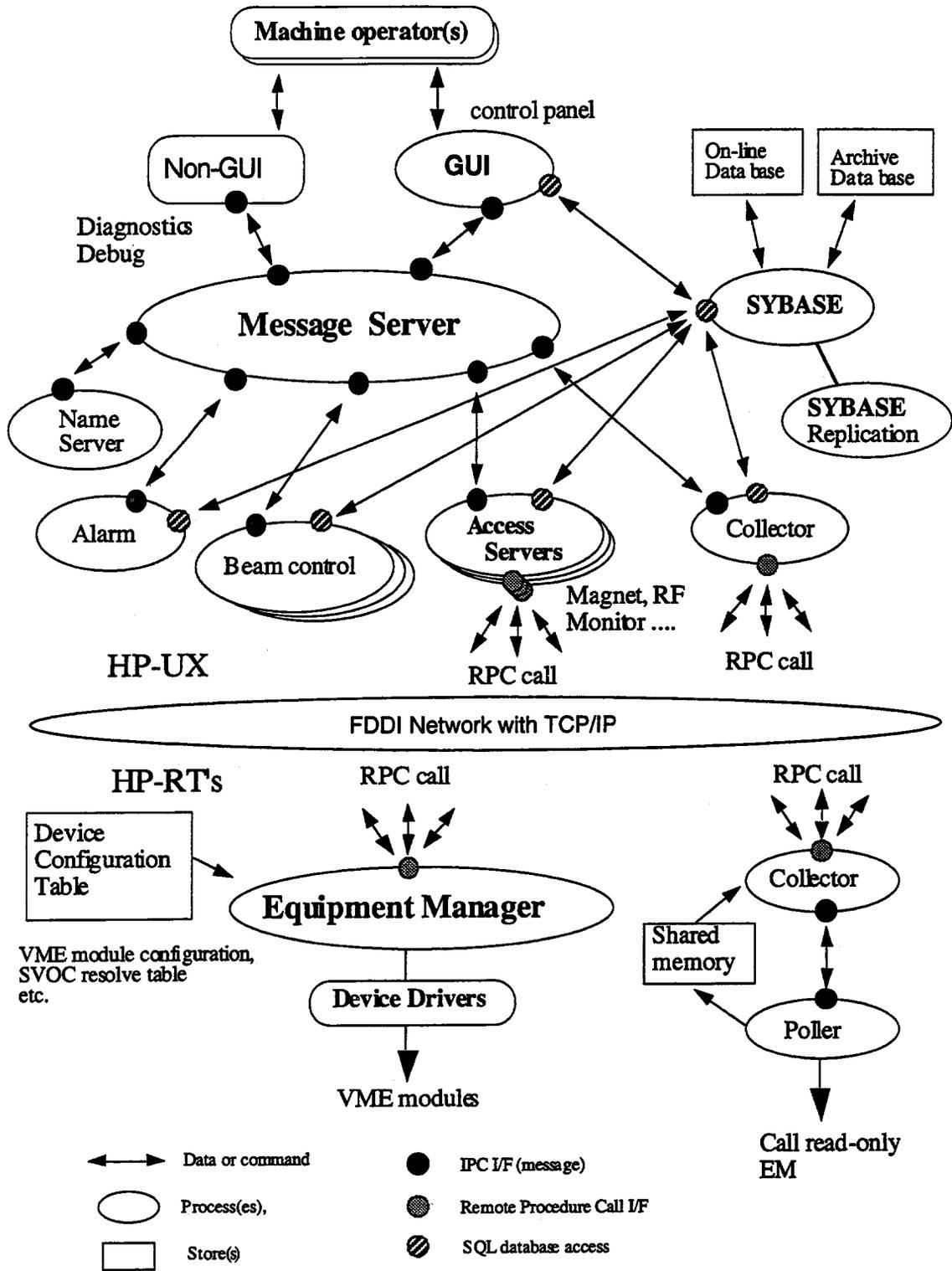


Fig. 3. The software structure of the storage ring control.

3.5 Human-machine interface

We are using a commercial GUI builder, X-Mate, for developing the operator interface. The look and feel of X-Mate is based on Motif 1.2 with X11 protocol. X-Mate does not necessarily require a window manager such as mwm because it has its own window system on top of Xlib. This is a good point for mission critical 24-hour operation because it is free from the memory leak problem of the Motif window manager. X-Mate provides a good development environment with a comfortable editor. Users are able to make widgets in WYSIWYG without knowing Motif programming. An interface part of the application program has to be written into the call-back routines.

4. Equipment access

4.1 Access scheme

Figure 3 shows the software scheme for equipment access. An operator command is created using the GUI and is sent out to the Message Server (MS), (see 6.1). The MS will forward the message to the Access Server (AS) after checking its syntax and access control status. The AS parses the message and resolves the destination of the message by asking the Name Server (NS). A composite sequence is analyzed in the AS, and messages are sent to the Equipment Manager (EM) running on the remote CPU board. Each equipment group, e.g. Magnet, RF, Vacuum etc., has its own AS. The synchronization of operations over different equipment is handled here with limited accuracy. The EM is a core process which manages VME devices connected to equipment controllers (see 4.2).

We have an additional cyclic data-acquisition path to get the equipment status. The Poller process running on the CPU board sends pre-registered messages to the read-only EM to get the equipment status. The data is stored in memory by the Poller and taken by the Collector server process on the same board; after collection the data is sent to the Collector client process on the operator console and finally saved to the on-line database by the DBMS, Sybase.

4.2 Equipment access

We designed a generic "Equipment Manager" control scheme based on the object-oriented concept [5]. The EM server process running on the front-end CPU board of the VMEbus realizes a device abstraction concept. It controls accelerator components directly without knowing the low-level physical device configuration. The operator commands are resolved by using control semantics and formed into a character string message like an English language syntax S/V/O/C, where the operator action is translated to a verb (V) and controlled equipment is into an object (O) and so on (see 4.3).

Once the EM receives the abstracted command (message), it is translated from logical equipment to physical device(s) by using the translation table. The translation table consists of a device configuration table and the attribute list of equipment. These tables keep the hardware addresses of VME modules and calibration constants etc. Tables are created by an off-line program at the higher level and downloaded to the EM at initialization. After translation, the resolved information of an operator command is passed to the VME device drivers and finally sent to the front-end equipment. The response from the equipment is also abstracted and put into the C of the S/V/O/C format. The reply is sent back to the machine operator through the AS and the MS.

The only way to access the accelerator components is through the EM. The Equipment Manager can be accessed by the Poller locally or by a control application program on the remote consoles by sending a command to the AS in the same manner.

4.3 Commands

The S/V/O/C control message consists of the subject (S), verb (V), object (O) and complement (C). The subject has process number, application name, account name, and the host name of the sender which issued the message. The information of the S is used for access control, exclusion control, and security check. The verb is a control action, the object is the equipment to be controlled, and the complement is a value to be written. All accelerator components are abstracted from the physical layer. Programmers of the AS and the Poller are required to know only the abstracted logical object (equipment) itself. The details of control channels and status bits are attached to the object as

attributes. The naming scheme for the controlled equipment is chosen to be suitable for beam control at the higher level. Each element and sub-element of the S/V/O/C message is delimited by the characters "/" and "_". For example, whenever an operator wants to know the current in a storage ring bending magnet, a character string "get/sr_mag_ps_B/current" is the message to be sent. The C returned from the EM can be like "123.45A" or "fail".

5. Network

5.1 Scheme

As can be seen in Fig. 1, the control system is connected to the duplicated FDDI backbone network with a transmission rate of 100Mbps. There are 6 FDDI nodes for the Spring-8 machine network. One is for the linac and the synchrotron and the others are for the storage ring. All nodes of the storage ring connect to the Ethernet/FDDI switching hub, which provides 10Mbps full bandwidth to each port. We use the optical fiber between the FDDI nodes and the VMEs in order to avoid magnetic interference. The FDDI optical fiber cables are laid around the maintenance corridor of the storage ring and to the injector building. The total length of the cable is about 4500m, and almost all the fibers are laid in a pipe from Air Blown Fiber System (ABFS). The ABFS has multiple pipes inside, and machine control uses one pipe for a bundle of four fibers.

The primary database server will be in the control room and directly connected to FDDI at that node. We will use the replication server system on the secondary database server which will interconnect the machine control network and the beamline user's network. The secondary database server open to users will be a gateway managing access to the control network by machine users.

5.2 Protocol

We selected TCP/IP for the network communication protocol and Simple Network Management Protocol (SNMP) for the network management. The TCP/IP protocol is reliable and provides good performance. Reliability is the crucial feature for machine control. The performance of TCP/IP is acceptable for storage ring control and its architecture is suitable because the configuration of our network is relatively simple.

6. Process communication

6.1 Message distribution

An operator control command is sent by a character string as a message, S/V/O/C as described in 4.1. A message is delivered to the Access Server through the Message Server by using the UNIX System V message mechanism. The MS plays a role as a message distributor. The syntax and access privilege of every message passing over the MS are checked here and all transactions are logged. Each application client running on the workstation has to get a queue number from the MS. Assigned queues are stored onto the Client Registration Table(CRT) managed by the MS. The CRT is used for monitoring the running clients, and its status can be used for surveillance of clients as well. The message distribution scheme is designed to allow only one MS on each console. In the first version, there is no communication between MSs running on different operator consoles; however it may be useful to allow communication between them to enable or disable (exclusion control) clients on different consoles.

6.2 Remote communication

The Remote Procedure Call (RPC) is selected for communication over the network because it makes the client-server model more powerful and provides easier programming in the low-level socket interface. There are available commercial implementations of RPCs such as the Sun's Open Network Computing (ONC) and the OSF's Distributed Computing Environment (DCE) based on the HP/Apollo NCS RPC. We selected the ONC/RPC because it is more widely available for the systems which support NFS.

7. Database

7.1 Management scheme

Data for storage ring control are stored in a unified way and accessed by every process running on consoles in the client-server mode. On-line and off-line applications, distributed in several machines, can store and retrieve the data to/from the database over the network with remote procedure calls (RPC). We chose a commercial relational database management system (RDBMS), Sybase SQL server 10.

Sybase was chosen for its on-line performance as well as its capability as a replication server. A replication server running on server machines can manage the consistency of the data on both the primary and secondary servers. Users of off-line applications, or those which do not affect accelerator operation, are only allowed to access the secondary server. This access control reduces the load on the primary server, which is already heavily loaded by the operational database management. The replication server system plays an important role as a backup server as well.

7.2 Accelerator database

The storage ring database manages three categories of data, constant parameters and on-line and historical data. The parameter database stores relatively stable data of equipment, i.e. calibration constants, physical dimensions, relationships between components. These are managed in the normalized database. The data for status of the accelerator are stored in the on-line database. The Poller/Collector collects current settings and readings of equipment and stores them into the on-line database. The Poller/Collector system is designed to be fast enough to be able to write about 4000 channels of data, which is approximately 10 KB, to the on-line database once per second. The on-line database is denormalized to a flat binary structure to gain performance in updating. The data which are needed for further off-line analysis are partially extracted from the on-line database and stored in the historical database with a time stamp. The archiving cycle of the historical database depends on the data. Alarm status and its history are also written in both the on-line and historical databases.

7.3 Data access

Data in the database are retrieved by issuing an RPC-based SQL request or by using Sybase client-libraries. The higher level processes running on consoles can be SQL clients. On the other hand, processes running on the VME CPU boards can not access the Sybase server directly because that functionality of HP-RT is not available yet. The NFS is one way to access text files on a remote node.

8. Status

The design stage of the control hardware has been finished and development is accelerating. The installation of the VME system for magnets and RF has started in the storage ring. A pipe for the air blown fiber system has been installed along the SPring-8 accelerators. We have finished the basic design of the control software, and development of some of the server processes has started using prototype or spiral prototype methods. The conceptual design of the database has started recently.

We have installed the prototype control software on the magnet power supplies and confirmed that the basic control scheme (GUI-MS-AS-EM and Sybase) works successfully on the real system. An operator controlled the power supplies by using the GUI panel, and the data taken by the EM was stored in the Sybase database.

9. Plans

The beamline control system is expected to be similar to that of the storage ring in hardware and software, but work has not started yet. The first 10 beamlines, including insertion devices, are planned to be in operation by 1998. It is necessary to finish the beamline control system soon and to fix a clear methodology for control of the insertion devices from the machine control point of view. Digital signal processors (DSP) may be useful for the fast local feedback system for beam orbit correction and image processing of the X-ray beam shape. Recently, the R & D for the DSP system has started.

10. Acknowledgments

We would like to express our appreciation to A. Gotz of ESRF, C. Serre of CERN, W. McDowell of ANL, K. Cahill of FNAL, S. Howry of SLAC, H. Nishimura of LBL, J. Urakawa of KEK and many colleagues of the above laboratories for useful discussions about the ideas for the control system of the storage ring.

References

- [1] M. Hara et al., Proc. 4th European Particle Accelerator Conference, London, June 1994, p597.
- [2] T. Wada et al., Proc. of the Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Tsukuba, Japan, November 1991, p151.
- [3] H. Takebe et al., Proc. of the 4th European Particle Accelerator Conf., London, June 1994, p1827.
- [4] T. Masuda et al., these Proceedings (ICALEPCS'95, Chicago, USA, 1995).
- [5] A. Taketani et al., these Proceedings (ICALEPCS'95, Chicago, USA, 1995).

Family of Smart Devices on the Basis of DSP for Accelerator Control.

A.S. Chepurnov, A.A. Dorokhin, K.A. Gudkov, V.E. Mnuskin, A.V. Shumakov

Institute of Nuclear Physics, Moscow State University.

119899, Moscow, Russia,

chas@rtm-cs.npi.msu.su, (095)939-56-59

Abstract.

There are many modern technologies of electronic equipment design which ensure quick and cost effective solutions for the object-dependent level of hierarchical computer control systems. On the basis of the MIL-STD-1553B fieldbus, digital signal processors (DSP), reconfigurable FPGA and RISC microcontrollers a new family of devices has been designed. Devices can be connected in different configurations via MIL-STD-1553B fieldbus, RS-485 and RS-232 interfaces. The intellectual ability and functionality of the family members depend on concrete applications. Complex control algorithms, including direct digital feedback control and data processing, can be implemented by using a powerful fixed-point DSP. Distributed control and data acquisition systems could be developed based on this device set. These devices are placed in stand-alone small boxes. The main spheres of application are control systems of large experimental installations, data acquisition, "hard" real time systems and industrial control. Modules have been used for industrial accelerator control and for upgrading a CW racetrack microtron computer control system.

INTRODUCTION.

The idea to generate the family of devices described below appeared as a result of 10 years work by the authors in the field of designing different control systems for industrial and scientific applications and instrumentation.

The basic principles of construction have been chosen - concrete fields of application for the devices; general architectural principles appropriated for different control tasks; modern microelectronics component base among the dramatically large number presented today on the market; tools for component interconnection; hardware and software architecture.

During the last one and a half years this work has been implemented and the first "members of the family" have appeared. The devices form the basis for control system of an industrial accelerator [1] and they will replace parts of the racetrack microtron injector system [2,3]. Then the control system of the whole racetrack microtron will be based on these devices.

GENERAL ARCHITECTURE.

This family of smart devices is designed for the construction of distributed multilevel embedded applications according to the specification of "hard" real-time systems. Therefore devices should be very reliable, have the ability to use redundancy and, at the same time, allow the implementation of complex control algorithms including direct digital dynamic stabilising. To increase the versatility of the system components scalable hardware should be realised. The Open Systems concept requires that interconnection between components should be via a standard interface: the fieldbus. The devices cover the low and intermediate levels (nearest to the control equipment) of multilayer control systems. But, it is possible to connect a low-level system directly to supervisor and operator levels based on standard software and hardware via the standard interfaces supported by our devices. In other words, in standard, well known architectural solutions, our devices in many cases, but not in all, could replace the intermediate and low levels traditionally formed with the help of different universal expensive crate-based systems (CAMAC, VME, VXI) and industrial PLCs and ensure a cost-effective technical solution. Features of the described devices should reinforce this quite questionable statement.

In figure 1 examples of possible architectural solutions and applications of smart family devices are shown. One of the basic principles taken into account during the design is that: *intellectual resources responsible for the handling of the control task in real-time should be spread as evenly as possible and feedback loops should be closed locally and digitally, where possible.*

Interfaces.

Interconnection of different parts should be as standard as possible. At the same time supervisory functions from the top level of the control system should be fully supported to ensure access to the lowest level of control algorithms. Therefore the following interfaces have been chosen for interconnection: MIL-STD-1553B, RS-435 and RS-232C.

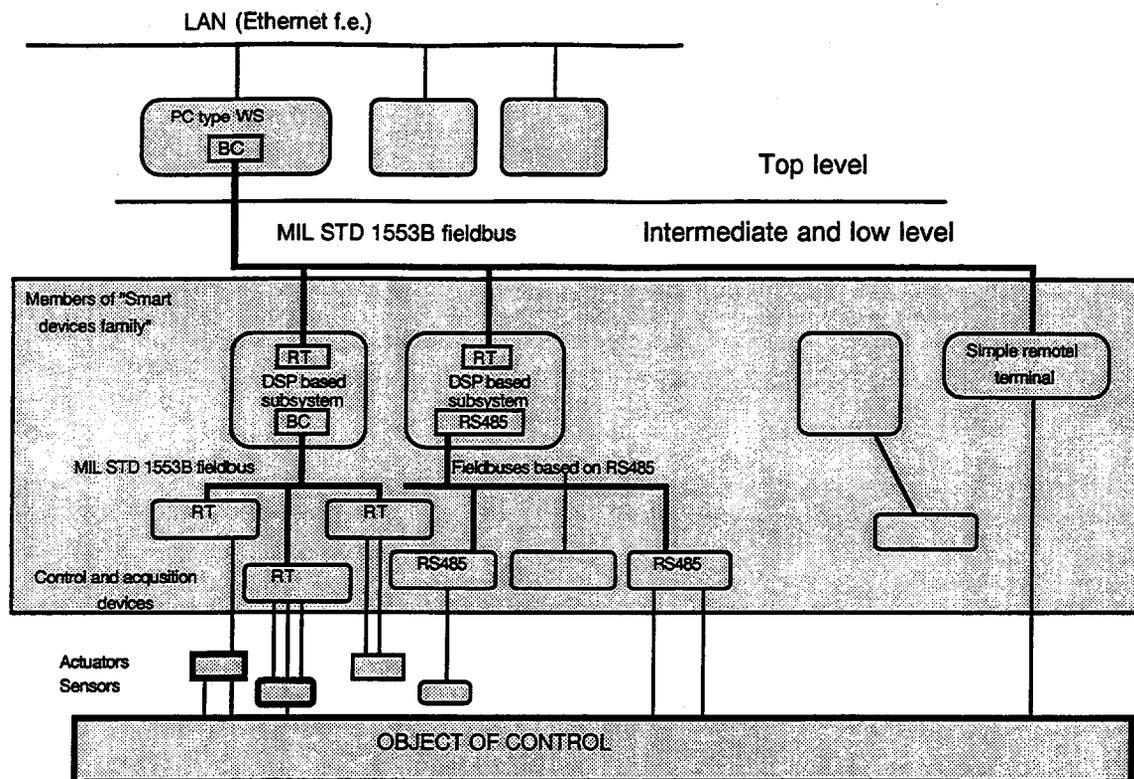


Figure 1. Location of the smart devices family in the standard control architecture.
MIL STD 1553B bus elements: RT- remote terminal, BC - bus controller.

The MIL-STD-1553B fieldbus interface is very suitable for "hard" real-time applications - aerospace onboard control, alarm interlocking, etc. Moreover it was designed for such applications and there is nothing comparable which is currently standardised. Many of the functions supporting high reliability are supported at the hardware level. This interface is very widely used in different accelerator control systems [4-7]. Another reason for our choice is that the interface is not message-passing oriented but oriented to blocks of subsystem memory exchange. There are some disadvantages to the interface - relatively high cost, low data transmission rates, limited address field in the standard message and the small number of remote terminals connected to one branch.

The RS-485 physical interface has been chosen as an additional one to ensure compatibility with a great number of industrial PLCs available on the market. Features of our hardware architecture ensure the support of different logical levels of fieldbus such as BITBUS, PROFIBUS, S-BUS and others [9]. Different "smart sensors and actuators" physically placed near the control object could be connected via this interface. The appearance of very cheap RS-485 transceivers with galvanic isolation lends support to the validity of our choice. The basic principle that *data should be processed as near to the place of generation as possible and the length of any wires with analog signals should be as short as possible* is implemented in this way.

The very well-known RS-232C interface could be used for connections with standard instrumentation equipment and for local testing and reconfiguration of "smart devices". In addition, an implementation of RS-232C allows the use of the device for small experiment automation, increasing the universal character of device application.

Planned for the design are special adapters between two different standard interfaces - MIL-STD-1553B and IEEE-488, for example, to allow the connection of standard powerful test equipment

SMART DEVICE HARDWARE.

It is a tragedy to any active working designer in the field of microelectronics equipment that every day fresh chips with higher and higher performance and cheaper and cheaper cost appear ...

We had to choose at least three base classes of components together with appropriate instrumentation and CAD systems in order to design this family of controlling devices - base microprocessor, base microcontroller and base type of programmable logic devices. The right choice should provide a fast and effective design process.

As a base processor for general and special applications in "smart devices" the TMS320C5x generation of fixed point DSPs of Texas Instruments have been chosen. These DSPs are fabricated in accordance with static CMOS technology. The combination of an advanced Harvard architecture (separate buses for program memory and data memory), additional on-chip peripherals, more on-chip memory and a highly specialised instruction set is the basis of their operational flexibility and

speed. The third generation of fixed-point 16-bit processors has a very good performance and characteristics: 35-50 ns single-cycle instruction execution time and can execute more than 28 MIPS. The C50x DSP processor can have up to 9K words of on-chip data and program RAM, up to 16K words boot ROM, JTAG scan path, one or two serial ports and a maximum address space of 64K words for data, 64K words for program and 64K words for I/O ports [10].

As a base microcontroller the PIC 16/17 product line from Microchip has been chosen. PIC 16/17 microcontrollers are RISC based on Harvard architecture.

As a base family of programmable logic devices, EPLDs and FPGAs (Electrically Programmable Logic Devices and Field Programmable Gate Arrays) from Xilinx has been chosen. Cheap EPLDs or FPGAs with small numbers of gates are very suitable to replace conventional logic chips used as glue logic in complex microprocessor devices. Quite expensive FPGAs with a large number of gates up to 20000 could be used for the silicon implementation of complex control algorithms or for interface functions such as MIL-STD-1553B interface including codes, message processing and answering blocks.

Automatic adjustment of the parameters for analog channels, such as zero shift, can be used with a special scheme of analog to digital and digital to analog channels.

The internal structure of the base processor control device is shown in figure 2. It consists of the processor core, input/output system and external interfaces.

The microprocessor core.

The microprocessor core consists of fixed-point, 57MHz TMS320C51 processors, an external boot ROM, optional fast static RAM separate for data and program, a control logic block and a crystal oscillator.

The low level software is activated after power-up or system reset and supports stand-alone operation and/or loading of the control program from ROM, a with maximum size of 8K words. Additional software can be loaded via an external interface (MIL-STD-1553B at present). Optional RAM for data and/or program can be mounted if the control algorithms require it. The maximum size of external RAM can be 32K words for data and 32K words for program.

Input/output system.

Information from the external world is received in analog and discrete digital form. Control signals can be generated in the same way. Sixteen differential or 32 non-differential analogue signals passed through input passive filters feed a multiplexer. The signal after the multiplexer is amplified and goes to the input of a 14-bit fast ADC. The instrumental amplifier on the multiplexer output has programmable 4-step gains. Up to four DACs with settling times of 1.5 μ s are used for analog control.

Direct digital control algorithms could be easily implemented by digital closing of the feedback loop. The number of effective bits in all operational scales is not less than 12, which ensures an accuracy of measurement good enough for most of the analogue parameters in control systems.

The input/output subsystem is situated in the I/O region of the processor memory and is presented as register-oriented devices.

External interface subsystem.

The MIL-STD-1553-B interface is based on a special hardware scheme. It consists of parallel to serial and serial to parallel codes generating serial data (Manchester II), additional control logic and a TTL to bipolar converter hybrid scheme for feeding a special pulse transformer, connected to twisted shielded pair.

The MIL-STD-1553B interface is situated in the I/O region of processor memory and presented as a register-oriented device. A complete set of Remote Terminal functions is built into the control software.

A full duplex on-chip serial port on the DSP provides direct communication with serial devices. The serial port is fully static and thus will function at arbitrarily low clocking frequencies. In order to realise a complete RS-232C interface, simplify hardware and decrease processor loading, a PIC16C54 microcontroller is used. It is used for data buffering and transmitting and receiving additional control interface signals. An additional system function implemented by the microcontroller is a system watchdog.

A second on-chip serial port is a TDM (time-division-multiplexed) one. It allows the device to communicate serially with up to seven other TMS320C5x devices. The port can be configured in either multiprocessor or stand-alone mode. In our case the TDM serial port operates stand-alone. The RS-485 transceivers are directly connected to serial port lines. Different configurations of external connection (one, two or three line) can be used, depending on synchronisation mode and the demands of the logical part of the interface.

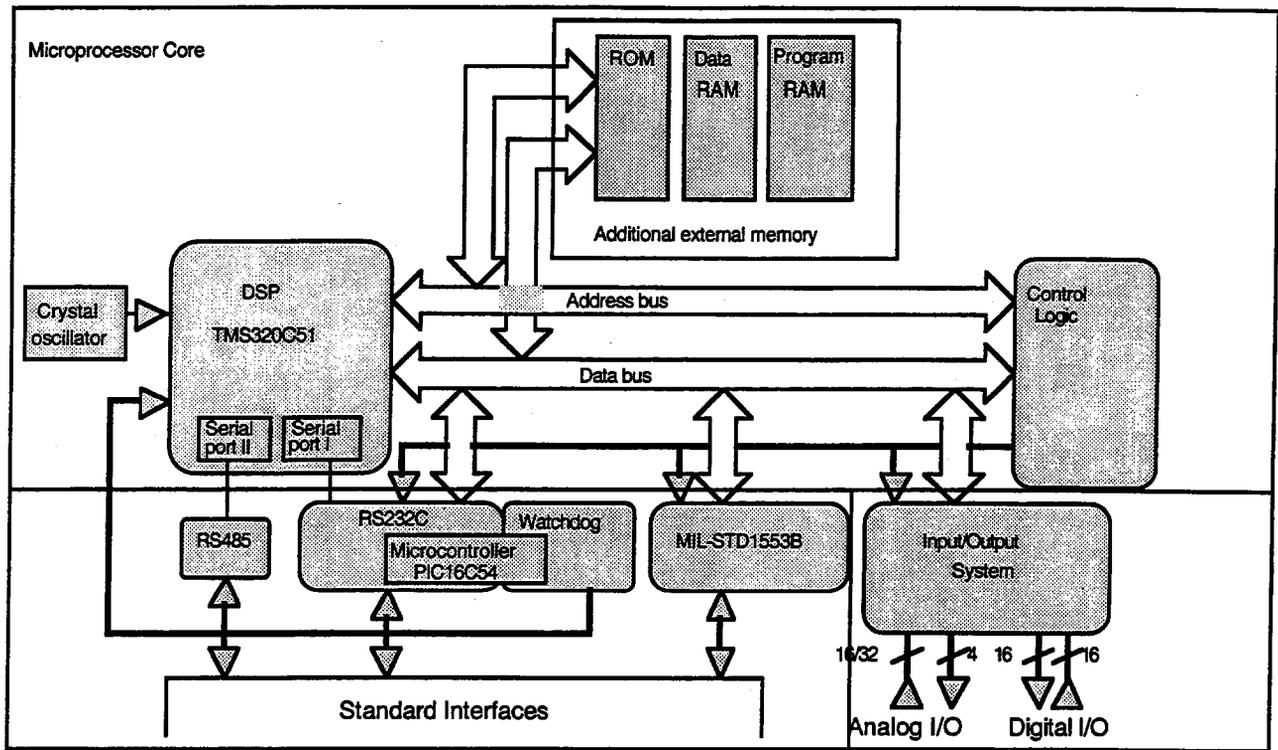


Figure 2. Internal structure of base processor controlling device.

An example of simple control devices.

An example of one possible simple control device is described below. This device is used for the simultaneous control by four stepping motors for positioning an experiment's target in space [11]. The main and practically only element of the device is the microcontroller PIC16C57 [12]. The PIC16C57 single-chip microcontroller is a low power, high speed, full static CMOS device containing ROM, RAM, I/O and a central processor unit on a single chip. It has a high-performance RISC-like 8-bit CPU with a 200 ns instruction cycle. The architecture is based on a register file concept with separate buses and memories for data and instructions (Harvard architecture). The data bus and memory are 8-bits wide while the program bus and program memory have a width of 12-bits. Peripheral features consist of 20 I/O pins with individual direction control, watchdog timer, power saving mode etc. The performance and peripheral features of the microcontroller are enough to generate correct phase signals for the stepping motors and to receive and transmit data via the RS-232C interface. The structure of the device is shown in figure 3.

SOFTWARE SUPPORT.

Software development for a control system is one of the most difficult and expensive parts. It is a peculiar feature of our times that the speed of hardware design is significantly raised by the use of powerful CAD tools. Yet software design, especially for embedded and real time applications, is still a very hard and slow process.

This smart device family has been designed to realise our intent to implement the use of the shared memory idea for control applications. We expect this approach to seriously simplify application object-specific software. The hardware will be more complex, especially for a communication system supported by a shared memory mechanism [13].

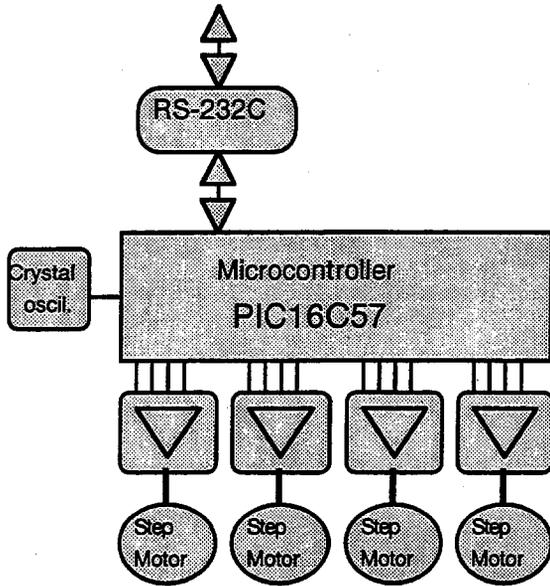


Figure 3. Structure of simple device for step motor control.

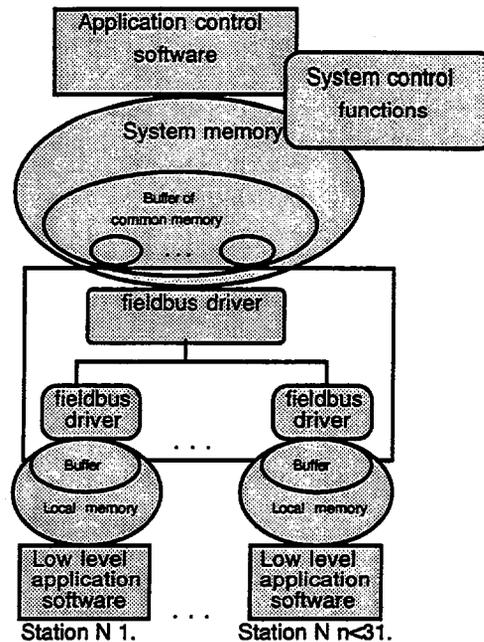


Figure 4. Structure of software system organisation with sharing memory approach.

According to this approach, high level application software knows nothing about the structure of the low level hardware. The message-passing method of data exchange between subsystems and processes is avoided. Application software is operating on a high level and directly accesses data and processes reflected in the memory of high level system. Moreover, application software doesn't check data validity; it presumes that data always correspond to reality. These data are the result of low level software operations on the powerful low level DSP-based devices and transmitted by the specially organised fieldbus, which ensures the validity of the data in the memory of the high level system. Due to the special architectural features the following idea is realised - *all the data reflecting the state of a control object is processed and transmitted for storing and supervising at the rate of its physical generation*. And vice versa, if the control system were to control the object, it could generate a reply, write it to memory and forget it, because it could be sure that the signal would influence the right object at the right time. It is necessary to know only the real physical nature of the received and transmitted (controlling) signals.

The general structure of the software is shown in the figure 4.

High level software support.

An IBM PC compatible computer has been used for studying and testing high-level software, supporting the shared memory approach. An ISA-bus-based MIL-STD-1553B interface card has been used.

Application software consists of different control, supervisory and data base algorithms and depends on the controlled object and control algorithms. The application software is independent of the structure of the controlling hardware, the type and number of actuators and the data acquisition channels. It operates only with the reflection of the external world in correct blocks of shared memory. This approach continues the tradition of the software that has been designed and is operating today in the control system of the racetrack microtron injector [2]. Man-machine interfaces and data base functions can be implemented in this part of software.

High level software can be written without special knowledge of real-time programming technology and the hardware structure of the control system. Programmers should know the physical properties of the controlled objects and algorithms and the rules of correct object operation. The one thing that should be done is the correct initialization of the driver supporting the memory reflecting mechanism.

The bus controller functions of the MIL-STD-1553B standard are supported in hardware. In order to implement the memory block reflection, a special card-oriented driver has been written. It ensures software support of the shared memory approach and compensates for the disadvantages of using a new interface card. It is necessary to point out that MIL-STD-1553B is very suitable for the implementation of the memory-shared mode of operation. Data block exchange instead of message-passing is one of the most important features of the interface.

A special interface card that supports in silicon many of the interface and driver functions is under design now based on FPGA technology.

System control and configuration is the most difficult part of the high level software. It can be written only by a person knowing the controlled object as well as the system structure. This software handles alarm interrupts and critical

loops. With the shared memory approach, it is possible not only to distribute data but to dynamically configure low level software, to control the real-time processing of the low level systems.

Low level software.

The structure of the low level software is shown in figure 5.

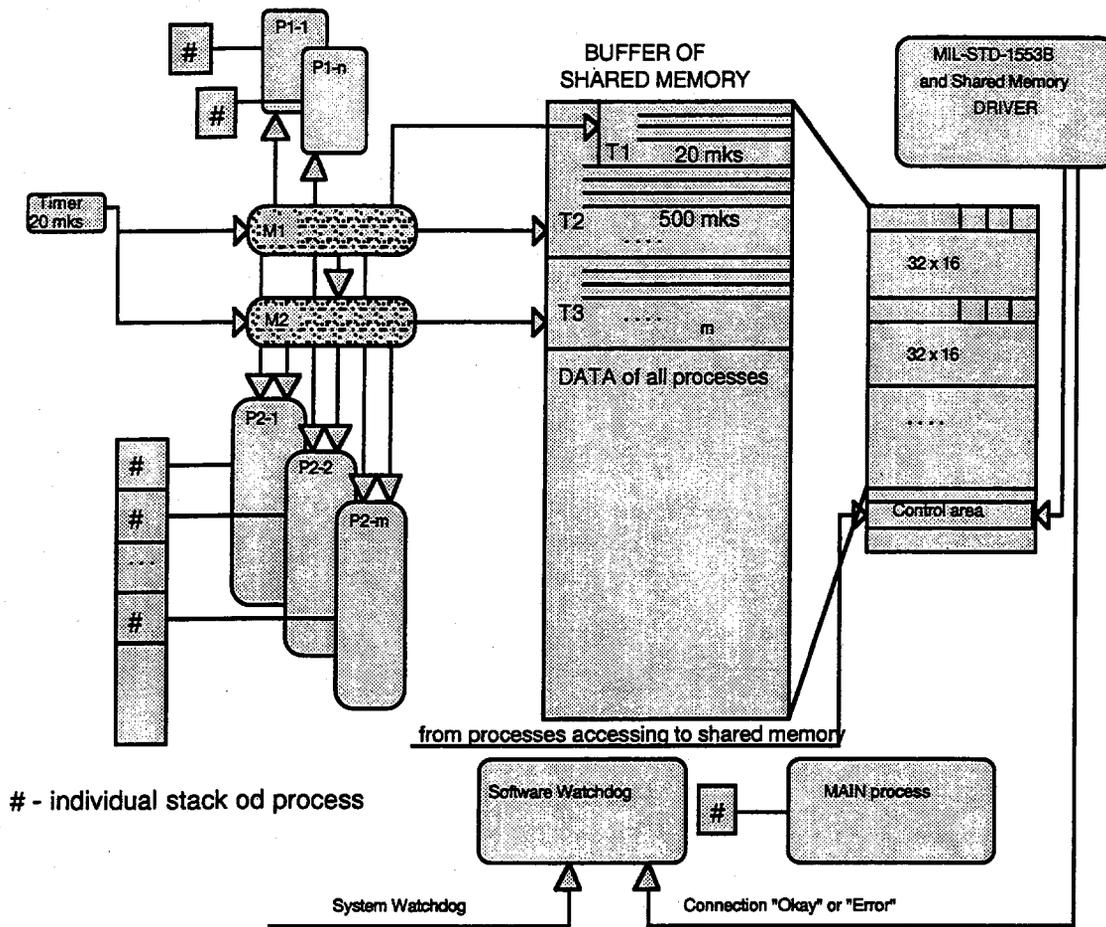


Figure 5. Structure of low level software.

The low level software system is based on monitors and processes. Memory is divided into a few blocks:

- shared memory, accessible from the top level;
- system data memory, not accessible from the top level;
- program individual process stack, accessible only for the process which is the owner of the stack.

According to the MIL-STD-1553B interface specification, 32-word length blocks could be exchanged during one cycle of network operation. Every 32-word memory block has an additional first word reserved for establishing the correct access to shared memory and is processed by the network driver.

Five different process types are used. The following three types of process exist in one copy:

- a network and shared memory support driver;
- a "software watchdog" to ensure overall control and check time synchronisation and process validity;
- an asynchronous background process for system control

They should be configured during program compilation and couldn't be dynamically changed during the system operation.

The network and shared memory support driver is the most critical part of this software because it must ensure the correct fieldbus time cycle operation during message exchange and ensure the correctness of data reflection in the common shared memory. Maximum loading of the processor by the fieldbus driver is less than 25% when the fieldbus loading is at a maximum. It should be pointed out that the remote terminal functions of the interface are implemented in software. Actual loading during normal operation is less than 15%.

Two types of synchronous processes exist. The first type (P1) is a single cycle "fast" process. The second type (P2) is a multi-cycle "slow" process. This process operates under the control of two monitors. The monitors are synchronised by a 20 μ s cycle timer. A rate monotone scheduling algorithm is used [14]. The first monitor (M1) only initiates processes. The second monitor (M2) switches processes and stops them.

There are three description tables supporting monitors, operation and the scheduling algorithms. The first table (T1) is a description table for the first type of process. The "time step" of table scanning by the M1 monitor is 20 μ s for P1 processes. The length of the T1 table is limited in today's realisation to 500 μ s. The total sum of P1 type process times should be less than 500 μ s. The number of processes is currently 25. T1 is a first "special" element of the description table T2. It is scanned by the M1 monitor element-by-element with a time step 500 μ s. According to this table P2 processes are initiated. The length of the table is limited by the maximum number of implemented processes. This table can be dynamically changed. The T1 table can only be changed after stopping all P1 type processes. Due to this mechanism, P2 type processes could be loaded and initiated via the fieldbus and stopped and unloaded without stopping system operation. Access to this table via shared memory ensures dynamic control of the initiated process.

Information about all initiated P2 type processes is stored in the third description table - T3. This table is scanned by the M2 monitor with a 500 μ s step and stops a process if its time is over.

According to a special time cycle (currently 1s), a "software watchdog" checks the state of the software system and the number and state of all initiated processes. If there are uncontrolled or excessively time-consuming operating processes, they are destroyed. The "software watchdog" handles interrupts from a hardware watchdog and the network interface (for connection failure) for correct process ending and switching off of the system.

Examples of P1 type processes are: reading data from one of the input ADC channels, internal clock, one step of a dynamic stabilizing algorithm and processing of digital inputs. Examples of P2 type processes are the control of motor movement and current scanning in a steering element.

All low level software is written in C50 assembler.

IMPLEMENTATION EXAMPLE.

The systems described above can be used in the following applications:

- distributed control systems for complex experimental installations;
- distributed data acquisition and processing systems with low and middle data rates;
- scientific experiments automation;
- industrial automation;
- stand-alone industrial controllers.

On the basis of the described hardware and software approaches the control system for the industrial accelerator CWELL 0.6/6.0 have been designed and implemented [1]. This accelerator is used for cacao powder processing with an electron beam. The output energy of the continuous electron beam is 0.6 MeV and the output beam power is 6.0 kW. The structure of control system is shown in figure 6.

One "smart DSP-based" device enough for the implementation of all the "hard" real-time algorithms. Moreover, direct digital feedback loops for high voltage and RF power stabilization are implemented. Based on cheap and well known hardware, the PC-type computer is used for man-machine interface support and storing of operational data.

This technology will be used soon for upgrading the control system for the Moscow racetrack microtron [14].

CONCLUSIONS.

The described "family of smart devices" is one more set of tools for the full scale automation of different types of objects. All the features and advantages of the set become evident during hard real-time system implementation. The shared memory approach simplifies high-level application software whereas complexity of the interconnection interface is increasing.

ACKNOWLEDGEMENTS.

The authors wish to thank ICALEPCS'95 Chairman Dr. Peter Lucas and the Organising Committee of the conference for the financial support for conference participation which initiated the appearance of this article.

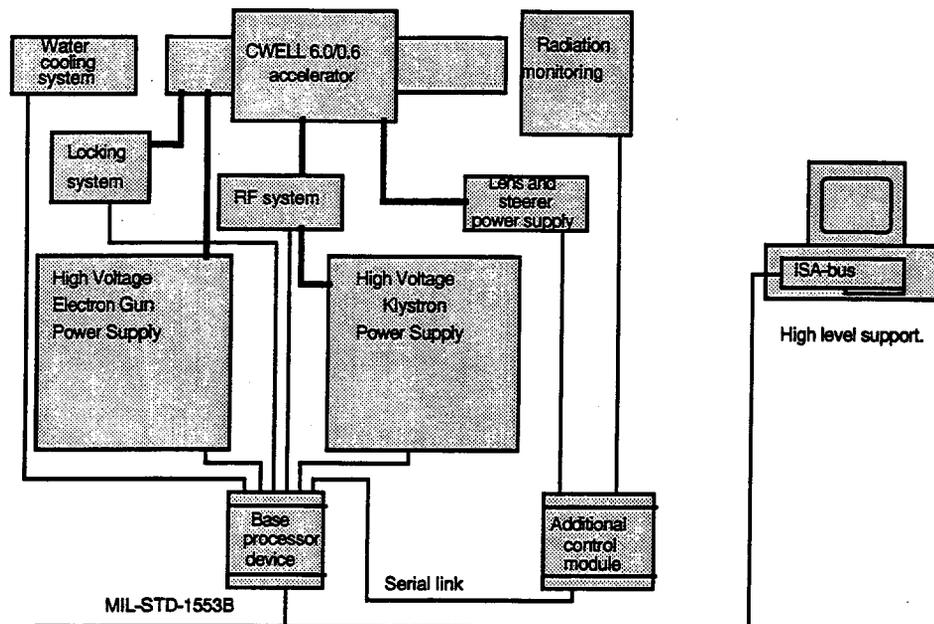


Fig.6. Structure of industrial accelerator control system.

REFERENCES.

- [1] A.S. Alimov, K.A. Gudkov et al. Experimental Study of a Prototype High-Current CW Linear Electron Accelerator // Instruments and Experimental Techniques, Vol.37, No.5, Part 1, 1994
- [2] A.S. Chepurnov, I.V. Gribov, et al., Moscow University Race-Track Microtron Control System: Ideas and Developments //Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Tsukuba, Japan, KEK, 11-15 Nov.1991) Tsukuba, KEK,1993 pp.140-142.
- [3] A.S. Chepurnov, I.V. Gribov, S.Yu. Morozov, et al. Systems for Local Control of Race Track Microtron Accelerating Section. //Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Tsukuba, Japan, KEK, 11-15 Nov.1991.) Tsukuba, KEK,1993 pp.424-426.
- [4] P.G. Innocenti. The LEP Control System: Architecture, Features and Performance., Particle Accelerators, 1990, Gordon & Breach, Science Publ. Vol.2,pp.183-190
- [5] M.V. Tyrrell. The LEP Alarm System. //Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Tsukuba, Japan, KEK, 11-15 Nov.1991.) Tsukuba, KEK, 1993 pp.254-260
- [6] M.R. Shea, R.W. Goodwin, M.J. Kucera and S. Stirbu. ARCNET as a Field Bus in the Fermilab Linac Control System./Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Tsukuba, Japan, KEK, 11-15 Nov.1991.) Tsukuba,KEK,1993, pp.291-294.
- [7] D. Bulfone, P. Michelini et al. The ELETTRA Field Highway System./Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Tsukuba, Japan,KEK,11-15 Nov.1991.) Tsukuba,KEK,1993 pp.313-317
- [8] R. Rausch Tutorial on Standart Fieldbuses Application in Physics Laboratories, Invited Tutorial Paper ICALEPCS'93, Berlin,1993.
- [9] TMS320C50 User's Guide //2547301-9721 revision D, January 1993, Texas Instruments Incorporated.
- [10] A.S. Alimov, V.K. Grishin, et. al. Studying of coherent mechanisms of X-ray generation on the electron accelerator NPI MSU. Preprint No 95 - 23/387. Moscow 1995. - 24 p.
- [11] Microchip Data Book 1994 Edition// April 1994, Microchip Technology DS00018G.
- [12] V.I. Vinogradov., A.S. Chepurnov, K.A. Gudkov, A.V. Shumakov. Modern Architectural Solutions on Automation and Computer Control Systems for Accelerator and Other Objects (These Proceedings).
- [13] A.D. Ferreri. Real-Time Scheduling Algorithms.// Dr.Dobb's Journal, Vol.19, Iss.15, December 1994, Miller Freeman Inc.
- [14] A.S. Chepurnov, K.A. Gudkov, A.V. Shumakov. Digital control of RTM linac RF system with DSP.//Proc. of the IV European Particle Accelerator Conference, London, 1994. vol.3, pp. 1845-1847.

Proton Storage Ring Control System Upgrade -- Interim Status†

Michael A. Oothoudt, Eric A. Bjorklund, Stanley K. Brown, Norman T. Callaway, Gary P. Carr, L. Robert Dalesio, Dolores B. Duran, John A. Faucett, David P. Gurd, Matthew W. Hardy, Margye P. Harrington, Jeffrey O. Hill, Debora M. Kerstiens, Andrew J. Kozubal, Franklin J. Naivar, Matthew C. Needes, David J. Ostrem, Jerry M. Potter, Bobby A. Quintana, Stuart C. Schaller, Fred E. Shelley Jr., Matthew W. Stettler, Michael E. Thuot, George D. Vaughn, David S. Warren, Robert E. Weiss, Rozelle M. Wright, Martha V. Zumbro

Los Alamos Neutron Science Center, Los Alamos National Laboratory, Los Alamos, New Mexico 87545

The Los Alamos Neutron Science Center (LANSCE) is in the process of a significant upgrade to its facilities. As part of this upgrade, the control system for the Proton Storage Ring (PSR) is being upgraded to an EPICS-based[1] system following the standard control system model. Goals of the overall upgrade required implementation of the new system with special attention to maintaining the functionality and high availability of the old system. The first stage of the control system upgrade has replaced about 75% of the old system and has been in high availability use for over two months. The old and new systems are briefly described and the strategies used to meet the upgrade goals are discussed.

1. INTRODUCTION

The LANSCE facility (formerly known as LAMPF, the Los Alamos Meson Physics Facility) provides 800 MeV proton beams for several experimental areas. A principal facility is the Manuel Lujan Neutron Scattering Center (MLNSC), where a high peak current proton beam is scattered off a tungsten target to produce intense neutron beams for materials science studies. The high peak-current proton beam is produced by accumulating beam from the LANSCE linac in the Proton Storage Ring (PSR) and extracting the stored beam in a single 270 ns turn.

In 1994 an upgrade of the PSR was begun to improve beam delivery for MLNSC. The goals of the upgrade relevant to the control system are

- routine, sustained 8 months/year operation of LANSCE,
- beam availability >80% by 1996; >85% by 1998,
- less than 5% downtime from intervals >8 hours,
- reduced beam delivery operating costs, and
- 100 μ A routine operating current at 20 Hz by 1998

An upgrade of the PSR control system was approved as part of the overall upgrade and work started in mid 1994. Budget, available manpower and the LANSCE operating schedule precluded performing the entire control system upgrade before beam tuneup began for the 1995 production period. An interim architecture was installed that simultaneously supported all old and new software and hardware in the control system. The hybrid system was successfully fielded in June 1995 for testing, equipment checkout and beam tuning. The new system has been in highly reliable beam production for over two months.

2. THE ORIGINAL PROTON STORAGE RING CONTROL SYSTEM

The architecture of the original PSR control system[2] is shown in Figure 1. The system used a centralized database on a VAX computer running VMS. There were approximately 3500 hardware channels and 400 software channels in the system when the upgrade began. Approximately once every 2/3 of a second the VAX wrote all command channels to the remote computers and read all output data from the remote computers over a CAMAC serial highway. The PDP-11 remote computers commanded and read CAMAC modules in CAMAC crates on their local serial CAMAC highways. In addition the VAX directly addressed CAMAC crates on the VAX serial highway; these crates interfaced to knobs, operator displays, the ring buncher RF wave-form generator and the ring Beam Position Monitor (BPM) system. Other data were acquired and commanded over Ethernet from the Linac Control System (LCS). The LCS is an independent system using a demand-based architecture[3].

† Work supported by the U.S. Department of Energy.

The operator interface to the control system was through dumb terminals, Elographic touch panels on Lexidata display screens and knob panels and these devices resided on the VAX Unibus or the VAX CAMAC serial highway. In addition, operators had access to LCS facilities through peripherals from that system. Over the lifetime of the system the central computer was upgraded from a VAX-11/750 to a VAX 4000/300 and the remote computers were upgraded from LSI-11/2 in-crate computers to standalone PDP-11/73s.

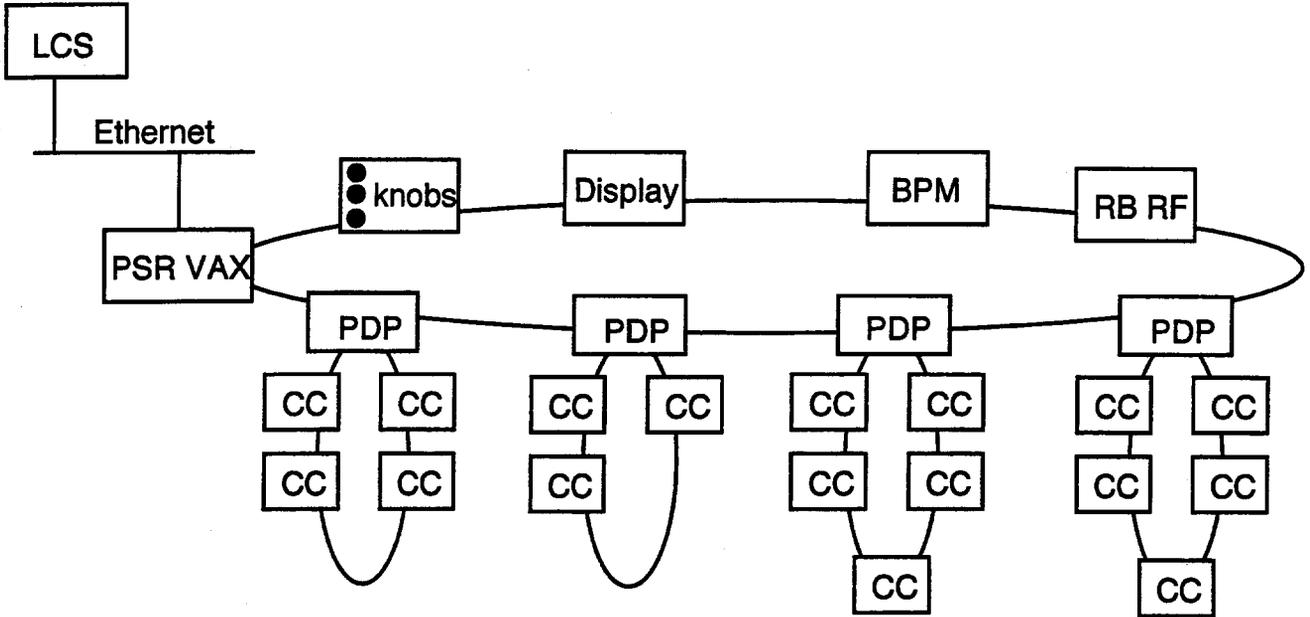


Figure 1 - Architecture of the PSR control system before upgrade. "LCS" is the Linac Control System. The ellipses are CAMAC serial highways. "CC" is a CAMAC crate. "BPM" is the interface to the Ring Beam Position Monitor System. "RB RF" is the ring buncher RF waveform generator.

An upgrade of the control system was desired for several reasons. Applications were, in general, developed in Fortran by the programming staff and even they required long, expensive periods of time to field new applications. The high cost and long turn-around time of new applications were considered major flaws of the system. Furthermore, while the original control system had good availability (99.7%) in the recent past, signs of end-of-life failures were beginning to appear, including multi-day down-times that fortunately occurred when beam was not scheduled for delivery. Because much of the hardware was 10 or more years old, manufacturer support was limited and in some cases was no longer available. Maintenance costs were rising because of lack of manufacturer support and dwindling local expertise. Failures of the long CAMAC serial highway were difficult to troubleshoot. Lexidata displays showed intermittent failures that were never adequately explained. Hardware features (such as low maximum program memory in the PDP-11s) limited ability to expand the system. Finally, the periodic migration of data limited the responsiveness of the system; e.g. the minimum possible response time for knobbing a channel was 1.33 seconds.

3. THE NEW PSR CONTROL SYSTEM

The new system is based on EPICS[1] software and a distributed architecture. Figure 2 shows the architecture of the new control system as implemented for 1995.

3.1 Hardware

For budgetary reasons the individual serial CAMAC loops and crates have been retained. However, it was possible to reconfigure some of the serial loops to reduce their lengths and to remove crates of harp electronics no longer in use; this has reduced the incidence of serial highway problems and made trouble shooting problems easier. Several unreliable stepper motor modules were also replaced with Allen Bradley (AB) DACs; this change is still under discussion, since some system experts believe the improved device control does not adequately compensate for running magnets to zero current when power to the DACs is lost.

Remote computers have been replaced with Motorola 68040 monoboard I/O Control (IOC) computers in VME crates. The IOCs run local databases that read and control the modules on their local CAMAC serial highways and Allen Bradley crates. Most data in this system is polled by the IOCs at 1 to 5 Hz; this rate may be increased to improve responsiveness of the system when the system is complete enough for optimization. Programs on other IOCs or Operator Interface (OPI) computers obtain data through EPICS Channel Access by posting interest in particular data fields; data is shipped by an IOC to interested programs over Ethernet when changes occur outside predefined deadbands. Building the IOC databases to provide the functionality of the old control system was the major part of the upgrade effort.

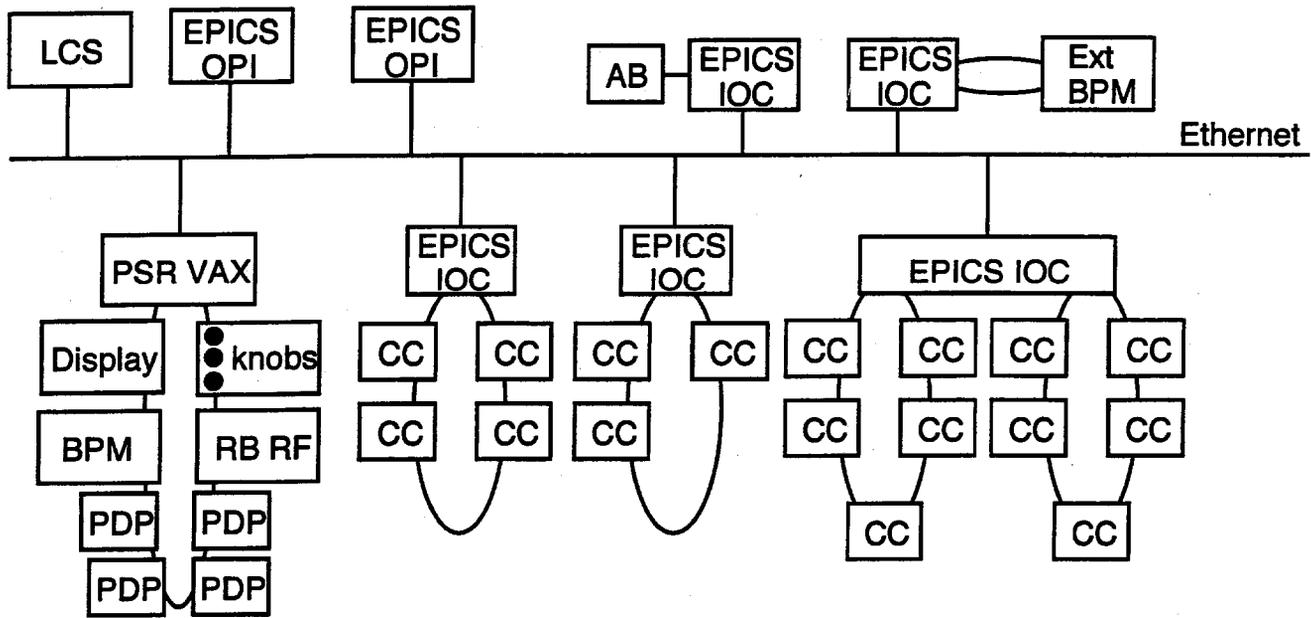


Figure 2 - Architecture of new PSR control system used in 1995. "LCS" is the Linac Control System. The ellipses are CAMAC serial highways. "CC" is a CAMAC crate. "RB RF" is the ring buncher RF waveform generator. "AB" is an Allen Bradley electronics crate. "BPM" is the interface to the Ring Beam Position Monitor System. "Ext BPM" is the Extraction Line BPM system. PDP-1s are present for the fallback strategy.

3.2 Software

The Operator Interface (OPI) is provided by Sun Sparc 5 workstations communicating with the IOCs over Ethernet. Two OPIs are provided in the control room PSR console with two 20 inch monitors and a mouse on each. Screens to replace old PSR applications were mostly developed with the EPICS edd screen editor and displayed with the dm display manager. Four screens requiring data and file manipulation not provided by edd/dm were built with the tcl/tk package[4]. All but five of the old PSR graphical applications were converted for 1995 operations.

Three new applications were installed as part of the upgrade plan: a display of the state of the Radiation Security System logic, a display of beam spill monitors in the PSR and a display of extraction line BPMs. Design and implementation of the first two new displays took four man-weeks (compared to an estimate of 52 man-weeks in the old PSR system.)

3.3 Interchanging Data Between Systems

Many of the OPI screens require data from both the LCS and the old PSR systems. Gateway software[5] provides EPICS Channel Access to this data from the OPIs. Because of implementation details, the gateways provide fresh data at a 0.1-1 Hz update rate rather than on-change. This update rate has been singled out by users of the system as being too slow.

Several applications in the old PSR system were not converted to EPICS for 1995 production due to budgetary and manpower constraints. These applications and many LCS utilities require access to data that is now in the EPICS IOCs. Data migrator software[5] was created to periodically transfer a fixed list of data channels between the EPICS IOC databases and the PSR/LCS databases. A side effect of implementing the migrators was that implementing knobs in EPICS became unnecessary for 1995; the existing LCS knob system could control EPICS

devices through the migrator path. Furthermore, the problem of time correlating data stored to disk by EPICS and LCS was avoided, since all data could be recorded by LCS. The added complexity of the migrator software did increase the trouble shooting problems. However, deferring these two jobs until future years helped to meet the delivery schedule for 1995.

4. PRESERVING HIGH AVAILABILITY WHILE UPGRADING A PRODUCTION SYSTEM

A principle goal of the overall PSR Upgrade was to improve availability of the entire PSR beam delivery system. The PSR control system ran at an availability of 99.7% during the most recent previous beam production period. The upgraded control system was expected to run with at least this availability in 1995, while maintaining all functionality of the old system. Higher availability in future years was also expected. These challenging goals for a new, hybrid system were attacked by a series of efforts:

- **Options Review:** As part of the planning process in early 1994, a thorough review[6] was made of previous proposals for PSR control system upgrades and new possibilities were identified. A total of seven options were identified, including "do nothing." Most of the options were expected to become as obsolete as the current system within a few years. The EPICS tool kit was selected as the basis on which to build the upgrade.
- **External Review:** After an initial plan for the upgrade was constructed, an external review was held. Control system experts from CEBAF, DESY and SSCL reviewed the proposed plan and could "identify no showstoppers," but made several suggestions that were folded into the planning.
- **Fallback:** The requirement for high availability of the new system left little time for debugging the system once beam delivery began. It was conceivable that running with beam could show flaws, in spite of testing without beam, that could prevent the initial tuning of PSR. Therefore a "fallback" strategy was developed such that the control system could revert to the old PSR control system if necessary. As shown in Figure 2, PDP-11s and old display systems were maintained in the system for 1995 operation. Hardware for the new IOCs was installed such that a single set of cables (the local CAMAC serial highway) could be manually swapped between an IOC and PDP-11. The migrators insured that the PSR database contained a copy of all current setpoints. The fallback system was tested before beam delivery started and was demonstrated to resurrect the old control system in less than 20 minutes. While the fallback system did meet the requirement for assured operation, control system developers felt it incurred extra development costs and limited enhancements to the system.
- **Collaboration:** The budget for training the developers on EPICS was limited to 6 one-hour classes. To provide further training, a collaboration was formed with local EPICS experts to do the upgrade. Approximately half of the manpower used to construct the system came from the EPICS experts. Highly successful teams of PSR and EPICS experts were assigned to work on projects for the upgrade. To aid this collaboration, the PSR developers moved into the same office building as EPICS experts. The proximity of the two sets of people significantly aided cross training and communication.
- **Screen Review:** The budget for training users of the new system was also minimal. Furthermore it was very important that the system initially fielded in June of 1995 should support all functions of the old system. To address these two issues, a series of screen reviews was held. Screen developers met with accelerator operators, technicians, engineers and physicists who would be using the screens. Instead of lengthy requirements and users manuals, prototypes (sometimes very crude mockups) of the screens were reviewed. These screen walk-throughs were exceptionally successful in catching misconceptions by control system developers and getting suggestions from customers. Furthermore the customers were trained on using the system during the reviews.
- **Customer Participation:** One of the accelerator operators was detailed to work with the control system developers for four months. He was assigned to build the screens for the major utility used to control the beam lines. His active and detailed participation during construction of the system made a major impact on the usability of the system.
- **Testing:** Much of the PSR hardware was not operated during the previous 18 months. Therefore an attempt was made by the control system developers to check every channel in the new system before beam delivery. With the help of operators, an attempt was made to operate every device in the system, including tests of interlocks. A channel list of the system was used as a check list. Problems with software, CAMAC hardware and devices were found. Unfortunately many devices in the systems were not ready for operation, so only a subset could be tested and corrected where necessary. Later the system owners performed another channel-by-channel checkout of the system which uncovered further problems.
- **Training:** Operators were trained on using the EPICS edd/dm tools for building screens. Special captive accounts were set up to allow using these tools without learning Unix. After beam production began, operators built a number

of very useful screens that were not in the upgrade plan. These screens made the new system significantly more useful and aided in getting reactions from customers.

- **Coordination:** Up to the point of channel-by-channel checkout, coordination meetings for the upgrade were held weekly. At this point, brief daily meetings were held to analyze problems uncovered. The problems and solutions found were published in publicly readable forms.
- **Standing Shifts:** During initial tuning of the PSR, a 25% contingency was built into the schedule for working on control system problems. Control system developers stood shifts in the Central Control Room 16 hours per day with a developer on-call for the remaining eight hours per day. This allowed rapid response when problems were identified and helped developers better to understand how the control system was being used. Customers were also given the feeling that the system had not been "thrown over the wall."
- **Observation:** After beam was in production, developers spent significant amounts of time in the control room observing the use of the new control system. Assistance in use of the system was provided and problems were more readily identified.
- **Polling for Input:** After the system had been in use for several weeks, reports of problems stopped coming in. However, there were rumors of difficulties. Regular meetings with each operating crew were scheduled to poll for any input on the system. This uncovered a significant number of "inconveniences" that were preventing efficient operation of PSR. These polling meetings will continue throughout the operation period.
- **CQI:** One significant class of problems uncovered by the polling meetings was magnet instability. A Continuous Quality Improvement team was formed with software, electronics and power supply members. This effort has uncovered problems with software calibration, CAMAC electronics, field wiring, grounding and power supply internal controls. Progress is being made on these problems, some of which date to design decisions made in the early 1980s.
- **Customer Critique:** A meeting was held with customers to critique the new system after it was in production use. The overall summary was that the customers "got what they needed." Two exemplary practices were identified:
 - ◊ Screen Reviews
 - ◊ Operators trained to create screens that are improving facility operation.

Areas needing improvement were also identified:

- ◊ Final control system checkout overlapped too much with system experts' checkout of equipment. Schedule future control system upgrade work to be completed before other people need to use it.
- ◊ Some screens did not get reviewed. Insure all future screens go through the review process. Review existing screens that were not reviewed.
- ◊ Parameter and calibration constants need better configuration control. Make it easy for non-experts to determine what constants are in use.
- ◊ Some CAMAC hardware has an unacceptably slow response time (e.g. some multichannel ADCs with 8 second cycle time). Upgrade this hardware.
- ◊ Only the operators were trained in screen building. Technicians, engineers and physicists also need to be trained.
- ◊ More training is needed on trouble shooting the new system.

Finally, a number of changes and enhancements for the new system were identified.

5. CONCLUSIONS

Highlights of the PSR control system upgrade to date include:

- After 56 days of production use, the new PSR control system has run with 99.946% availability, handsomely meeting the requirements of the upgrade. The majority of the control system downtime was due to CAMAC hardware problems.
- Customers said they "got what they needed" to do their work.
- Operators find the new OPI user interface preferable to the old touch panels.
- Operators are rapidly creating new screens to improve accelerator operation.

The old control system is approximately 75% replaced for 1995 operation. The cost of the work to date is approximately seven man-years plus \$88K for hardware and software purchases. Costs for other upgrade options were

estimated to be 30 to 70% higher for a similar amount of work. A rough cost breakdown shows 5% of the cost for planning and prototyping, 47% for IOC database work, 36% for non-database work and 12% for system commissioning.

Work has begun to complete the upgrade by April 1996 for the 1996 equipment checkout, beam tuning and production. In addition to converting the ring BPM and ring buncher systems to EPICS, much of the infrastructure in the old control system (e.g. diagnostic software) needs to be converted. Documentation, error logging, knobs and magnet control problems need to be addressed. Speed of the gateway software needs to be improved.

A future goal is to have a single control system for the entire LANSCE facility. Outline studies have begun to estimate the scope of the work necessary to convert the Linac Control System to EPICS.

ACKNOWLEDGMENTS

The authors wish to thank Dave Barker, Johnny Tang and Mary Wise (CEBAF), Ying Wu (Duke University) and Mark Rivers (University of Chicago) for providing baseline code and support for the CAMAC library, driver and device support. Further thanks go to Johannes van Zeijts (CEBAF) for providing a version of tcl/tk with EPICS channel access included.

REFERENCES

- [1] L. R. Dalesio, et al., Nucl. Instr. and Meth. A352 (1994) 179.
- [2] P. Clout, et al., Nucl. Instr. and Meth. A247 (1986) 116.
- [3] G. P. Carr, et al., *Proceedings of the 1987 Europhysics Conference on Control Systems for Experimental Physics*, CERN Yellow Report 90-08 (1990) 107.
- [4] J. K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, Reading, Mass., 1994.
- [5] S. C. Schaller and M. A. Oothoudt, these Proceedings.
- [6] M. A. Oothoudt, et al., *Evaluation of PSR Control System Upgrades*, AOT-6 Technical Report AOT-6-94-25, Los Alamos National Laboratory, 1994.

The Case for Commercial Software

R. T. Westervelt, R. B. Rothrock, P. N. Clout
Vista Control Systems Inc.

Abstract

This paper examines the current trend in the particle accelerator community to use commercial rather than in-house software to develop process control systems. We will analyze why this trend has developed and discuss specific applications suited to commercial, rather than internally developed, software. We will conclude with features to consider in choosing commercial software for control projects.

Introduction

As a vendor of software suitable for particle accelerator controls, we advocate the purchase of commercial, as opposed to in-house, software solutions. However, our staff and management are long-term members of the particle accelerator community; many of us have lived on both sides of the fence first as in-house developers at laboratories and in industry, and then as commercial developers for Vista Control Systems. In this paper we examine reasons why commercial software provides an increasingly attractive option to home-grown control software. We analyze the trend toward purchasing commercial software and we discuss reasons why commercial software is often chosen over software developed in-house. Along the way we address some of the issues to consider in choosing a commercial approach.

Current trends

The use of computer control systems for particle accelerators began during the very early stages of computer technology development. No software was available, so accelerator personnel developed everything from scratch internally, from operating systems to programming languages to networks to databases. This situation resulted in many internal developments and enabled remarkable progress in accelerator and experimental physics. Today many accelerator controls groups have worked hard to maintain this position at the cutting edge of software technology.

As the computer industry matured, however, many of the fundamental building blocks for control system development have become available commercially. For example operating systems and networking technology available commercially have almost completely replaced home-grown solutions and hardware in both scientific and industrial computer applications.

To examine the trend toward commercial software implementation in particle accelerators we have studied the number of references to commercial software made in proceedings of previous ICALEPCS conferences. We count only those references to commercial software when it played a crucial role in the accelerator control system; we count only once any paper reporting the use of several commercial software packages. The dramatic trend toward use of commercial software reported in the proceedings is shown in figure 1.

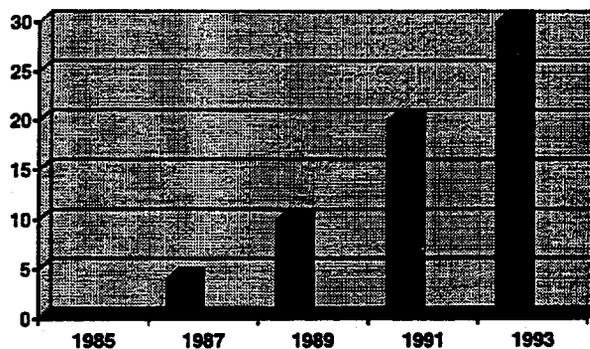


Figure 1 Commercial Software Usage Reported in ICALEPCS

Because ICALEPCS represents a worldwide community of control systems developers, the graph shows the broad base of this trend. The continued increase in references to commercial software over the past few conferences also shows that this trend has not yet reached its limit.

Software Requirement of Physics Control Systems

The topic of the requirements for software for physics control systems has been extensively discussed. One good summary is the panel discussion on software sharing from the last ICALEPCS (1). That panel discussion and the work being done on software sharing have identified the key application areas that are common across control systems and represent the areas that might benefit most from investment in commercial technology. The types of applications needed in the modern control systems and the characteristics that good components must have are listed below:

- User interface development
- Console manager
- On-line help
- Database access
- Real-time database management
- Sequence builder
- Simulation access
- Data logger
- Archiving system
- Alarm system
- Equipment access
- Startup/recovery mechanism
- System configuration tools

Some of these application areas are common to both industrial and scientific control systems. Industrial systems have the same general requirements as scientific ones for user interface development, help, real-time database management, data logging, and alarm management. This overlap in functional requirements contributes to the synergy between commercial applications and scientific control systems. This synergy in turn drives the trend for using commercial software in scientific control systems. Some other application areas provide common ground for commercial and industrial control systems, but the degree of overlap is not so great as in those applications mentioned above. Equipment access on an industrial level tends to be centered around access to various types of commercial PLCs with less use of faster and more diverse data interfaces, such as VXI, VME and CAMAC. On the scientific side this equipment mix tends to switch more toward high-speed interfaces, reflecting the need for gathering data on the time scales associated with acceleration cycles. Like scientific control systems, industrial ones need access to simulations, but in the industrial case the model codes tend to be built directly into the control system itself and are routinely executed as part of such a control system. For industry there is less human interaction with the models as part of the control process.

Why commercial software abounds

Economics: Economics plays perhaps the greatest role in the move toward commercial software in the accelerator and physics communities, where fundamental changes are taking place. In the United States, for example, the SSC has been canceled and funding is tight at other laboratories. Commercial software can reduce the total cost of the basic software by at least a factor of ten (2).

One of the arguments for in-house software development has been cost control. But the reality is that one underestimates the budgets for in-house projects, and inflates them with expanded requirements for commercial ones. We at Vista Control Systems have experienced first-hand the impact of misjudging in-house project duration and effort required: twice we chose to develop tools to re-engineer a core product, rather than purchasing commercial tools to facilitate the re-engineering process. We felt we could develop such tools for half the cost of a commercial product, but in the end our costs to develop them more than doubled because we underestimated the manpower required and because we expanded our requirements as we began the re-engineering project.

Personnel costs can be underestimated also. Consider the development of user interface software. Most modern graphical user interface systems such as Xwindows/Motif or Microsoft Windows contain enormous capabilities accessed through large and complex APIs. Teaching a staff to use all of the capabilities requires a large investment in time—time better spent developing control system requirements, managing the system and expanding applications. Because many commercial packages provide user interface management systems there is no need to invest time developing expertise that is difficult to retain after operations begin and that will someday be outmoded.

Don't think personnel costs will be eliminated, though, with commercial software. Because they are by necessity generic solutions for a variety of industrial and scientific controls, commercial systems do not often address the specific needs of the particle accelerator community or any particular facility. Staff must take advantage of the extendibility built into these products to tailor control system software.

Compatibility: Often, in the name of expedience, internal software packages fail to provide compatibility between software versions. However the pressures of the marketplace have made backward compatibility an essential component of a good software release for commercial vendors. At Vista we have more lines of code in the regression test suites for our database software than lines of code in the Vaccess database product. We also perform extensive beta testing—a luxury not often afforded in-house developers.

Support: Commercial software packages generally provide professional support, training and documentation. These elements enable a smooth, speedy implementation of control software, ensure controlled start-up costs and provide continuing support despite years of operator and staff turnover.

Low risk: In application areas common to industrial and scientific controls commercial software is a sound, tested alternative to that developed in-house.

Source code arrangements: Customers of commercial software are often concerned about access to the source code. The need to fix bugs quickly and to add new functionality is important to industrial and scientific customers. And all customers want to guard against the software vendor going out of business or abandoning the software product.

Standard software engineering and legal practices, though, allay fears concerning source code access. First, good software is extensible so that users can expand it to suit their needs. An example is the IDL data analysis package which interfaces easily to other programs. In the PC market most major software vendors provide scripting packages to extend functionality. A favorite example of this extensibility is the adding of a new device driver to an operating system. Access to the operating system source code may be useful, but if that system is well documented and adequate examples are provided, having the source code is not necessary.

As a vendor we do not recommend that customers modify source code due to the danger of introducing errors and making it difficult to provide support. Further, we provide legal and administrative mechanisms to address concerns about our company abandoning a software product. Source code escrow agreements provide customers with the source code in the event that the company does cancel maintenance of any product.

Conclusions

Our conclusion is that carefully chosen commercial products provide a sound economical alternative to in-house software. With a wide variety of flexible, well-supported, extensible commercial packages on the market, in-house development is now only one option, not a necessity. When beginning to look at the commercial options, we suggest considering a software package that provides

- much of the functionality you need in order to control your process effectively.
 - mechanisms for your in-house staff to build additional functionality into their software controls.
 - backward compatibility between software versions.
 - professional support, training and documentation.
 - a history of effective implementation in both scientific and industrial applications.
- a• n acceptable source-code escrow agreement.

If one can find these qualities in a commercial software package, the chances are that the package will greatly benefit his control systems project and allow his organization to devote greater effort and more resources to its prime tasks.

References

- (1) B. Kuiper et.al., Panel Session on Software Sharing, NIM, A 352 (1994), pp 501-515.
- (2) P. Clout, "Sharing Control System Software," *Proceeds of the 1993 Particle Accelerator Conference*, Vol. 3, 1993, pp 1801-1805.

Integrating the commercial software package LabVIEW with Fermilab's Accelerator Control NETWORK

W. BLOKLAND

Fermi National Accelerator Laboratory*

ABSTRACT

An overview is given of the interface developed between the in-house networking protocol, ACNET, and the commercially available software package for instrumentation control and analysis, LabVIEW. The interface supports data reading and writing access by consoles over ethernet and token ring networks. In addition, LabVIEW applications can read and write data of other nodes by communicating over ethernet with the TCPORT server on a VAX. Tools are provided to help the developer implement LabVIEW programs and make entries in the ACNET database. A summary of applications and experiences is given.

INTRODUCTION

The Instrumentation Group of the Accelerator Division at Fermilab deals with the development and maintenance of sensors, actuators, and signal processing hardware, as well as making the measurements available for accelerator monitoring or control. Many of our tasks involve the use of computers. The tasks using computers can be subdivided in the following categories:

- 1) *Modeling*: Models are used to determine the behavior of an accelerator-related process or to predict hardware performance for measuring accelerator parameters. Examples of these types of applications are beam position module frequency response, pulse propagation through hardware components and filter calculations.
- 2) *Development*: During the development of a hardware component (e.g., a VXI card or an analog processor) a prototype setup is needed to verify the operation of the component.
- 3) *Maintenance*: Testing and calibration of equipment or other hardware can be a repetitive task; automation of these tasks saves time and money. An example of such a task is the calibration of an RF module, in which a signal generator supplies a test signal and a digitizing scope measures the quality of the RF module's processing.
- 4) *Accelerator Operation*: A monitoring or control system takes data and provides results to the operators or a supervisory program. Such a system is always on-line and must run reliably and communicate over the accelerator network.
- 5) *Accelerator Diagnostics*: To diagnose the operation or confirm a model of the accelerator, a data-acquisition system must be set up to take the required data. After the measurements are taken, the system is no longer needed and its components can be used for other purposes.

In looking for a solution that can handle all or most of these tasks efficiently, we derived the following requirements for a general purpose instrumentation platform:

- 1) the programming environment should support fast prototyping for one-time only uses, include extensive graphic display capabilities to make interpretation of data easy, and include an analysis library to support modeling;
- 2) control capability for a wide variety of instruments (e.g., VXI/VME, GPIB, CAMAC);
- 3) support for off-the-shelf software and hardware;
- 4) high reliability for on-line applications;
- 5) capability to communicate with the control system.

While a VME-embedded computer with the capability to communicate with the control system was available, this was not chosen as a general purpose platform for several reasons. To set up the embedded computer, a programming expert familiar with the details of the communication interface was required (the expert might or might not be available) and the system was difficult to use for fast prototyping because it lacked drivers for various instrumentation and had no graphics support or analysis library for data presentation and modeling. In addition, the cost of a VME/VXI crate had to be included even if the instrumentation to be automated was GPIB or CAMAC. Instead, the commercial package LabVIEW was chosen because it does support fast prototyping and graphics display, comes with hardware interface to most instruments, including software drivers, and includes an extensive analysis library. At the time, LabVIEW was only available on the Macintosh computer, which was also the desktop platform in the Division, resulting in the Macintosh/LabVIEW combination as the instrumentation platform. The advantages of selecting a desktop computer as

* Operated by the Universities Research Association under contract with the U.S. Department of Energy.

the platform for running instrumentation applications is both the availability of software and hardware and that only one computer is needed to perform the regular personal desktop duties as well as bench-top instrumentation applications. The main disadvantage is that embedded computers typically have real-time operating systems designed for fast and reliable data-acquisition and control applications. Choosing a desktop like the Macintosh will mean that certain applications cannot be implemented.

To add the capability of communicating with the control system, an interface to the ACcelerator NETwork (ACNET) was written. The integration of LabVIEW with ACNET is described in the following section, followed by an overview of applications and experiences.

INTEGRATION OF LABVIEW WITH ACNET

Communication

The ACNET protocol associates data with physical devices, for example, the value of the voltage of a power supply. In LabVIEW, an ACNET device is associated with an analysis result, a variable, rather than a device. To retrieve data, the ACNET protocol requires that remote nodes have a task called RETDAT to handle requests for data readings, and a task called SETDAT to handle data settings. On the LabVIEW nodes, the tasks are implemented in C and run as completion routines. The implemented RETDAT task supports single-reply, multi-reply interval-based, and multi-reply event-based on a single message containing requests for data from one or more devices. The SETDAT task supports a single-set of data on a single message containing requests for data settings of one or more devices. The RETDAT and SETDAT tasks talk to a server task that was already written for the Macintosh as part of the Lineac control system (see [1]). The server connects the tasks to a tokenring or ethernet network and provides ACNAUX (ACNET AUXiliary) node statistics services, such as which tasks are connected, the number of messages handled, and uptime. The data accessible by RETDAT and SETDAT is stored in shared memory that the LabVIEW program accesses through a library of read/write routines for the different variable types (see [2]).

From the operator's point of view, the LabVIEW/ACNET interface makes a LabVIEW node look like just another node in the accelerator network, so that the standard console environment can be used to read out or control a LabVIEW application.

As more complex applications were built, the ability to be read or be set was extended with a new interface that enables a LabVIEW program to read and set devices on other nodes, for example, the reading out of beam energy or current store number, or the setting of a timing gate. This interface, TCPORt, connects LabVIEW as a client to the VAX TCPORt server which will, on behalf of the client, issue the proper ACNET requests. The data requested from nodes must be scaled according to the database entries. To avoid having to reimplement these data scaling services and database lookup facilities already available on the VAX, the TCPORt server performs these functions, including permission checks and activity logging, before requesting the settings or sending scaled data to the TCPORt client on the Mac. Because the client is completely implemented in LabVIEW itself, it can also run on the other computers on which LabVIEW runs.

Because the instrumentation platform uses a desktop computer, additional commercial communication software is readily available. Utilities like Timbuku (PC/MAC) and PlanetX (X-Windows) can remotely control a node before an ACNET console application has been written. These utilities display the screen of the remote Macintosh and give complete control as if it were the local computer. The utilities are also used after the system has become operational for diagnostic purposes by the system expert because it gives much more control over the node than a console application. Other often used communications utilities are Appleshare and FTP programs to up/download data and program files.

Database

The Fermilab control system utilizes a centralized database. When a control console makes a request for data for a particular device, the database matches the name of the device to look up on which node (e.g., the Mac running LabVIEW), the device resides and the SubSystem Device Number (SSDN) required by the node to identify the device. In addition, the database retrieves information about the size of the data and on how to transform and scale the data once retrieved from the node. To correctly retrieve the data, the database and the node must have matching information about the device. This is implemented by using one source, a spreadsheet table, to generate entries for the database and initialize the node. The table can be created and edited by a commercially available spreadsheet program like Excel. Each row represents a device entry, with each column representing an attribute of the device (see table 1).

6/16/94	Demo	Version 1.2										W. Blokland
Device name	SSDN #	Read/Write	Var type	Bytes/ elem	Elem/ device	Initial value	Update rate	Com Unit	Eng Unit	Long/ Short	Nota tion	Description
T:DEMTRAT	1	RW	I32	4	1	50	60	unit	step	short	dec	Demo step ratio
T:DEMSTP	2	RW	I16	2	1	1	60	unit	step	short	dec	Demo step number
T:DEMVAL	3	RW	SGL	4	1	1	60	unit	step	short	dec	Demo value
T:DEM	4	R	SGL	4	4000	0	120	unit	step	short	dec	Demo array
T:DEMSTR	6	R	STR	1	256	Hello	120	unit	step	short	dec	String
T:DEMTMP	7	R	SGL	4	1	85	600	Fahr	step	short	dec	TGC202 Room Temperature

Table 1. The device table used to initialize the LabVIEW/ACNET interface.

The table is used to initialize the ACNET interface each time the LabVIEW program is started. By doing this from a table instead of hardcoding this into the program, modifications to the device table do not require a recompilation of the code or involvement of an ACNET programming expert. A LabVIEW utility, Ac_MakeDabbel, is used to convert the device table to a database file with device entries. This way, a developer does not have to be familiar with the syntax of the database entries. The format of the device table is also suitable as documentation for what devices are defined for the node.

Development tools

To help a developer use the ACNET interface, a template of a LabVIEW/ACNET application is available. The template includes the initialization and termination of the ACNET interface and has a location on the diagram where the developer can insert the LabVIEW application (see figure 2). The developer only has to fill out the name of the device table, over which network the interface should communicate (or even both), and the name of the project. The template includes an error logging display and a statistics page. The statistics page shows the registered devices, the ACNET access activity as well as computing load on the node and the program versions of the ACNET interface. To help install the required software for the ACNET interface, an installer disk is available.

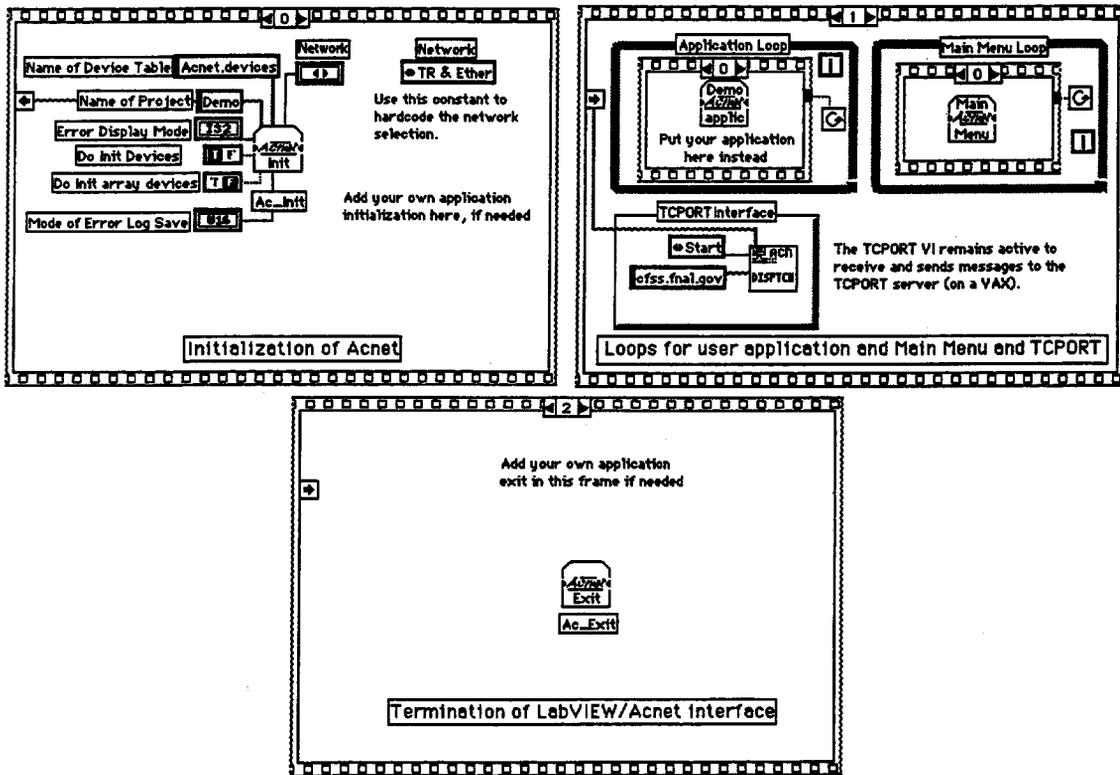


Figure 1. The Diagram of the Template for using the LabVIEW/Acnet interface.

To assist the development and documentation of LabVIEW system, the utility VI Hierarchy has been created (see figure 2). It displays the hierarchy of calling routines and the information about the operation of the routine. The utility allows the developer to browse through the hierarchy and print out documentation. Even though the graphical nature of LabVIEW automatically provides a graph of the data flow of each routine, a high level description of the functionality, much like the description of a C routine, is still needed to help others understand the program (see figure 3).

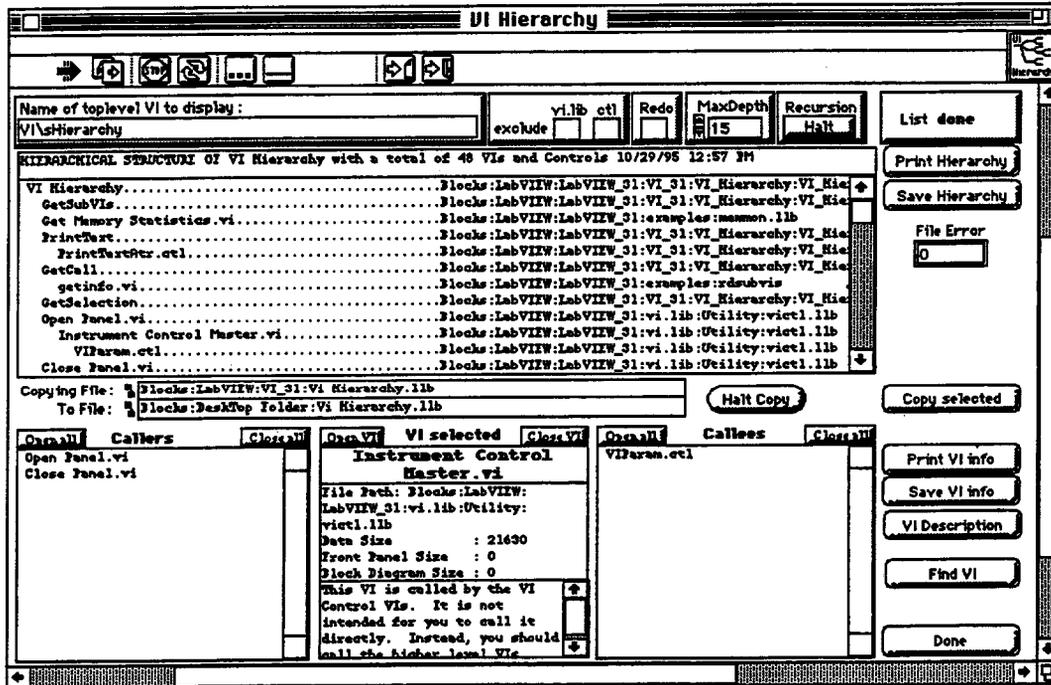


Figure 2. The Hierarchy Browser.

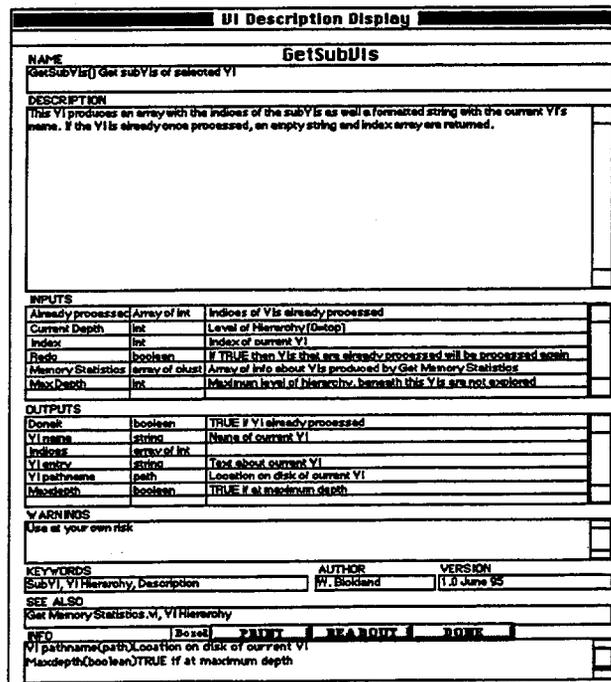


Figure 3. Documentation for a LabVIEW routine.

To make it possible to test or diagnose the LabVIEW application without requiring the use of the console environment, a LabVIEW utility, ParameterPage, was created to read out indexed devices, plot array devices and time plot a device. The ParameterPage utility uses the TCPOR interface to request the data from any ACNET node (see figure 4). The TCPOR utility also makes it possible to prototype a console application in the LabVIEW environment before writing it in the console environment.

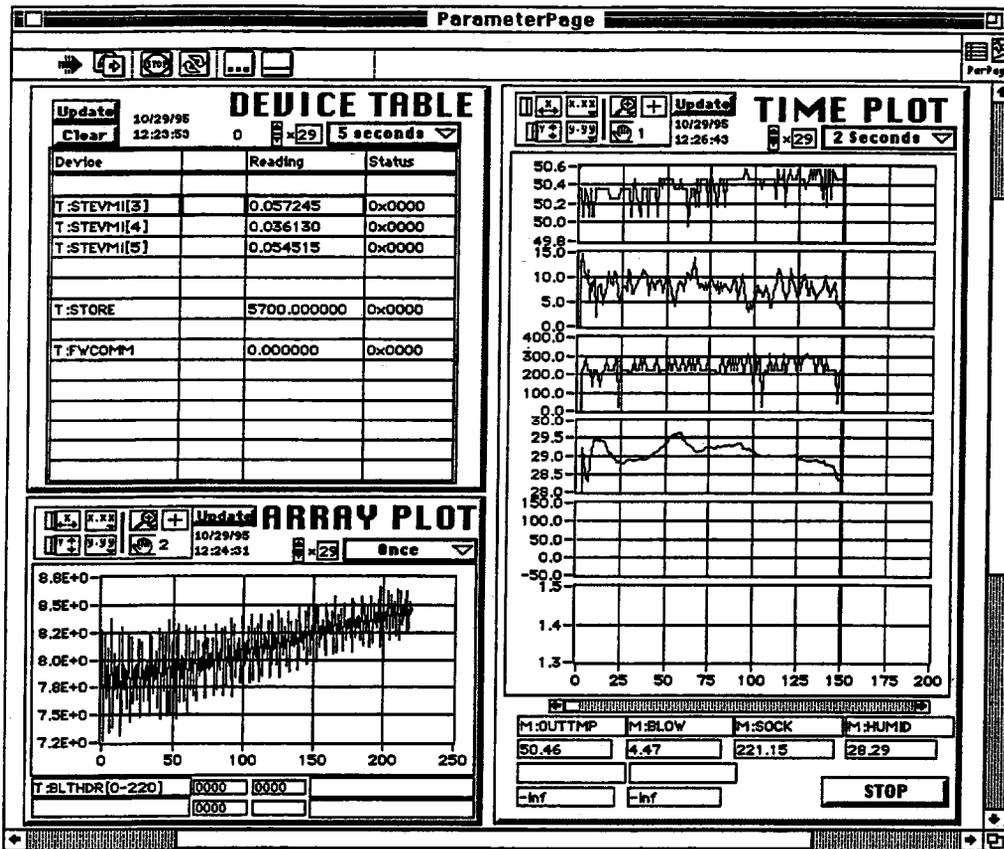


Figure 4. The LabVIEW parameter page.

In case a Macintosh system crashes and remote control is no longer possible over ACNET or with other means, the power to the computer can be cycled by an independent control channel using a CAMAC card. This proved to be very helpful during development and commissioning phases.

APPLICATIONS

A wide variety of applications have been made using LabVIEW. An overview of them is given in table 2. The table lists the name of and the use of the application. The next column of the table, labelled type, contains the abbreviations of instrumentation tasks listed in the introduction chapter. The operation of the application can be triggered by an accelerator event (e.g. beam injection), by the user, by a supervisory program, or the program repeating as fast as possible (cycle). The following two columns list the amount of data (in bytes) that must be acquired and how it is analyzed. The next two columns show the type of instrumentation and whether it was bought or developed in-house. The last column shows the time it takes to acquire, analyze, and present the data. This time gives a general idea about the time-frame of the applications, as the times are not directly comparable because different computers are used (68000 versus PowerPC Macintosh). Specific applications are discussed in [3,4,5,6], and [7].

Application	Use	Type	Trigger	Data Amount	Analysis	Instrument	Com-mercial	response time
Beamline Tuner	Injection tuning of steering magnets	Ac Op	Injection Tevatron	5 kb	N-L damped oscill. fit	VXI digitizer	yes*	5 seconds
Synchrotron Light Monitor	Transverse emittance measurement	Ac Op	Cycle	100 kb	2-D non-lin gaussian fit	framegrabber/HV supply	yes*	<1 s/bunch
Collision Point Monitor D0	D0 Collision point determination	Ac Op	Cycle	10 kb	Rectifying integrator	GPIB scope	yes*	90 sec^
Collision Point Monitor B0	B0 Collision point determination	Ac Op	Cycle	10 kb	Rectifying integrator	GPIB scope	yes*	75 sec^
Booster Ion Profile Monitor	Turn by turn transverse emittance	Ac Op	Injection Booster	2.5 Mb	Non-linear gaussian fit	VME digitizers	yes*	40 s/20k turns
MR Ion Profile Monitor	Turn by turn transverse emittance	Ac Op	Injection MR	8 Mb	Non-linear gaussian fit	VME digitizers	yes*	20 s/65k turns
Sampled Bunch Display	Long. emittance & int in MR/TeV	Ac Op	Cycle	10 kb	Integration & moments	GPIB scope	yes*	20 s/12 bunches
Accumulator Flying Wires	emittance measurement	Ac Op	User	20 kb	Non-linear gaussian fit	VME digitizers	yes*	20 sec/2 wires
Tevatron Flying Wires#	emittance measurement	Ac Op	Supervisory program	100 kb	Non-linear gaussian fit	VME digitizers	yes*	10 s/72 bunches
MR Tune monitor	Tune measurements	Ac Dia	Injection	1 kb	Fourier transform	GPIB signal analyzer	yes	5 sec
MR coupled Bunch	Coupled bunch mode measurements	Ac Dia	User	10 kb	two-point correlation	GPIB digitizer	yes*	15 sec
MR coupling	Transverse coupling	Ac Dia	User	10 kb	cross correlation	GPIB signal analyzer	yes*	10 sec
MR kicker profile	Kicker profile	Ac Dia	Injection	1 kb	Waterfall plot	VXI digitizers	yes*	5 sec
Accumulator Beamloss	Pager Alarm when beam drops	Ac Dia	Cycle	1 kb	comparison	modem	yes	5 sec
Digital Damper	Damping of coupled bunch modes	Dev	Initialization	1 kb	None	VXI modules	no	inits hardware
MECAR	status display and filter download	Ac Dia	User	10 kb	filter calculation	NA	NA	1-5 sec
High voltage power supply	Testing HV supply	Mnt	User	10 kb	Plot	GPIB scope	yes	10 sec
Beam Position Modules	Testing of RF modules	Mnt	User	2 kb	signal response	GPIB sig gen & scope	yes	30 sec
Beam Position Detector	Testing of sensor	Mnt	User	2 kb	signal response	GPIB netwrk analyzer	yes	10 sec
Bunch Spacing	Bunch arrival times at detector	Mod	User	10 kb	addition of delays	NA	NA	< 1 sec
Sampled Bunch simulation	Model of system's signal proagation	Mod	User	10 kb	FFT	NA	NA	< 1 sec
* timer cards are made in-house	^ most time spent in GPIB transfers		# under development					

Table 2. Examples of Applications of the Instrumentation Platform

EXPERIENCE

Reliability

We have had excellent results with the reliability of the hardware platform and the software. Several nodes had uptimes of about half a year only to be shut down by a site-wide power outage. None of the desktop hardware has failed in the several years that nodes have been on-line. During the development phase, the programs could hang or crash, but once these bugs had been removed (often incompatibilities of extensions) the systems ran reliably. On a downside note, on every upgrade of the software (e.g., new versions LabVIEW or Mac OS) we found that problems arise that affect the reliability. Therefore we are very careful in upgrading our software, especially new versions of the development environment.

Code sharing and documentation

The ability for code sharing is often the pride of a new programming paradigm. How did we do with LabVIEW so far? Code for instruments that was made available by the companies was used quite a lot, not always right out of the box but often with small modifications to be suitable for our applications. The Acnet interface code was used in all on-line applications and did not require any modifications. Many of our applications share the same analysis routine, a non-linear gaussian fit. Initially, a fitting routine coming directly out of LabVIEW's analysis library was used but this was adapted to our use and improved in speed (see [3]). Other code used to store data on disk (circular buffers), or pop-up menus has been reused some but not yet as much as we would like. In these cases it was very hard to find common ground for the different applications.

Development

We did find that programming in LabVIEW enabled more people with different backgrounds to set up a system. The graphical nature of LabVIEW saves one from making the typical syntax errors in text-based language and thus saves time. One of the most powerful debugging features is that one can run each routine by itself interactively (without having to create a main program) or, when run as a part of a program, view the parameters and results, possibly in graphical format.

Even though a LabVIEW program is a graphical diagram of data flow, it is still possible to do bad programming and make the code incomprehensible. Proper documentation and proper programming styles are still very important to complete the development with a maintainable system. We have put documentation on a WWW server to provide easy access and have been using it quite a lot ourselves during development. We are also standardizing the development cycle for LabVIEW systems. This is described in [8].

Software/hardware availability

As mentioned before, we were able to use a lot of the drivers supplied by the vendors of the instruments. In the case of the synchrotron light monitor, (see [3]), almost everything, including the software, was bought, from frame grabbers to high voltage power supplies, and was easily integrated using LabVIEW. In all cases of the on-line systems, the only in-house developed hardware were the timings cards (either CAMAC or GPIB) used with Fermilab-specific accelerator timing network. Because the platform uses a desktop computer, we were able to use utilities like screen snapshot takers, installer programs, and editors to help us document the development of the application.

The non-embedded computer

One of the advantages of having a non-embedded computer, but one that interfaces to the instrument or crate, is that it is easier to upgrade. The outdated desktop can still be used for many other purposes as long as speed doesn't matter. An embedded computer is rather a special purpose machine and it will be hard to find other uses. The desktop market also develops faster than the embedded computers and, because it a mass market, prices are typically lower. We were able to use desktops with PowerPC chips long before a PowerPC embedded computer was available. If your application uses VME or VXI modules, an embedded computer saves you space, but if you use a GPIB instrument and you don't need the crate other than to put the embedded computer in, a desktop computer will save you space.

One main issue to consider carefully is how one applies commercial products. If you buy a specific purpose computer, it is likely that the company is geared to support your type of application. If you use a desktop computer and desktop type application, the more you use it out of the mainstream use, the less support you can expect. So not only do you get less support from the company (they make their money of their mainstream users) but also you will have to do more work yourself to implement the features you want to add to the mainstream use. We did not want to go further than to write an ACNET interface and, at this point, have no plans to try and implement an application requiring a fast and deterministic feedback loop. The Macintosh OS and LabVIEW as well are not engineered for those purposes. We found though, that this did not limit us too much, since much of the real-time operations can be done by the ever smarter

instruments.

CONCLUSIONS

The LabVIEW/Macintosh combination with the addition of the ACNET interface has given us a general purpose instrumentation platform that is easy to use and able to implement a wide variety of applications, including those used for the continuous accelerator operation. We limited ourselves to applications that didn't require fast and deterministic feedback loops. Most of our applications provide results to operators, requiring only that the results are returned within the range of the operator's patience. We found that the integration of commercial software and hardware with ACNET enabled us to buy off-the-shelf products, thus reducing time and cost and making life a lot easier.

REFERENCES

- [1] B. Peters, "Macintosh Parameter Page", (to be published).
- [2] W. Blokland, "An Interface From LabVIEW To The Accelerator Controls Network", Proc. of Accelerator Inst. Workshop, Berkeley, USA, 1992, pp. 320-329.
- [3] A. A. Hahn and P. Hurh, "Results From An Imaging Beam Monitor In The Tevatron Using Synchrotron Light", HEACC'92, Hamburg, Germany, July 1992, pp. 248-250.
- [4] W. Blokland, "A VXI/LabVIEW-based Beamline Tuner", PAC'93, Washington, USA, 1993.
- [5] E. Barsotti, "A longitudinal Bunch Monitoring System using LabVIEW and high-speed oscilloscopes", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 466-472.
- [6] J. Zagel, "Booster Ion Profile Monitor using LabVIEW", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 384-390.
- [7] J. Steimel, "Fast Digital Damper for the Fermilab Booster", PAC95, 1995.
- [8] W. Blokland, "A LabVIEW-based Accelerator Instrumentation Platform", EPAC94, London, GB, 1994.

STATUS of DANTE, the Control System for DAFNE

L. Trasatti.

INFN, Laboratori Nazionali di Frascati
P. O. Box 13, 00044 Frascati (Rm) ITALY

Abstract

DANTE (DAΦNE New Tools Environment) is the control system for the DAΦNE Φ-factory under construction at the Frascati National Laboratories of the Italian Institute for Nuclear Physics (INFN).

Commercial components have been used extensively: Macintosh™ computers are used throughout the system and LabVIEW®[1] has been chosen as development environment.

Three consoles and ten peripheral computers out of 60 have been installed successfully. The system is now ready for the commissioning of the first part of the accelerator complex.

I. DAΦNE

The DAΦNE accelerator complex [2] consists of a two ring colliding beam Φ-Factory, a 510 MeV e⁺/e⁻ injector for topping-up and of a commercial LINAC operating at the same energy.

From the point of view of the peripheral hardware, the accelerator complex consists of about 1000 devices, which will be controlled by 5 consoles and 60 peripheral CPUs.

II. SYSTEM STRUCTURE

The system is divided into three levels (see Fig. 1):

PARADISE (PARAllel DISplay Environment) is the operator interface level. Several consoles, all equivalent, communicate with the rest of the system through high-speed DMA buses and fiber optic links.

PURGATORY (Primary Unit for Readout and GATing Of Real time Yonder) is the second level of the system. It contains two CPUs:

- CARON receives commands from the consoles and forwards them to the appropriate peripheral CPUs. CARON also uses a polling mechanism to check the status of the peripheral CPUs, relaying to the consoles any error and warning messages.

- LOGGER is in charge of saving the status of all the system elements as well as the log of commands and errors at definite time intervals or at operator request.

HELL (Hardware Environment at Low Level) consists of 60 CPUs distributed around the equipment in VME crates. Each CPU (DEVIL) reads the status and values of a set of elements, checking for any abnormal events and generating appropriate error messages. Every CPU contains a memory that is part of the Real Time database of the whole system. There is a record for every element and this is updated by the DEVIL whenever a significant change in the parameters is detected. These memories are accessed directly from the consoles through direct memory access, taking advantage of high speed buses which do not require any protocol, since all communication is effected through point-to-point links, without a network structure.

Interrupts are not used anywhere in the system, increasing reliability and ease of debugging.

The structure of the system has been described in detail at ICALEPCS '93 [3].

III. LABVIEW

We have chosen LabVIEW as a development environment, because of the many advantages in human interface design and ease of programming it provides. After we decided to use it on the consoles, we saw that it would also be extremely useful for the peripheral CPUs and we decided to build our own VME CPUs using the logic board of a commercial Macintosh (LCIII) and designing an interface to VME and VSB. This allowed us to have LabVIEW running on the whole system, with obvious advantages in simplicity and coherence.

LabVIEW has proven reliable and robust and perfectly capable of coping with the requirements of a large system.

110 MBytes of software have been written so far, and we expect the system to grow while new parts of the accelerator complex start functioning.

One of the strongest points in favour of this environment is the large amount of debugging facilities available. These we have found particularly useful during installation.

Speed is not one of the strongpoints of LabVIEW. However we have very rarely met problems where this is a limitation. In these few cases, such as saving a large cluster of data to memory, we took advantage of the possibility of implementing CINS, which are dedicated subroutines in C that use all of the power of the machine. This limitation will be much less important when machines with the new PowerPC CPUs will become available.

IV. MACINTOSH

The choice of the Macintosh computers has been helped by the large amount of expertise available in INFN on these machines. However, we have had no reason to regret the choice. A very large amount of extremely good and cheap software is available for the Macintosh and we have been easily able to take advantage of it.

TDM (The Diskless Mac) has been used to download the OS and the dedicated applications to all the peripheral CPUs, via ethernet. This makes the system much more controllable, since all of the peripheral software can reside on a single server.

TIMBUKTU is a program that allows one to take complete control of a remote machine (video, keyboard and mouse) through ethernet. We have extensively used this during debugging, to check on the behaviour of the peripheral CPUs from the consoles.

The migration path to the Power PC, which we have not yet used, gives the system the possibility of evolving easily with the growing hardware market.

In one particularly demanding case, where the time requirements were very strict, we have written a routine that effectively turns off all OS interventions on the CPU, giving the application program full control.

V. HIGH LEVEL SOFTWARE

We have developed an interface between LabVIEW and the "physics" environment, which consists essentially of a large body of FORTRAN programs written by the accelerator physicists. Two methods of operation are possible:

- to incorporate FORTRAN routines in a LabVIEW VI to perform complex calculations without rewriting the code;
- to allow access to the Control System from a FORTRAN program, to recover data from the Real Time Database on the peripheral CPUs and send commands to the system.

A large body of FORTRAN routines has been integrated and is available to the programmers, to insure software coherence throughout the many high-level applications that will be written. C is also available to the programmers.

VI. STATUS

We have installed the part of the system necessary for the commissioning of the LINAC, which will be the first part of the accelerator complex to start functioning.

Three consoles are operative, together with the second level (Purgatory) and ten out of the sixty peripheral DEVILs.

Work is in progress on the installation of the remaining parts of the system, firstly the transfer lines, then the Accumulator and finally the two rings. We expect to be able to anticipate the needs of the accelerator physicists throughout the whole commissioning stage.

VII. ACKNOWLEDGEMENTS

We would like to thank the Accelerator Group of the LNF for continuing discussions and encouragement.

VIII. REFERENCES

- [1] LabVIEW® National Instrument Corporation, 6504 Bridge Point Parkway, Austin, TX 78730-5039
- [2] G. Vignola et al., talk presented at the San Francisco Particle Accelerator Conference, May 1993.
- [3] G. Di Pirro et al., DANTE: a control system based on Macintosh and LabVIEW, talk presented at the Berlin ICALEPCS '93.

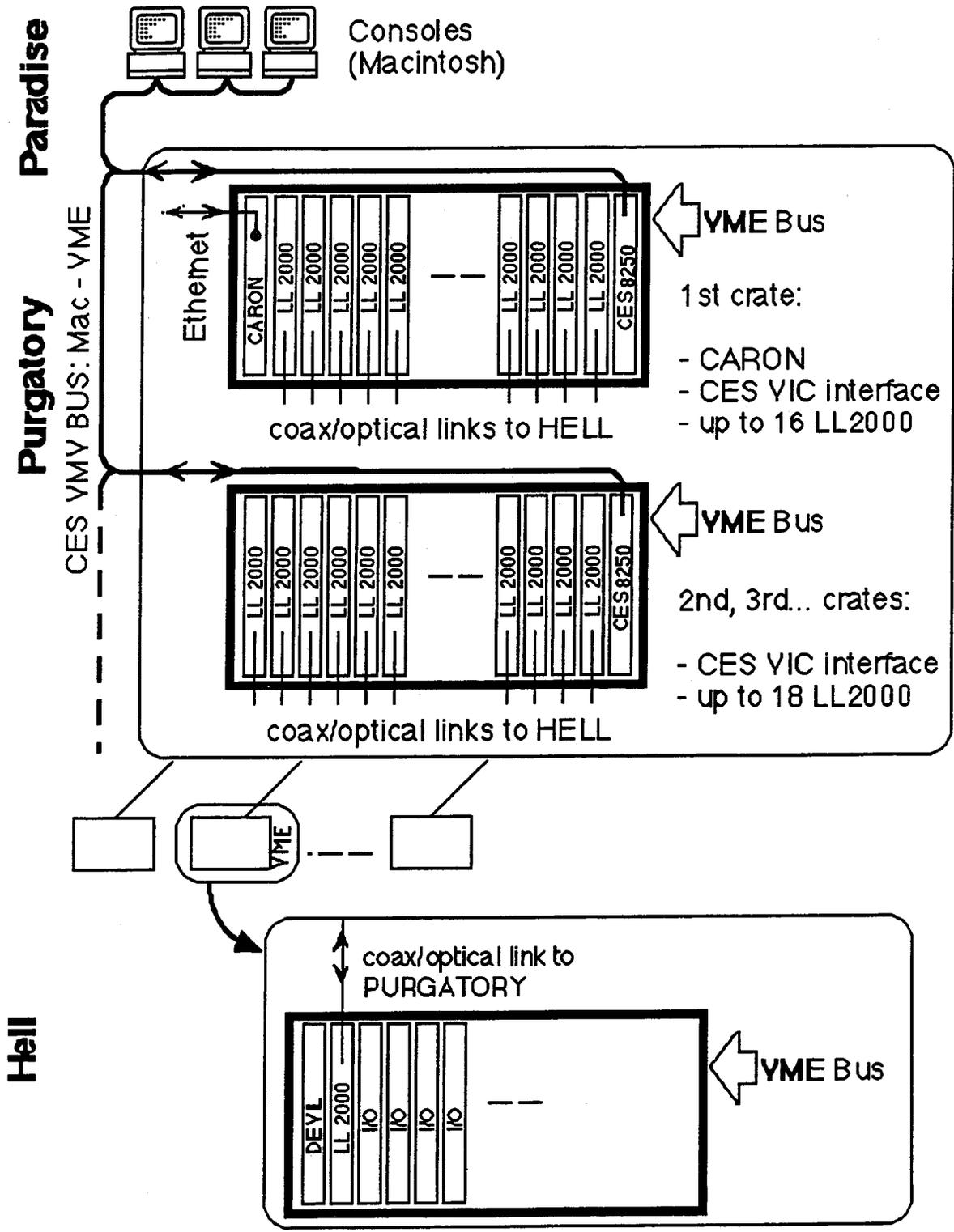


Fig. 1: Control System Schematic Diagram

The Architecture of Vaccess

Version 3.0

M.D.Geib

Vista Control Systems, Inc.

I. INTRODUCTION

This paper discusses some of the features of the Vaccess V3 architecture and the related functionality provided by the architecture. The initial release of V3 is scheduled for early next year and will include at that time about four man years of effort.

Philosophy

The basic philosophy of the Vaccess V3 project was to produce a product that is backward compatible, is easily portable to a variety of computing platforms, supports long term development, and is of the highest quality possible.

Beyond that, the underlying philosophy of Vaccess, and Vsystem as a whole, is to provide a complete set of tools which can easily be used to implement control and monitoring systems. The customer should be able to configure the system as he chooses and not be restricted by the architecture of the toolset. To that end Vsystem supports all the components on all the supported platforms, with the one exception that Vdraw is not supported on VxWorks. Specifically, Vaccess databases can be hosted on all supported platforms along with the complete application programming interface (API). Planned support for the initial release of V3 includes OpenVMS, VAXELN, VxWorks, Digital UNIX, SUN Solaris SPARC, and HARRIS Power UNIX. The initial release will include equivalent functionality and full interoperability between all supported platforms. Internally, Vista also has V3 running on SUN Solaris x86 and UNIXware.

Objectives

The following are the major objectives established for the Vaccess V3 project.

Backward compatibility

Vaccess V3 must be backward compatible with previous versions of Vaccess.

Inter-version support

Beginning with V3, Vaccess will support networked systems that include platforms running different versions of Vsystem. This mixed version support allows for large multi-node systems to be upgraded a single node at a time, without the requirement of taking the entire system down to upgrade all systems running Vsystem.

Portability

One of the main reasons for embarking on the V3 project was to support additional computing platforms. The previous version, V2, supported OpenVMS and VAXELN. The new targets

identified for future support included UNIX, VxWorks, and Windows NT. With such a wide variety of platforms to support, one of the most important objectives of the project was to produce a product which is easily ported to new platforms.

Maintainability

V2 was difficult to maintain. The system had a very high level of coupling and low cohesion, both characteristics of systems that are difficult to modify and maintain. From the very beginning V3 was designed and implemented with maintainability in mind. The V3 re-engineering project made extensive use of formal analysis and design methods.

Support for future enhancements

Because of the high level of coupling, for one thing, it was becoming almost impossible to add or enhance functionality in V2. Support for many new features was included in the design of V3, but in addition, the design and implementation of V3 allows for new functionality to be easily added in the future.

II. OVERVIEW

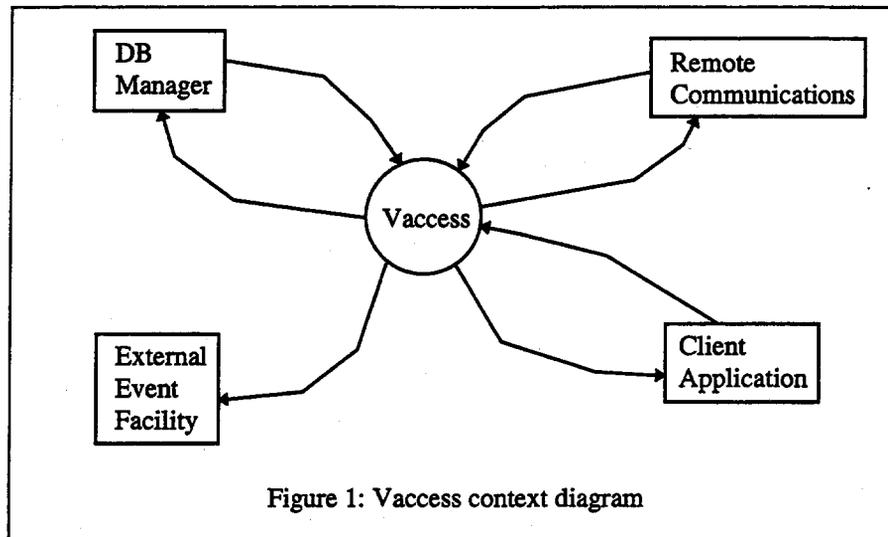


Figure 1: Vaccess context diagram

Figure 1 shows a simplified data flow diagram for the context of Vaccess. This diagram shows the relationship between Vaccess and the other facilities it interacts with. As can be seen in the diagram the remote communications facility is not part of Vaccess but is implemented as a separate facility, to make it easy to replace or modify. Likewise, the external event facility, which handles the delivery of Vaccess events is implemented as a separate facility to minimize any dependency of Vaccess on how this facility is implemented. The box labeled *DB Manager* is a new facility in Vaccess V3. The *DB manager* provides such services as mapping a database into memory, managing the database definitions and locations, and controlling access to databases. Since many of the *DB manager* services depend on the OS services provided by the platform it was decided to implement them in a separate facility that can be tied more closely to the platform.

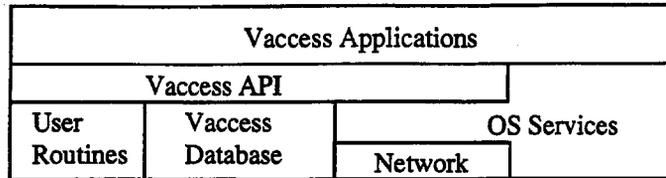


Figure 2: Vaccess application layers

Vaccess structure

Vaccess is made up of a real-time database and an API for accessing databases. Figure 2 shows the relationship between a Vaccess application, the Vaccess API, the available OS services, and the Vaccess database. The layer labeled *User Routines* are the handlers and conversion routines invoked directly by Vaccess, not the Vaccess event callbacks. The API transparently supports the access of databases hosted on remote platforms. In addition, the API provides for synchronized access to the data structures that make up the real-time database when access is local (that is, on the platform which is hosting the database). An active database consists of a number of data structures that are mapped in memory.

Supported configurations

There are no restrictions on the number and type of platforms which participate in a Vaccess based system. Any platform supported by Vaccess is equivalent in every way with all other supported platforms.

The number of databases supported on a single node is limited only by the amount of memory available. Any single process can connect to 65536 databases, local or remote in any combination. Each Vaccess V3 database supports 65536 channels. These are not hard limits, and can be increased in a future version of Vaccess.

III. DATABASE STRUCTURE AND FUNCTIONALITY

A Vaccess V3 database is composed of a database header, sorted tables, database channels, a table for storing information about each process and thread connected to a database, and two dynamic memory pools. The memory pools are used for event support, dynamic channel creation, and for storing channel fields with dynamic sizes.

Channel Types

The type of a Vaccess channel is related to the data type of the values stored in the channel. A channel is made up of many different fields that can have many different data types.

Vaccess supports channels of the following types:

- integer, single and double precision real values
- arrays of integer, and single and double precision real values
- character or single byte arrays
- time and time arrays
- binary and binary arrays

All channel types include both internal and external representation. All channel types also include support for timestamping.

Channel Functionality

All channels support the same basic functionality. Some additional functionality is provided in specific types; this functionality will be noted when present.

- A Vaccess event is generated when any field within a channel is modified.
- A Vaccess event is generated when any channel changes state (for example alarm state or clipping state).
- A standard set of fields for naming and describing the channel.
- A set of fields that specify any clipping limits for the channels value.
- A similar set of fields is provided to specify any limits for displaying the channels value.
- A delta value that specifies the required change in a channels value before it is considered significant.
- Alarm support for a user specified number of alarm levels. Included in the alarm block is a string field for specifying an alarm label or alarm message for a channel. A channel's alarm levels can also be specified to be relative to another channel's value, as either a percentage or a fixed offset. Integer channels can be configured to enter an alarm when the channel's value matches a specified match alarm value.
- A hardware block can be configured to include any number of hardware related parameters. It also includes a number of text and integer data fields that can be used to identify any hardware associated with the channel. The hardware block also contains the name of the hardware handler routine which is invoked by Vaccess during a read or write to the channel. The handlers function is to interface with hardware to read or write the database value.
- A conversion block can be configured to include any number of conversion related parameters. A conversion block is included in the channel when a user specifies a user supplied conversion routine for converting a channel's external value to the internal, and vice versa. The conversion also contains the name of the user-supplied conversion routine invoked by Vaccess to perform conversions. Both external-to-internal and internal-to-external conversion routines can be specified. As an alternative, a channel can also be configured to include a linear conversion by specifying a slope and intercept.

Support for threads

V3 of Vaccess supports the use of threads. Earlier versions prevented the use of threads.

Threads can be used to achieve high levels of concurrency in a program. For access to remotely hosted databases, threads can be used to improve response to remotely executed routines.

Polled or Event Driven

Vaccess supports the development of polled or event driven systems, or a mixture of the two methods.

Platform independent data

All files produced and used by Vsystem are platform independent. Database files can be generated on one platform and used on all supported platforms. Likewise, Vdraw screen files can be produced on one platform and used on all platforms.

Flexible and efficient

Vaccess V3 supports a very flexible database configuration. When defining a database the user has control over what structures and features are included in each channel. For channels that require a minimum of functionality, the memory resident representation of the channel is very small, consuming a minimum of memory resources. Only channels which are explicitly defined to include optional features and structures consume the resource to provide those requirements. In addition, there are no limits on the number of fields included in a channel's definition. For example, a channel can be defined to provide 1 or 1000 hardware parameters; the only practical limit is available memory.

Vaccess V3 also supports the creation of database channels at runtime, while a database is active. A user program can create a new channel without shutting down the entire system and re-activating a new database. Other processes can ask to be notified with a Vaccess event when a new channel is created.

A user can now create sorted tables in addition to the standard tables provided by Vaccess. Vaccess provides a sorted table that holds the channels in alphabetical order and one that sorts the channels based on hardware block values. The user can create additional tables that are sorted using routines supplied by the user. The user sorting routine can access any channel data to use in the sorting.

IV. NETWORK SUPPORT

Vaccess provides the network support that allows for systems to include multiple nodes. Systems based on the Vsystem tools can be a single computer or many computers. The network support is totally invisible to all the applications that make up a system. The low level network support has been implemented in modular way to make it easy to support different network transports in the future.

Transparent database location

Vaccess databases can be hosted on any platform supported by Vsystem. The actual location of a database is not fixed and easily changed. All of the Vsystem tools, as well as all user developed programs, function the same whether the database is located locally or remotely. Complete systems can be developed without concern for which platform will ultimately host the real-time database(s). During development, each developer can have a private database to eliminate problems that might arise if multiple developers were using and modifying a single database. Once the system is ready, all the programs run, unchanged, by simply changing the database definition(s) to point to the correct database.

Support for threads

As mentioned above, Vaccess V3 supports the use of threads. For programs that use threads to increase concurrency the same advantage is gained for remote access to a database. Each thread within a program establishes an independent communications channel to the platform hosting the database. Independent communication channels for each thread within a process allows for multiple remote functions to execute in parallel, improving the overall response to remote access.

Heterogeneous platform support

There are no restrictions on the type of platforms that participate in a networked system based on Vsystem. Any of the supported platforms can communicate and interoperate with any other supported platform, in any combination or mixture.

V. EVENTS

Vaccess events can be generated on almost any database access or change of state. As with many of the Vsystem tools, a user program can request to be notified when any event occurs. The request for notification specifies the name of a user supplied routine which is the event's callback, an optional user specified argument which is passed to the callback routine when invoked, and a specification for the event. Event specifications can be very specific or general in nature so that a single callback routine can handle many different events. When the callback is invoked it is passed a structure that includes the time of the event, the specific event that occurred, the user's argument, and the requested channel and database data.

User specified event data

When an event request is made, the user can specify what channel data to include in the data passed to the callback routine. For example, if a channel goes into an alarm state, the user may need to know the current value of the channel, the current alarm levels, and maybe the internal value of the channel. In previous versions of Vaccess the callback routine would have to read all this information from the channel, except the channel value, with additional API calls. In V3 any channel data can be passed to the callback routine. Eliminating extra API calls improves the performance of the callback routine, especially when the database being accessed is hosted remotely.

Time modified events

Requests for event notification can now include a number of time related options. The request can specify that the delivery of an event be delayed by a specified delta time or until an absolute time. For change of state events, like alarms, if the channel's state again changes before the delay time expires, the original event is canceled and never delivered. This feature might be used to prevent numerous alarm events from being delivered on a channel that is going through a transition which results in the value moving in and out of alarm rapidly.

An event notification request can also specify the minimum time that must pass between the delivery of an event. This action effectively throttles the delivery of an event. For example, if a channel's value is being changed every 100 milliseconds there is no need to deliver events to a Vdraw screen at that rate. Vdraw can specify some time interval at which the events can be delivered to it for display update. This feature is very effective in reducing the load on a system that would normally generate events at a high rate.

VI. REQUIRED OS FACILITIES

Vaccess V3 was designed to port to a wide variety of platforms. To help reduce the work of porting, a minimum set of OS services have been used. This also makes it easy to determine if a new platform will support Vaccess. Following is the list of required services or facilities that a platform must provide to support Vaccess.

threads

V3 of Vaccess supports and makes use of threads. Threads have been used to minimize the use of platform-specific services and facilities. For example, the event callback routines are now executed as a separate thread in a user's program. Earlier versions of Vaccess used OpenVMS AST and VAXELN event services to asynchronously deliver events.

global shared memory

The Vaccess database is a memory resident set of data structures. All local processes that access the database map the database through the API calls. In order for more than one process to map the database, the platform must provide global shared memory services.

broadcast signaling

In order to minimize the process specific information stored in a Vaccess database, it was decided to use a broadcast type signal to inform a process's event processing thread when a new event had been delivered to it. When this signal is generated, all event processing threads in all processes connected to a database are notified to check for a new event.

VII. SUMMARY

The architecture of Vaccess V3 is the result of achieving the objectives established for the project. Those objectives include inter-version support, portability, maintainability, and support for future enhancements. In addition, some of the design decisions were the result of the requirement that V3 be backward compatible with previous versions.

The portability of V3 has been proven by the variety of platforms on which it is already running. Likewise, it has already proven to be simple to maintain. The backward compatibility has been verified by porting many applications from the previous version of Vaccess to V3 with almost no changes to the source required.

Outsourcing the Development of Specific Application Software using the ESA Software Engineering Standards:

The SPS Software Interlock System

B. Denis

CERN - 1211 Geneva 23 - CH

ABSTRACT

CERN is considering outsourcing as a solution to the reduction of staff. The need to re-engineer the SPS Software Interlock System provided an opportunity to explore the applicability of outsourcing to our specific control environment. The ESA PSS-05 standards were selected for the requirements specification, the development, the control and monitoring and the project management. The software produced by the contractor is now fully operational.

After outlining the scope and the complexity of the project, a discussion on the ESA PSS-05 will be presented. The success factors and the difficulties of development under contract will also be discussed. Finally the maintenance aspect and the impact on in-house developments will be addressed.

INTRODUCTION

Outsourcing software development is being considered as a solution to the reduction of personnel at CERN in the coming years. The major goal of the SPS Software Interlock System (SSIS) project was to assess the suitability of this approach for the development of specific accelerator application software.

Today the SSIS project is finished, the application software has been successfully developed by an Italian software company and it is fully operational.

This paper shortly describes the SSIS project and explains how the use of ESA PSS-05 Software Engineering Standards [1] made the outsourcing process possible. It identifies the limitation of PSS-05 for developments under contract and summarizes the major success factors.

THE SSIS PROJECT

The system characteristics

The SPS Software Interlock System has two major functions. First, it provides a mechanism for application software to inhibit the beam in the SPS. Second, it monitors more than 150 pieces of equipment, makes sure that their state corresponds to the intended mode of operation and stops the source of particles in case of danger to costly equipment. This software system needs to be very reliable and available 24 hours a day.

Another characteristic of the system is that it is deeply integrated in the SPS/LEP control system and has to rely on many in-house developments. The hardware part of the system has been developed at CERN, the SSIS software communicates with other home-made software systems ([2],[3]) and the user interface has been developed using tools made at CERN ([4]).

Although not concerned with beam dynamics, the system contains knowledge of the properties and functionality of accelerator equipment and is thus specific to CERN.

As part of the SPS Control System Migration Project, the SSIS project was the re-engineering of an existing subsystem. There was a very strict dead-line for the replacement of the system. The first part of the system was to be ready 6 months after the development commenced and was mandatory for the SPS accelerator start-up in March 1994.

The project phases

1. Problem definition

This phase covered the definition of the scope of the project, the specification of the requirements, the analysis of the possible customer-supplier relationship and the tendering process. ESA PSS-05 standards were evaluated and selected during this phase of the project. The phase ended with the contract being awarded to an Italian company.

2. Main development and first operation

During this phase, the selected contractor developed the first version of the SSIS software based on the user requirements stated in the contract. The software was delivered to CERN in two parts. The first delivery dealt with the mandatory functionality of the system required for the SPS start-up in March 1994. The second part of the system was delivered and accepted at the beginning of May i.e. one month and an half late compared to the original planning.

The second part of the system was installed in parallel to the old existing system. The SSIS system database was gradually introduced and the performance was monitored up to the end of 1994.

3. Evolutive maintenance

Based on a first usage of the system, new requirements were identified. An updated version of the specification was produced and the original developer was asked to make an offer to implement the enhancements.

The offer from the contractor was accepted and the acceptance tests of the second version of the SSIS software took place in January 1995.

HOW DID ESA PSS-05 IMPROVE THE OUTSOURCING PROCESS

The choice

During the early phases of the project it became clear that there was a need for a well-defined terminology and for a rigorous way to structure the project.

After evaluating different software engineering standards and methodologies, ESA PSS-05 standards were selected for the following reasons:

- these standards are international in the sense that they do not favor any national standard or method;
- they are rather complete and they cover every phase of software development;
- many ESA suppliers in Europe are already familiar with these standards.

Definition of customer-supplier relationship

Potential conflicts between CERN and external suppliers were identified at the beginning of the project: interpretation of the requirements, high cost of requirements changes, unclear project completion criteria. ESA PSS-05 standards have been used to reduce the risk of such conflicts by defining as precisely as possible the relationship with the supplier.

In this regard, the standards provided guidance to specify:

- precise and structured user requirements;

- systematic acceptance tests using structured requirements;
- clear project phases and milestones;
- project functions including project management, verification and validation, and quality assurance.

Maintenance

One initial objective of the project was to avoid becoming subordinate to the contractor for maintenance. The advantage of the ESA standards where maintenance is concerned are numerous.

First, the end-of-phase documents (functional analysis, design documents, etc.) were rigorously reviewed by CERN at the end-of-phase milestones defined by the standards. This allowed CERN reviewers to strictly monitor the progress of the project but it also provided CERN technical reviewers with the knowledge of the system allowing them today to diagnose problems and fix minor bugs.

In addition, the standardized documentation should allow anybody, including other contractors, to maintain the software. This allowed CERN not to become dependent on one supplier and if the contract was prematurely canceled by either party, at least the documentation issued since the beginning would be available.

Limitations

Although the usage of PSS-05 was a key success factor, the following limitations are worth noting in the use of these standards in the frame of information system procurements:

- no support for the tender process,
- no guidance on how to allocate customer and supplier responsibilities,
- no guidance for tailoring the standards to the nature (i.e. size, criticality) of the project,
- no guidance on how to use the standards for life cycle approach other than the waterfall approach (i.e. prototyping evolutionary approach, iterative approach).

OUTSOURCING SUCCESS FACTORS

ESA PSS-05 standards made it possible to have well-defined contractual relationship. The quality of the contractual relation together with a deep involvement of CERN staff members allowed a real mutual cooperation which is probably the main reason for the project success.

The success factors can be summarized as followed:

- well-defined terminology,
- precise and structured specification of the product allowing rigorous acceptance tests, precise specification of the project activities and the responsibility for each activity (these activities are not limited to the development activities, but include project management, quality assurance, and verification and validation),
- well-defined customer-supplier interface reduced to a few people in both organizations, organization of joint reviews in order to monitor the project progress but also to technically validate the analysis, the design, the code and the associated documentation,
- technical competence of customer staff,
- systematic recording of the discussion results,
- establishment of a good communication with the supplier based on a mutual trust and an effort to understand the other party's problems.

CONCLUSIONS

ESA PSS-05 standards provided a means to specify a clear customer-supplier relationship by addressing not only the product but also the procedures required for the development. Using a methodology and applying software quality principles has a cost. Nevertheless, the quality of the product obtained in terms of documentation, completeness, reliability, and availability has proven up to now that this cost will surely be positively balanced by the cost of operation and maintenance of an in-house solution.

In a research environment one shot developments (the so-called waterfall approach) are rarely satisfactory. A prototyping or an evolutionary approach is often more suitable. These approaches are more realistic but the customer-supplier relationship becomes more complex and difficult to define. In addition, software projects are not always pure development, they are often migration, maintenance or feasibility studies. There is obviously the need for a framework larger than PSS-05 in order to support complex procurements otherwise only simple, well-known and well-defined systems will be considered as candidates for outsourcing; such projects are very rare in our environment

The customer must be actively involved in the customer-supplier relationship. This is required not only to have successful projects but also to prevent the customer from becoming too dependent upon its supplier. New skills are thus required from CERN staff members and suitable training covering requirement elicitation, project management and communication skills should be provided.

ACKNOWLEDGMENTS

The author would like to acknowledge K.H. Kissler and R. Lauckner for their support throughout the project. He would also like to thank F. Roeber who acted as an active reviewer and technical expert, and to C. Despas who really involved himself in the project as an active end-user.

REFERENCES

- [1] ESA Software Engineering Standards, PSS-05-0 Issue 2, European Space Agency, Feb 1991.
- [2] P. Charrue et al., The equipment access software for a distributed UNIX-based accelerator, Proc. ICALEPCS'93, Berlin, Germany, 1993.
- [3] M. Vanden Eynden, SPS/LEP Software Synchronization Mechanism V1.0, CERN note SL/Note 93-71(CO) (1993).
- [4] Tarrant, M. Vanden Eynden, SL X-Window Migration Project: Xsl Library and UIL Templates, CERN note SL/Note 93-66(CO) (1993).
- [5] B.Denis, Outsourcing Specific Accelerator Application Software to Industry - A Practical Experience, CERN-SL-95-72 CO (1995).

Control Theory with Applications to Accelerators

John D. Fox and Haitham Hindi
Stanford Linear Accelerator Center

Our talk attempted to serve as an introduction to important concepts of modern control theory; it was tutorial in nature. The following references should be interesting to those who want to expand their understanding of modern control theory concepts.

- (1) Franklin, Powell, Naeini, "Feedback Control of Dynamic Systems," Addison Wesley
- (2) Franklin, Powell, Workman, "Digital Control of Dynamic Systems," Addison Wesley
- (3) Kailath, "Linear Systems," Prentice Hall
- (4) Kailath, "Lectures on Wiener and Kalman Filtering", Springer Verlag
- (5) Bertsekas, "Dynamic Programming," Prentice Hall
- (6) Meditch, "Optimal Linear Filtering and Stochastic Control," Prentice Hall
- (7) Hindi, "Control Theory with Application to Accelerators," lecture notes, US Particle Accelerator School, Duke University, January 1994.

Adaptive, Predictive Controller for Optimal Process Control*

S.K. Brown,^a C.C. Baum,^b P.S. Bowling,^a K.L. Buescher,^b V.M. Hanagandi,^b
R.F. Hinde, Jr.,^c R.D. Jones,^d W.J. Parkinson^c

^a Accelerator Operations and Technology Division

^b Applied Theoretical Physics Division

^c Engineering Sciences and Applications Division

^d Center for Adaptive Systems Applications, Inc.

Los Alamos National Laboratory, Los Alamos, NM 87545

ABSTRACT

One can derive a model for use in a Model Predictive Controller (MPC) from first principles or from experimental data. Until recently, both methods failed for all but the simplest processes. First principles are almost always incomplete and fitting to experimental data fails for dimensions greater than one as well as for non-linear cases. Several authors[1] have suggested the use of a neural network to fit the experimental data to a multi-dimensional and/or non-linear model. Most networks, however, use simple sigmoid functions and backpropagation for fitting. Training of these networks generally requires large amounts of data and consequently very long training times.

In 1993 we reported on the tuning and optimization of a negative-ion source using a special neural network[2]. One of the properties of this network (CNLSnet), a modified radial basis function network, is that it is able to fit data with few basis functions. Another is that its training is linear resulting in guaranteed convergence and rapid training. We found the training to be rapid enough to support real-time control.

This work has been extended to incorporate this network into an MPC using the model built by the network for predictive control. This controller has shown some remarkable capabilities in such non-linear applications as continuous-stirred exothermic tank reactors and high-purity fractional distillation columns[3]. The controller is able not only to build an appropriate model from operating data but also to train the network continuously so that the model adapts to changing plant conditions.

The controller is discussed as well as its possible use in various of the difficult control problems that face this community.

INTRODUCTION

Remarkable progress has been made during the last several decades in the control of many different kinds of processes. A large body of theory has been developed to aid in this venture. "Classical" control theory deals well with linear processes and even with some non-linear processes as long as they are unidimensional. The problem is that there are no processes that are truly linear. All processes demonstrate some non-linear characteristics. How then can we say we have made remarkable progress? The control "surfaces" of many processes, although non-linear, are treated as approximately linear, i.e., they contain areas that are very nearly linear that are connected by discontinuities. As long as control is maintained within these very nearly linear areas, the process can be controlled. Approximate methods have been developed that provide excellent control of such processes. However, processes that demonstrate true non-linearity and have more than one dimension turn out to be not only in the majority but represent the most interesting, i.e., those processes for which good control would be most beneficial. Many processes in the chemical industry fall into this category. The optimal tuning of a surface-plasma negative-ion source is another example. The ion source tuning experiment was discussed at the last ICALEPCS conference[2]. Several authors[1] have suggested that the use of neural networks be applied to such cases. These networks demonstrate the ability not only to address non-linear problems but also demonstrate the ability to adapt to processes through experimental data.

The perceptron is one of the oldest and most basic classes of adaptive networks[4]. It was born from an attempt to model the functions of the brain, hence the neuro-physiological terms. The fundamental unit is called a neuron. Several neurons are gathered into a mathematical layer. Several layers are mathematically stacked together making a network. Each neuron has a number of inputs each of which may have a different contribution to the output. To provide for this

*Work supported in part by the U.S. Department of Energy

difference, each input is allowed to have its own weight. These inputs are added together and the output is passed through a filter that turns on (provides an output of 1) if the sum exceeds some value or turns off (provides an output of 0) if the sum is lower than the critical value. Figure 1 attempts to demonstrate this idea. A neuron of this type is able to

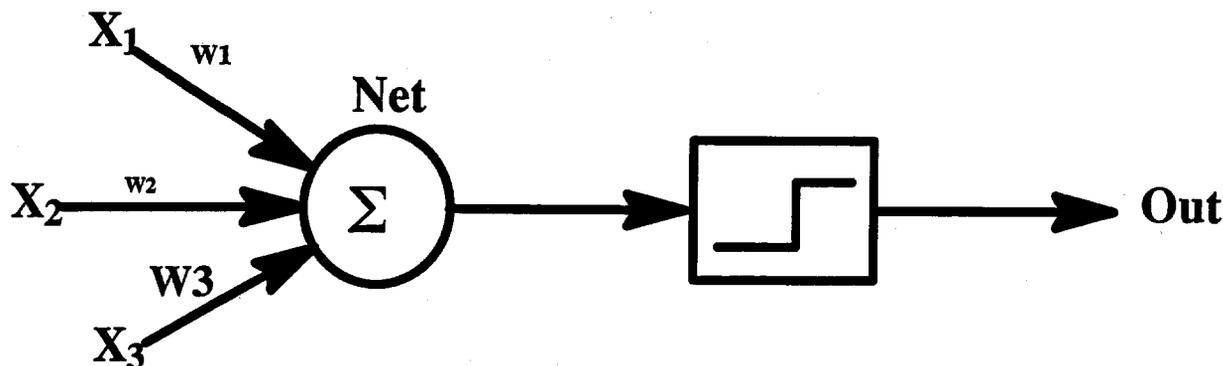


Figure 1. Diagram of a perceptron showing its functional parts.

perform linear discrimination, i.e., can classify inputs as belonging to one particular set or another since it uses a linear combination of inputs to perform the classification. By adding more neurons and allowing them to interact, more than one discriminant can be generated providing for classification into more than one set. Thus, a network of two neurons with two inputs, each inputting to both neurons, and two outputs is able to generate the two input/two output truth tables for the logical functions AND, OR, NAND, and NOR. One of the big disappointments in the early days was the inability of a single layer perceptron such as just described to represent the simple XOR function. However, by adding an additional layer the XOR can be simulated.

The process of weight adjustment is often called "learning" and it is this feature that makes neural networks so useful. As new input/output pairs are generated they are used to adjust the weights. Rosenblatt[4] proposed a training algorithm for single layer perceptrons:

1. Apply the input vector and calculate the output.
2. Adjust the weights.
 - a. If the output is correct, perform no adjustment.
 - b. if the output is zero but should be one, add each input to its corresponding weight.
 - c. If the output is one but should be zero, subtract each input from its corresponding weight.
3. Repeat steps 1. and 2. for all training vectors as many times as necessary.

Symbolically this becomes:

$$\delta = Target - OUT$$

$$\Delta_i = \eta \delta IN_i$$

We've introduced η as a learning rate so the speed of learning can be controlled. The weight adjustment for the $n+1$ st iteration step becomes:

$$w_i^{n+1} = w_i^n + \Delta_i^n$$

Since the hidden layers, if there are any, are not connected to the output, it is impossible to train them this way. We can, however, form an error term in the usual way, i.e., calculate the sum of the squares of the differences of all the targets and the outputs. This error term will be a minimum when the output of the network is a good least squares approximation to the target output. If this error term depends smoothly on the weights, we can simply differentiate the error term with respect to the weights and change the weights so that the error is moved in the direction of the negative of the gradient. When the gradient reaches zero, we will have reached a minimum (albeit it might be a local minimum). There is a problem with the perceptron, however. The step function filter that we used is not a smooth (differentiable) function. We can, however, approximate to it with a smooth function called a sigmoid. Figure 2 shows a sigmoid function. The reduction of the overall error which caused us to change to a sigmoid function was required to provide for training of neurons in the hidden layer(s). This process of backing the error into the hidden layer is known as

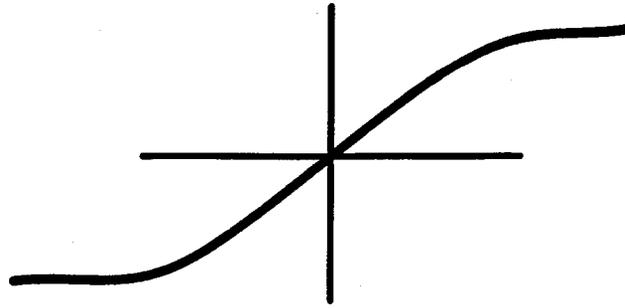


Figure 2. A sigmoid function.

backpropagation. The multilayer perceptron with backpropagation learning is currently the most popular network for real applications. It has the nice property of being able to extract global features from data that would normally be hard to find. It does have some problems, however, which must be either dealt with or lived with[5]:

1. There is nothing to guarantee that the minimum found by the learning algorithm is or is even close to the global minimum.
2. Weights can become so large that the sigmoids become saturated and paralysis of the learning can occur.
3. Training data can be overfit, thus decreasing the ability of the network to generalize.
4. Training is often very slow and requires much data.

Since this paper is not about perceptrons, why, one might ask, have we gone through all of this. There are two reasons. Firstly, from an historical perspective it is interesting and therefore good information to have and secondly, the reasons for and methods of training all types of networks are based on the methodology just described. Since the multilayer perceptron is the most popular network, often when people consider neural networks, the multilayer perceptron is the only network considered. If one purchases a network package for a computer it most likely will be a multilayer perceptron that uses a backpropagation algorithm for training. There are other networks. The body of this paper will consider a different network.

NETWORKS AS FUNCTION APPROXIMATIONS

What is really happening when we apply training data to a set of functions and adjust the weights of the inputs? In reality, we are approximating the relationship of a set of input data to an output function by using a set of basis functions. If we were to generate a perfect fit, we would be able to use a set of coordinates for which there had been no training and would be able to predict what the output would be. In other words, by accomplishing a multidimensional "curve" fit, we will have generated a model of the process from which we had obtained operating data. When fitting a function with a set of basis functions the more properties the basis functions have in common with the function to fit, the fewer bases will be required and the faster the fit will converge. Note that a one-dimensional periodic function is easy to fit with a series of sine functions, a Fourier series. If one thinks about fitting that same periodic function with a set of sigmoid functions, it is intuitively obvious that many sigmoids would be required to supply all the inflection points, positive and negative slopes, maxima and minima, etc., that would be required. This is one of the reasons that a network built of multilayer perceptrons requires many nodes and much training data. We must expose the network to all the vagaries of the function we're trying to fit because the sigmoid doesn't extrapolate with anything that is interesting. The question then becomes, what kind of basis function can we use that extrapolates better and contains more of the properties of the function we're trying to fit.

Using ideas of self organization and recurrency and introducing a cost function that is related to physical properties such as free energy and entropy, one can derive a basis function with some really nice properties. Detailing those arguments and providing the derivation is far too burdensome for this paper. However, one can demonstrate the network and training algorithms by just choosing a basis function of a particular form. That is the approach we will

take. Let us identify the function we're trying to approximate (fit) by $g(\vec{x})$. We begin with the identity:

$$g(\vec{x}) = \frac{\sum_{j=1}^N g(\vec{x}) \varrho_j(\vec{x})}{\sum_{k=1}^N \varrho_k(\vec{x})}$$

Now here is the magic. We define the basis functions as multidimensional Gaussian functions:

$$\varrho_j(\vec{x}) = \beta_j e^{-\beta_j [\vec{x} - \vec{x}_j]^2}$$

where β_j is the width of the Gaussian and \vec{x}_j is the center and \vec{x} is one of the input vectors. This is called a modified radial basis function (RBF). Note that the function spreads in all dimensions from the center. $g(\vec{x})$ can be approximated by a Taylor expansion about \vec{x}_j where only the first two terms are retained:

$$g(\vec{x}) = \phi(\vec{x}) = \sum_{j=1}^N [f_j + (\vec{x} - \vec{x}_j) \cdot \vec{d}_j] \frac{\varrho_j(\vec{x})}{\sum_{k=1}^N \varrho_k(\vec{x})}$$

The f_j is the zero order term in the Taylor expansion and the d_j and its associated coefficient is the gradient term. Defined in this way, this network differs from the usual RBF in two ways:

1. a normalization term and
2. the addition of the gradient term which, by the way, is linear.

We will regard these terms as adaptive weights and since they are linear, the training can be very fast. Further, since they are linear, the cost function (least square) is quadratic in both and consequently has only a single minimum. Because the basis function itself is multi-dimensional and has, in any single dimension, both positive and negative curvature as well as two inflection points and a maximum, it takes on more of the character of the function we're trying to fit. Therefore, it takes fewer basis functions to generate a good fit. Because of these properties, it also extrapolates well which means it requires less training data. We will use these properties to good benefit.

A training method can be derived using gradient ascent. We will add a learning rate. The method is known as the α -LMS method. Using η as the learning rate coefficient, the learning method for both f and d follow:

$$f_j^{p+1} = f_j^p + \eta [g(\vec{x}_p) - \phi(\vec{x}_p)] \frac{\varrho_j(\vec{x}_p) \sum_{k=1}^N \varrho_k(\vec{x}_p)}{\sum_{i=1}^N [\varrho_i^2(\vec{x}_p) + \beta_i (\vec{x}_p - \vec{x}_i)^2 \varrho_i^2(\vec{x}_p)]}$$

$$d_j^{p+1} = d_j^p + \eta [g(\vec{x}_p) - \phi(\vec{x}_p)] \frac{\beta_j (\vec{x}_p - \vec{x}_j) \varrho_j(\vec{x}_p) \sum_{k=1}^N \varrho_k(\vec{x}_p)}{\sum_{i=1}^N [\varrho_i^2(\vec{x}_p) + \beta_i (\vec{x}_p - \vec{x}_i)^2 \varrho_i^2(\vec{x}_p)]}$$

This network has been named by its originators the Connectionist Normalized Local Spline (CNLS) Network. We will refer to it as CNLSnet. To reiterate, the CNLSnet interpolates smooth functions well; it learns fast; local minima are not a problem; the network does not become paralyzed at extreme weight values; few nodes are required to approximate smooth functions. However, CNLSnet will fail if given too much irrelevant or redundant data. This is typically not a problem when it is used as part of a control strategy.

MODEL PREDICTIVE CONTROL

Much of "classical" control theory depends on the closed loop feedback of an error between the target value where we want the plant to be and where the plant actually is. This error is then passed through a transform to provide a correction term to the control. This is a linear process and depends on the linearity of the plant for performance. One can play a few tricks to maintain good linear control even with non-linear plants. The most straightforward way is to operate the plant

in a limited region maintaining approximate linearity. Of course, this won't work in all cases. Another trick often used with PID controllers is to generate a table of different gains that are applied as the process moves from one region to another. This is called gain scheduling. In this way, good and sometimes even excellent control may be obtained. By paying attention to plant design and optimizing the process, efficiencies of ninety-eight percent[1] are quite typical in the petrochemical industry. The remaining two percent inefficiencies are normally due to the difficulty of optimization during transient conditions when process parameters change in an unplanned or nonlinear manner. Good nonlinear optimization might raise this to ninety-nine percent efficiency. Even though this doesn't seem much it could mean a \$10 million dollar per year savings for a billion dollar per year plant.

Model predictive control breaks the feedback and makes control decisions based upon a comparison between where the plant is and where the model of the plant says it should be. With a perfect model one should be able to obtain perfect control. Although model predictive control has been widely used in the chemical and petroleum industries[1], difficulties in producing good first-principle models has limited its effectiveness. The usual procedure is to develop a linear model, based on empirical data and use that model in an optimization routine. Artificial neural networks provide another avenue. Using the same empirical data a neural network model of the plant is built. Because neural network models can "learn" to represent any well-behaved nonlinear function[6], they should be able to represent the plant more accurately, and thus lead to better performance.

Before describing the controller that is the subject of this paper, we need to discuss two uses of a neural network model and the implications of those uses on the training of the network. The first type of use was reported on during ICALEPCS '93[2]. In this application, an attempt was made, prior to any process control, to fully map and model the control surface. Thus, training data was gathered by incrementing the four independent variables over their allowed ranges. This generated 2401 data points. The model was tested against the control surface by finding the optimum operating point in the model and setting the control variables accordingly. Once the process had settled, the real operating point was compared to the model. If the model had not yet converged, that input/output vector was added to the training data and the network was retrained. This process continued until agreement was reached between the model and the process, in this case a negative ion source. At this point, real process control could begin and operation was optimized. This was basically a steady-state process with little, if any, transient problems. Long term, slow drifts seemed to be the only transients. However, the model seemed to change from one operating period to another thus the optimal operating point was unknown from one operating period to the next. This meant that the model had to be rebuilt each time the ion source was started. This type of model building and control was most appropriate for his kind of process.

Another usage is characteristic of processes where *a priori* knowledge indicates where the process should be operated. However, the output parameters may be dramatically changed during operation and their inputs or operating parameters may undergo drastic transients. This is characteristic of an ultra-purity fractional distillation column. The column is normally started with the same operating conditions. Once normal operation is attained the schedule may call for changes in the output to generate different products or the feed to the column can change dramatically. Also, most of these columns are outside, exposed to the weather and although they are normally well insulated, a rain or snowstorm can cause drastic changes in the operating parameters. It is the job of the controller to correct for these changes and maintain desired operation. The total control surface is not of much interest in this scenario. Further, because of the costs involved, nominal operation must be attained as quickly as possible. The operating scheme would be to fit the model to a few localized points around the normal operating point. As the process changes or is changed the controller will adapt to the new conditions by first extrapolating the model and then building a new model with the new data as it becomes available. Since training and adaptation is a continuing process, it can be done with a small amount of data and is, therefore, very fast. If time is included in the training vector, the model will also be able to predict temporal trajectories. Using this data, the process output can be made to follow certain desired trajectories as the operating parameters are changed, e.g., reduce or even eliminate the amount of overshoot as the process approaches some setpoint.

The procedure that is used to minimize the least-square error is a conjugate direction method known as the Fletcher-Reeves method[7]. This method is better than gradient descent because it uses line searches in directions that are orthogonal to the previous search direction. It has additional requirements, however. It requires both a training differential and a training gradient. It turns out that these are easy to come by. Recalling:

$$g(\vec{x}) = \phi(\vec{x}) = \frac{\sum_{j=1}^N [f_j + (\vec{x} - \vec{x}_j) \cdot \vec{d}_j] \frac{e_j(\vec{x})}{\sum_{k=1}^N e_k(\vec{x})}}{\sum_{k=1}^N e_k(\vec{x})}$$

This can be written:

$$\phi(\vec{x}) = \vec{w}^T \cdot \vec{\varrho}(\vec{x})$$

Since this is linear we can form:

$$\vec{\phi}_{i+1}(\vec{x}) - \vec{\phi}_i(\vec{x}) = \Delta \vec{\phi}(\vec{x}) = (\vec{w}_{i+1} - \vec{w}_i)^T \cdot \vec{\varrho}(\vec{x})$$

$$\Delta \phi(\vec{x}) = \Delta \vec{w}^T \cdot \vec{\varrho}(\vec{x})$$

We now have the training differential. It uses the same basis functions and the same training methods. We play the same trick one more time to obtain the gradient.

LOS ALAMOS MPC CONTROLLER

The MPC controller that is the subject of this paper uses CNLSnet to build its model. It uses temporal data to provide trajectories so that as setpoints are changed it will follow a predetermined function as it changes the control parameter with time. It does this by taking one time step and comparing where it thinks it should be with where the trajectory says it should be. The next step corrects any error as well as moving toward the setpoint. This continues until the setpoint is reached. If large enough errors are found during this process, network training takes place. Actually, one additional provision has been built-in. When first starting control, the controller has only three basis functions (actually this number is adjustable but three seems to work well). As it controls the process, if it finds a large enough error between where it thinks it is and where it really is, it has the ability to add additional basis functions up to a maximum, another adjustable controller parameter. The controller also has the ability to move a basis function to a more appropriate place should that be deemed necessary. This provides the controller with the greatest amount of flexibility as well as targeting only the appropriate part of the control surface.

To date, the controller has been applied to two simulations and has been installed in a commercial control system in preparation for implementation on a high-purity fractional distillation column in Ponca City, OK. The two simulations are an exothermic chemical reaction in a Continuously Stirred Tank Reactor (CSTR) and the high purity distillation of tritium in a set of four, ganged, cryogenic fractional distillation columns at the Tritium System Test Assembly at Los Alamos.

The equations that govern the behavior of the CSTR follow.

$$\frac{dX_A}{dt} = -X_A + D_a(1 - X_A)e\left(\frac{T_R}{1 + \frac{T_R}{\gamma}}\right)$$

$$\frac{dT_R}{dt} = -T_R + BD_a(1 - X_A)e\left(\frac{T_R}{1 + \frac{T_R}{\gamma}}\right) + \beta(T_J - T_R)$$

$$\frac{dT_J}{dt} = \delta_1\delta_2(T_{J_0} - T_J) + \beta_J\delta_1(T_R - T_J)$$

All parameters are dimensionless. X_A is the concentration of species A. T_R is the temperature inside the reactor. The temperature of the jacket is T_J and T_{J_0} is the temperature of the heat bath which controls the temperature inside the reactor. The rest of the parameters are constants such as heat transfer coefficients, activation energies, etc. These equations clearly demonstrate the non-linear nature of the process. Figure 3 is a schematic of a CSTR showing the reaction vessel, the jacket and the stirrer. The schematic also shows some of the controllers that are used to control the process.

Figure 4 is the steady state operating curve of the CSTR. Note that except at the points where the gain changes sign, T_R varies approximately linearly with T_{J_0} .

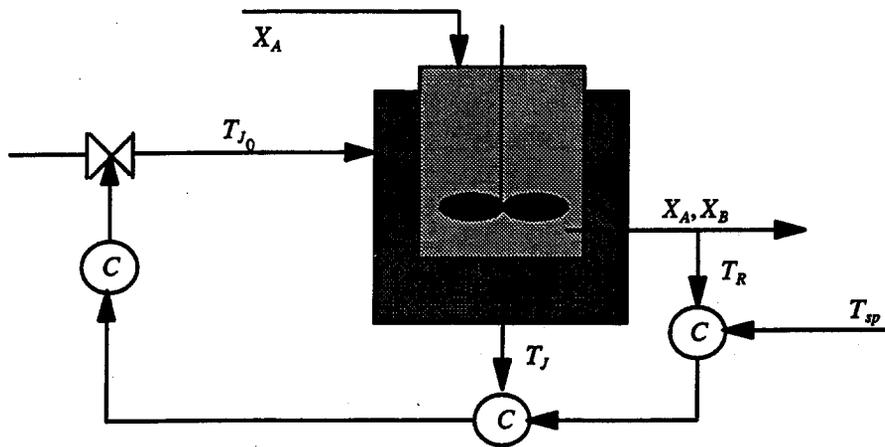


Figure 3. Diagram of CSTR

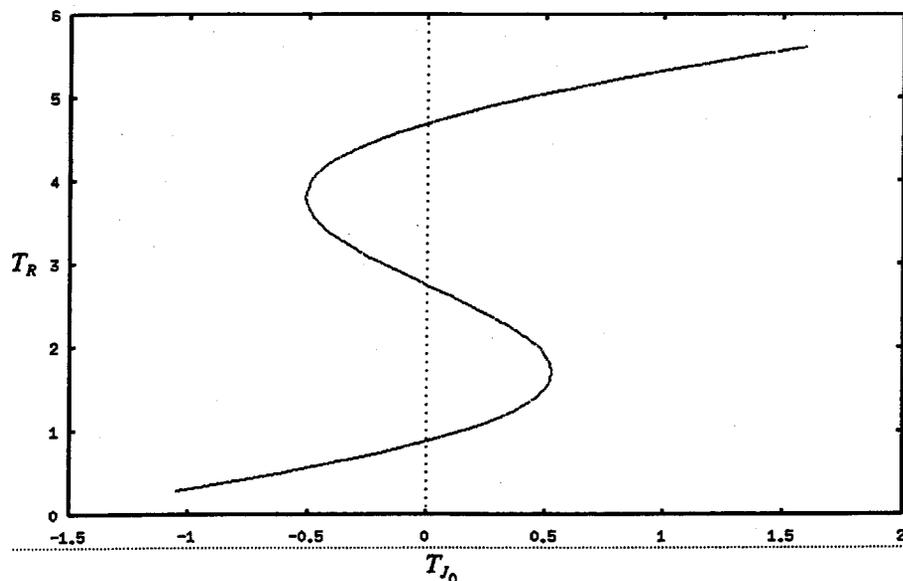


Figure 4. Steady State Operating Curve of CSTR

It turns out that the best operating place is on the part of the curve that exhibits negative gain. Controlling the process in this region is normally very difficult. This makes an interesting trial for the MPC controller. Figure 5 shows the data taken during the training run. Note the limits of the unstable region. When the training session started the control was set for the process to be below the unstable region and it settled into a steady state. The control was raised but not so much as to cross the stability boundary then was lowered. The process followed along as expected. The control was increased again but this time to a point so that the stability boundary was crossed and the process crossed the unstable region and operated on the upper part of the curve. The control was then reduced and the process returned through the unstable region to the lower part of the operating curve. It is important to note the relatively few data points that were generated while the process was in the unstable region. The reason that this is important is because that is where we are going to attempt to run the process. Figure 6 demonstrates just such a run. Note that whether there is a change in the setpoint or change in the plant (feed disturbance), the MPC is able to hold the process on setpoint in the unstable region as well as the stable region. When the MPC was applied to the simulation of a high-purity fractional distillation it again performed well, maintaining control in both linear and non-linear regimes.

CONCLUSIONS

The capabilities of model predictive control where the model is built by a neural network and is continuously adapted to the process has been clearly demonstrated. The controller has been taken from a theoretical proof-of-principle to a

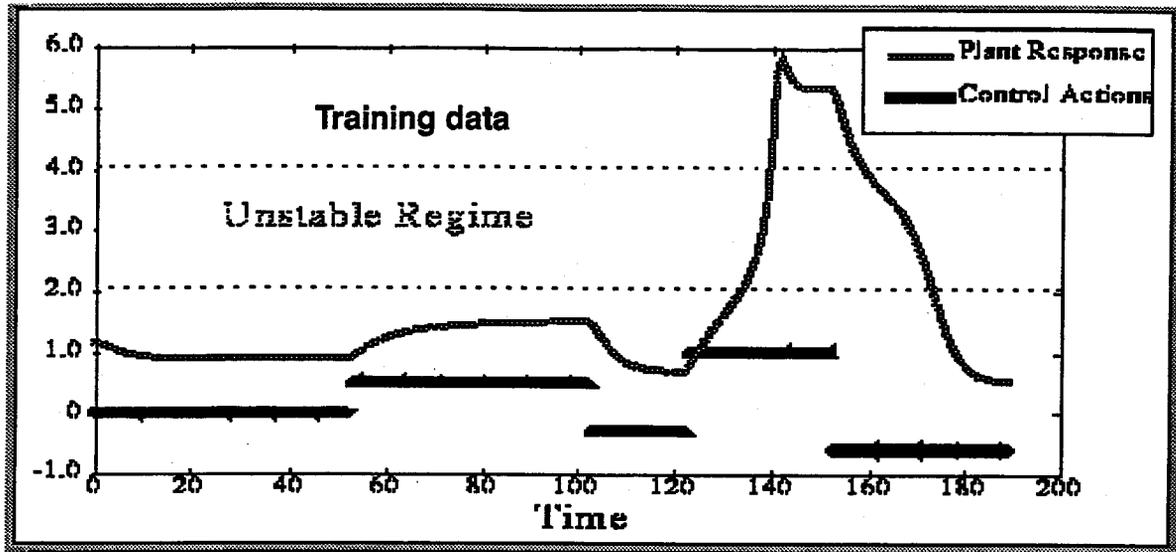


Figure 5. Data taken during training

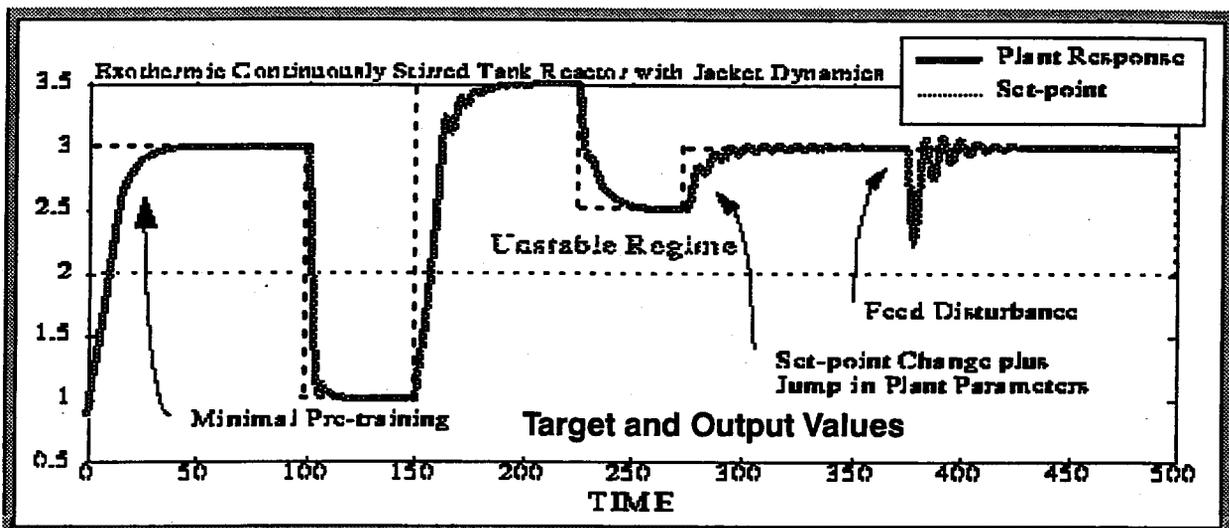


Figure 6. Control run showing MPC maintaining control in unstable region

controller ready for commercial use. This controller can be used in many different control strategies. We have shown its capability in two. In some sense, it might be more appropriate to take the place of PID control in linear processes because one would not have to worry about finding appropriate gains for the PID control. The MPC will adapt to the process. The trade-off is a bit more complexity with which to deal. We are in the process of attempting to implement this technology to provide adaptability in a state space formalism. The impetus for this work comes from the beam steering feedback done several years ago at SLAC and reported during ICALEPCS '91[8]. We will continue to improve on this technology as new and interesting control problems present themselves.

REFERENCES

- [1] P.J. Werbos, T. McAvoy, and T. Su, *Handbook of Intelligent Control* (Van Nostrand Reinhold, New York, NY, 1992) 286.
- [2] W.C. Mead, S.K. Brown, R.D. Jones, P.S. Bowling, C.W. Barnes, *Adaptive Optimization and Control Using Neural Networks*, Proceedings of the 1993 ICALEPCS Conference, Berlin, Germany, 1993, 309-315.
- [3] C.C. Baum, K.L. Buescher, R.D. Jones, W.J. Parkinson, and M.J. Schmitt, *Two-Timescale, Model Predictive Control of Two Simulated Chemical Plants: Lagged-CSTR and Distillation Column*, Los Alamos Report

(LA-UR-94-1039).

- [4] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, **65**, (1958).
- [5] J. Galbraith, R.B. Webster, C.D. Bergman, R.D. Jones, **Introduction to Adaptive Computation**, Unpublished book.
- [6] E. Blum, Approximation theory and feedforward networks, *Neural Networks*, **4**, 1991.
- [7] D.G. Luenberger, **Linear and Nonlinear Programming**. Reading, MA, Addison Wesley, 1984.
- [8] L. Hendrickson, S. Allison, T. Gromme, T. Himel, K. Krauter, F. Rouse, R. Sass and H. Shoae, *Generalized Fast Feedback System in the SLC*, Proceedings of the 1991 ICALEPCS Conference, Tsukuba, Japan, 1991.

Digital Feedback System for Orbit Stabilization at the SIBERIA-2 Light Source

A.Batrakov, S.Kuznetsov*, E.Levichev, V.Sajaev, V.Shilo
Budker Institute of Nuclear Physics, Novosibirsk 630090, Russia
*Kurchatov Institute, Moscow 123182, Russia

Abstract

An implementation of global/local feedback system on the 2.5 GeV SIBERIA-2 storage ring is reported. This project is based on parallel data handling in BPM modules and VME crate for communication and magnet correction control. We present special own-made beam position monitor data acquisition module, which includes ADC, DSP TMS320C31 and high-speed serial link. Paper describes software for simulation of feedback control and digital filtration. The motivation, approaches, results and future plans of this project are discussed.

1 INTRODUCTION

In the design of a high-brightness light source, the electron beam emittance is a parameter of prime importance. Any vibrations that lead to distortions in the closed orbit will result in a larger effective emittance. Together with the brightness reduction, unwanted beam motion that causes the incident light position and angle to vary can degrade the experimental advantages of synchrotron radiation.

SIBERIA-2 is a dedicated light source with an electron energy of 2.5 GeV that was developed by the Budker Institute of Nuclear Physics (Novosibirsk) for the Kurchatov Institute (Moscow) and commissioned at the beginning of 1995 in Moscow [1]. The typical beam stability tolerance for modern light sources is about 10% of the beam size and divergence. For SIBERIA-2 this criterion means that we have to position the beam orbit accurately to the 10-15 μm level. An active feedback system, which detects and counteracts undesired beam fluctuations by correcting the closed orbit, is essential to achieve the necessary beam stability [2]. Potential sources driving the variation in electron beam are: ground vibrations, magnet power supply ripple and thermal drift. A specific feature of the low-emittance light source magnetic lattice is the significant amplification factor for any mechanical vibrations of the magnetic elements. For example, for SIBERIA-2 due to the strong focusing, quadrupole lens motion will transfer to beam motion with the amplification factor (rms) $M_{x,z} \simeq 30$. To stabilize the orbit oscillations, a fast digital feedback system is proposed. A single module composed of a DSP (digital signal processor) board together with 4-channel 10-bit ADCs and 12-bit DACs has been developed. The module was linked to a standard pick-up station and the beam measurements which are necessary to design the feedback system (such as the low-frequency beam noise spectrum, system transfer function, etc.) were performed. A reference model that includes the real measurement results was prepared to test the beam stabilizing system. Using this model, a classical PID algorithm was implemented and a significant reduction of modelling beam motion was achieved in a frequency bandwidth up to 60 Hz.

2 LOCAL FEEDBACK SYSTEM

The local feedback system is intended to stabilize the beam at a single orbit point (where the radiation is emitted) without disturbance of the remainder of the beam trajectory. We use digital signal processing to avoid the problems connected with analog circuits, such as drift, offset, etc. and to increase the flexibility of the system. The local feedback system consists of a fast digital controller, an existing pick-up station, three steering magnets and power supplies. A block diagram of the feedback system is shown in Fig.1. The controller is a DSP system which is composed of a 32-bit floating point TMS320C31 board, a 4-channel ADC board with 10-bit resolution and a 3-channel DAC board with 12-bit resolution. The sampling rate of the system is 2.5 kHz. The sampling period may be divided into three parts. During the first part the ADC provides the measurements of 600 turns of the beam (414 nsec per turn) and stores the data into the memory. Then the TMS computes the average beam centroid

coordinates for the vertical and horizontal planes. This procedure provides us with a significant electronics noise reduction and enhanced accuracy in comparison with a single ADC reading. During the third part the output signal is produced using the PID controller and the system transfer function, and the steering magnets make a closed orbit bump to cancel the beam position displacement detected by the monitor. A digital bandlimiting filter (BLF) is inserted into the loop before PID controller for stability reasons.

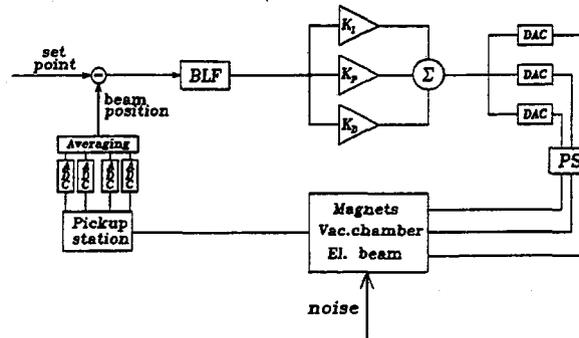


Figure 1: Block diagram of the closed loop.

The transfer function of the feedback system is composed of two components: $G(z)$ which corresponds to the controller and $H(z)$ which describes in the z-domain the transfer function of the rest of the system. In the case of SIBERIA-2 the last term is mainly determined by eddy currents in the aluminum vacuum chamber. To determine this transfer function we have measured the attenuation of the corrector field by the vacuum chamber with a low electron current. The electron beam was excited by a sinusoidal current in the frequency range of 0 to 150 Hz; the beam displacement was measured at the monitor and an FFT of this signal was performed. Fig.2 shows the transfer function of the system with the controller excluded. One can see that at a frequency of 100 Hz the attenuation is about -20 dB.

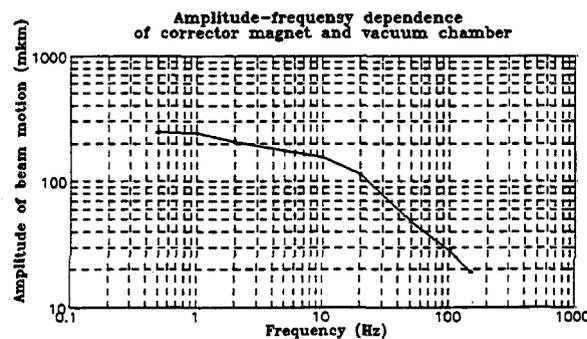


Figure 2: Amplitude- frequency dependence of a system consisting of corrector magnet and vacuum chamber of the SIBERIA-2.

The digital filter function of the PID controller $G(z)$ is decided by the control program. We use a classical PID controller algorithm with anti-windup reset and limitation of derivative gain [3]. In the z-domain $G(z)$ can be written as:

$$G(z) = K_p + \frac{K_i}{1 - z^{-1}} + K_d(1 - z^{-1}), \quad (1)$$

where K_p , K_i , and K_d are the proportional, integral, and derivative constants of the controller. A control program based on the PID controller algorithm was coded in C and downloaded to the DSP. The feedback system was

tested with the model that takes into account the results of response measurements that was done at the SIBERIA-2 (Fig.2). To describe the vacuum chamber influence, a digital filter $H(z)$ with four poles and four zeros was developed and implemented in the simulation code. The noise attenuation obtained from the simulation is plotted in Fig.3. It can be seen that at a frequency of 10 Hz the attenuation is around -30 dB, while for 60 Hz this value is decreased to -10 dB. For this test, the controller coefficients were: $K_p = 1.0$, $K_i = 0.003$, and $K_d = 8.0$.

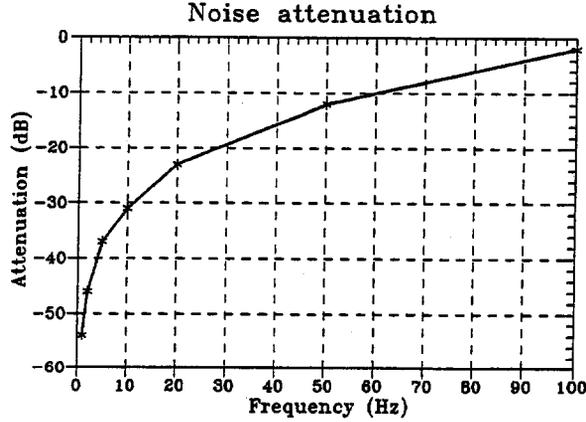


Figure 3: Noise attenuation of the closed loop with $K_p = 1$, $K_i = 0.003$, $K_d = 8$ and $T_s = 0.0004$.

3 GLOBAL FEEDBACK SYSTEM

A global orbit correction system includes 24 BPMs and 48 correctors in each plane. The global orbit correction may be considered as the extension of the local feedback correction after using the technique of singular value decomposition (SVD) [4] of the response matrix R_{ij} that connects the beam motion at the i -th pick-up station with the changing of the j -th steering magnet strength:

$$R_{ij} = \frac{\sqrt{\beta_i \beta_j}}{2 \sin \pi \nu} \cos(|\psi_i - \psi_j| - \pi \nu), \quad (2)$$

where (β_i, ψ_i) are the beta and phase functions for i -th beam position monitor, and similarly for j -th corrector (β_j, ψ_j) , and ν is the betatron tune. The response matrix is usually obtained by reading beam position displacement while varying the corrector strengths one by one. SVD reconfigures the BPMs and correctors into the same number of "pseudo" BPMs and "pseudo" correctors. In this new configuration space each BPM is coupled with a single corrector and vice versa. For each "BPM-corrector" pair the solution to the problem can be found by the method described above for the local feedback system. Then the backward transformation lets us know the strengths of the real correctors. As in the SVD technique, only matrix transformations take place so it looks very attractive to implement this method in a DSP.

At the local level, each BPM station electronics includes a 4-channel ADC and a DSP TMS320C31 which is connected via a serial data link with a Central Processor Board (CPB), located in a VME crate. The CPB, where we expect to use a TMS320C40, treats all the information and produces the output signal through a DAC to the corrector power supplies. The work of the CPB module includes the communication of the C40 with every BPM station. The performance of the C40 allows us to operate with the expected dataflow. We propose using a MC68030 or MC68040 VME master processor under the OS-9 operating system to provide the supervisory, I/O and file management functions. The system software for OS-9 has been developed.

4 DIGITAL SIGNAL PROCESSING

The intelligent BPM controller is based on the TMS320C31-40 Texas Instruments floating point DSP. The structure of the controller is shown in fig.4. The DSP has 32 KWords of local static memory and 32 KWords of dual ported memory (DPM). Both memories have an access time of 50ns. The DPM is shared by the processor and ADCs. The

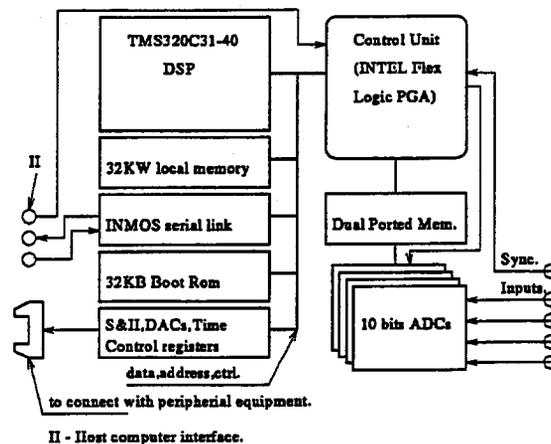


Figure 4: Structure of the DSP controller.

use of a DPM allows the ADC to write the data directly to the processor address space. This allows the use of the ADC data without an additional time penalty for data transfer. Peripheral registers to control an external board with a programmable delay generator, DACs and sample-and-hold registers are mapped into the DSP address space. A 64 kbyte ROM contains the primary bootloader and some application codes. These codes can be loaded into local memory during program execution. All on-board devices are managed by a dedicated control unit, implemented inside the INTEL Flex Logic PGA. The hardware interrupts are used by the DSP to synchronize the application software with external events. Interrupts when the ADCs have data for writing to the DPM and when the DPM is full are possible.

Ten-bit ADCs with common simultaneous synchronization and a maximum sample rate of 15 million samples per second are installed on the small piggy-back module at the moment. The six least significant bits in the DSP are reserved to increase ADC's resolution in future without software modification. The control unit drives the DPM address bus during the write process of ADC data to DPM.

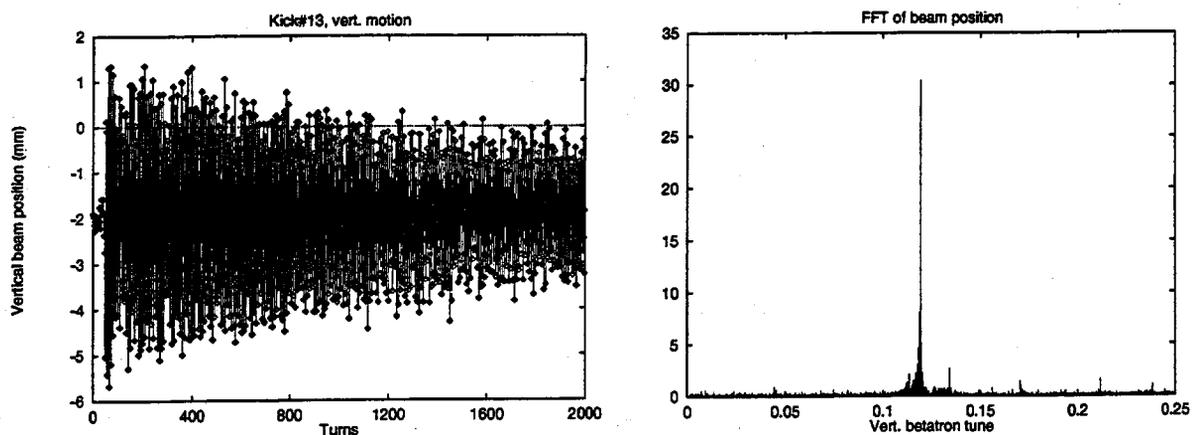


Figure 5: Beam position as a function of revolution number and its Fourier transform after vertical kick.

The INMOS serial link interface IMSC012 is used to provide I/O functions with the host computer and code downloading with a physical data rate of 10 or 20 Mbit/s. All signals can be transmitted/received up to a distance of 200 meters via 50 ohm coaxial cable using a home-made interface. The link speed for this distance is 10 Mbit/sec. Choice of the INMOS serial data interfaces allows the use of the transputer host software to load DSPs and to implement host I/O functions without development of additional host hardware or software.

In addition to the feedback system development, the DSP board provides the possibility of using various beam diagnostic techniques. Fig.5 displays the sampled signal as a function of revolution number and its discrete Fourier transform after the beam was excited vertically by fast kicker. The subsequent decay is clearly seen. This mode can be used for experimental nonlinear phase space study and dynamic aperture investigation. We expect that turn-by-turn measurements of beam position and current can be useful for the investigation of fast processes: beam loss during injection, ion accumulation, some kinds of coherent instabilities, etc. Fig.6 demonstrates the injection into the VEPP-3 storage ring. A "slow wave" corresponds to the energy oscillations. Due to the high computation capability of the DSP (2.5 ms for a 1024 point FFT) we can perform the various algorithms for signal treatment and investigate dynamic processes.

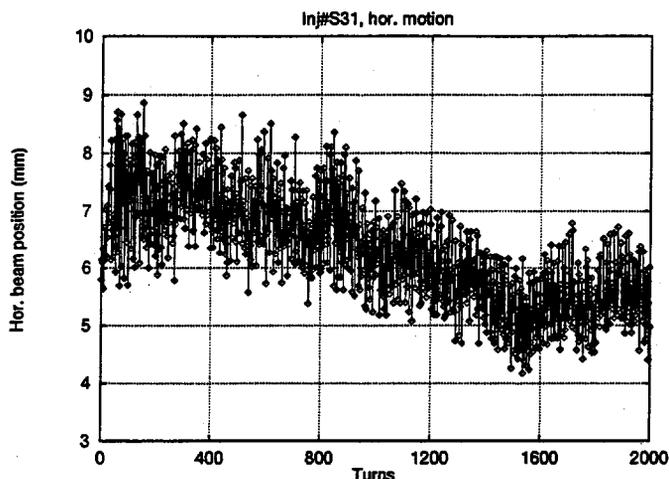


Figure 6: Injected beam: horizontal position vs. turn number.

5 Conclusion and acknowledgments

The development of the orbit correction feedback system for the SIBERIA-2 light source is in progress. The home-made DSP module which combines a TMS-processor and ADC and DAC boards was tested with the electron beam using the existing pick-up station. The necessary measurements of the system response, including the influence of the vacuum chamber, corrector magnet and power supply were carried out. A model of the full system was developed and a PID-controller was implemented in that DSP. The results of the simulation indicate an adequate suppression of the beam motion. We expect that the steering magnet and power supply unit with an improved bandwidth (up to 250 Hz) will be manufactured by the end of October and the total feedback system (in a local mode) will be tested with beam by the end of 1995.

The authors would like to thank Dr. A.Kalinin and Dr. V.Smaliuk who kindly help them to carry out the orbit measurements and also to Dr. S.Kovalev for developing the bootloader for the DSP.

References

- [1] V.N. Korchuganov et al., NIM 208 (1983), pp.11-18.
- [2] J.N.Galayda, Y.Chung, and R.O.Hettel, in "Synchrotron Radiation Sources - A Premier", ed.H.Winick, World Scientific, 1994, pp.344-376.
- [3] K.J. Astrom and B. Wittenmark (1990): Computer Controlled Systems -Theory and Design, second edition, Prentice-Hall, Englewood Cliffs, NJ.
- [4] W.Press et al., Numerical Recipes in C, Cambridge University Press, p.60, 1989.

Digital Closed Orbit Feedback System for the Advanced Photon Source Storage Ring*

Y. Chung, D. Barr, G. Decker, J. Galayda, F. Lenkszus, A. Lumpkin and A. J. Votaw
Argonne National Laboratory, Argonne, IL 60439, U.S.A.

ABSTRACT

The Advanced Photon Source (APS) is a dedicated third-generation synchrotron light source with a nominal energy of 7 GeV and a circumference of 1104 m. The closed-orbit feedback system for the APS storage ring employs unified global and local feedback systems for stabilization of particle and photon beams based on digital signal processing (DSP). Hardware and software aspects of the system will be described in this paper. In particular, we will discuss global and local orbit feedback algorithms, PID (proportional, integral, and derivative) control algorithm, application of digital signal processing to compensate for vacuum chamber eddy current effects, resolution of the interaction between global and local systems through decoupling, self-correction of the local bump closure error, user interface through the APS control system and system performance in the frequency and time domains. The system hardware including the DSPs is distributed in 20 VME crates around the ring and the entire feedback system runs synchronously at a 4-kHz sampling frequency in order to achieve a correction bandwidth exceeding 100 Hz. The required data sharing between the global and local feedback systems is facilitated by the use of fiber-optically-networked reflective memories.

I. INTRODUCTION

The Advanced Photon Source (APS) is one of the third-generation synchrotron light sources which are characterized by low emittance of the charged particle beams and high brightness of the photon beams radiated from insertion devices (IDs). In order to take full advantage of the intense synchrotron radiation, the incident intensity, position, and angle of the x-ray beam need to be tightly controlled [1-3]. Even though every effort is made to stabilize the electrical and mechanical components of the ring, there will inevitably be residual beam motion primarily caused by the quadrupole vibration.

The sources of vibration include ground vibration, mechanical vibration of the accelerator subcomponents, thermal effects, and so forth. These are manifested in the undesired particle and x-ray beam motion. This results in increased beam size and diluted beam emittance in the short term. An example of the long-term effect is the diurnal changes in the ring circumference and periodic shift of the x-ray beam at the user station [4]. At the APS, the stringent transverse x-ray beam position stability required by the current user community will be achieved through extensive beam-position feedback systems with the correction bandwidth exceeding 100 Hz [5].

The APS has 360 rf beam position monitors (BPMs) and 318 corrector magnets distributed around the storage ring, miniature BPMs for ID beamlines, and x-ray BPMs in the front end of x-ray beamlines for global and local orbit feedback. The real-time (AC) feedback systems, which are the main focus of this work, will use a subset of these.

The feedback systems can be largely divided into the global and local feedback systems according to the scope of correction. The global feedback system uses up to 80 rf BPMs and 38 corrector magnets distributed in 40 sectors. The primary function is to stabilize the selected perturbation modes of the global orbit. The local feedback systems, on the other hand, stabilize the source regions of the x-ray beams locally for angle and displacement.

An ideal local feedback system would not affect the rest of the closed orbit or other local feedback systems. In reality, the global and local feedback systems constantly interact with one another. The effect of global orbit feedback unavoidably interferes with the local feedback. On the other hand, the bump closure error in the local feedback due to bump coefficient error, magnet field error, eddy current effect, etc., causes global orbit perturbation and affects other local feedback systems. If this interaction is too strong, the feedback systems can become ineffective, oscillatory, or even unstable. In order to minimize such effects and maximize the feedback efficiency, it is necessary to decouple the global and local feedback systems and to compensate for the local bump closure error. The required data sharing between the global and local feedback systems is facilitated by the use of fiber-optically-networked reflective memories.

In this work, we will discuss the feedback control algorithms and hardware and software configuration for the APS orbit feedback systems. The remainder of this paper will be an overview of the feedback algorithm and system description in Section II, system performance results in Section III and the current status of system integration in Section IV.

II. FEEDBACK ALGORITHM AND SYSTEM DESCRIPTION

*Work supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-8.

The APS orbit feedback system has the capability of handling multiple local feedback systems for control of the insertion device (ID) and bending magnet (BM) x-ray beamlines and a global feedback system to minimize the global orbit distortion and thus to maintain high beam quality. In this section, we will describe the algorithms governing the operation of these feedback systems [5].

A. Control Algorithm

The signal processing core of the feedback systems is based on digital signal processing (DSP) [6]. The design sampling frequency F_s ($= 1/T$) or the number of feedback loops executed per second is 4 kHz. Figure 1 shows the schematic of a multichannel digital feedback system. The object to be controlled, or the plant, is represented by the matrix R . R_{inv} is the inverse-model matrix that controls coupling of various control points in the plant. In the case of orbit feedback, R is a composition of the global and local response matrices.

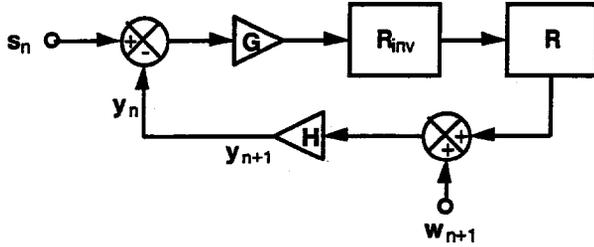


Fig. 1: The schematic diagram of a digital feedback system.

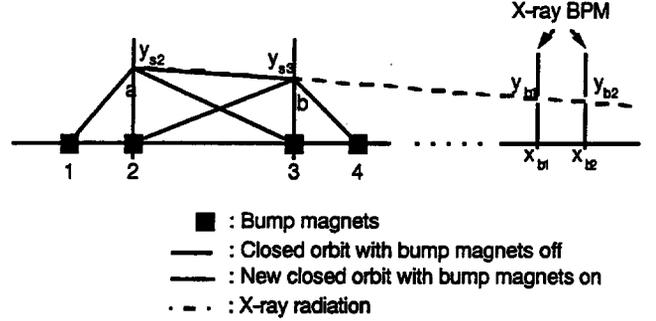


Fig. 2: Schematic of the local feedback system.

The gain matrix G is comprised of the low pass filter (LPF); the proportional, integral, and derivative (PID) controller; and any compensation filter (CF) that cancels the undesirable frequency dependence in the plant, such as the effect of the eddy current in the vacuum chamber. H represents the BPMs.

The difference equation describing the response of the feedback system in Fig. 1 is given by

$$y_{n+1} = H \cdot \{R \cdot R_{inv} \cdot G \cdot (s_n - y_n) + w_{n+1}\} \quad (1)$$

Applying the Z-transform to Eq. (1), we obtain

$$Y(z) = \left\{ 1 - F(z) \frac{1}{H(z)} \right\} \cdot S(z) + F(z) \cdot W(z) \quad (2)$$

where

$$F(z) = \frac{1}{1 + H(z) \cdot R \cdot R_{inv} \cdot G(z)z^{-1}} \cdot H(z) \quad (3)$$

$Y(z)$ is the Z-transform of $\{y_n\}$, $W(z)$ is the Z-transform of $\{w_n\}$, and so forth. The expression $1/(..)$ denotes the inverse matrix. The matrix $F(z)$ is the noise-filter matrix and with the substitution $z = \exp(-i\omega T)$, we can obtain the frequency response of the feedback system. The last term in Eq. (2) represents the residue of the perturbation in the orbit with feedback.

The response matrix R and the inverse matrix R_{inv} are shown in Fig. 3. The matrix R is largely composed of four component matrices: R_g , $R_{g,b}$, R_{lg} , and R_l ; and the inverse matrix R_{inv} is largely composed of three component matrices: R_{ginv} , $R_{inv,lg}$ and R_{linv} . These matrices are explained in the following subsections.

B. Local Orbit Feedback

A local orbit feedback system is based on a four-magnet local bump and will be the primary feedback mechanism to stabilize the local x-ray beamline for angle and displacement. The beam positions are detected by two rf BPMs inside the local bump and two x-ray BPMs as shown in Fig. 2 [2].

Four-magnet local bumps can be decomposed into two independent three-magnet local bumps, a and b , and transformation between the bump strengths and the beam positions is straightforward. The locality of the bump can be achieved by powering the bump magnets in certain ratios determined by local bump coefficients. Empirical derivation of these coefficients is discussed in [7]. The local response matrix R_l can then be reduced to a 2×2 matrix relating two beam

positions and two bump strengths. The local inverse matrix R_{iinv} is the inverse of the matrix R_i . In case there are multiple local feedback system, R_l and R_{linv} are aggregates of matrices as shown in Fig. 3.

Imbalance in the bump coefficients can cause local bump closure error and introduce global orbit distortion. This error is represented by the matrix R_{gl} , which is assumed to be zero in the model. Even though the coefficients are well matched at DC, eddy current effect in the relatively thick (1/2") aluminum vacuum chamber will cause significant bump closure error if orbit perturbation contains components with a high enough frequency. This is remedied by global feedback and self-correction of local bump closure error.

C. Global Orbit Feedback

The global feedback system attenuates global orbit distortion induced by bump closure error in the local feedback systems as well as other vibration sources in the ring. For beam motion detection, up to 80 BPMs distributed in the ring, or two BPMs per sector, are used. For orbit correction, the local feedback correctors are also used in addition to the 38 global correctors per plane, with the exception of sector 39 and 40 where the correctors were not installed due to space constraints near the injection point. The vacuum chamber at the location of the global corrector is made of thin stainless steel unlike the local feedback systems. Both kinds of correctors are treated equally in the algorithm and both contribute to the global orbit correction. However, by including the local feedback correctors in the global feedback system, the local bump closure error will be corrected most effectively. This has the same effect of having two sets of correctors at the same location, one set for local feedback and the other for global feedback.

Under this scheme, the dimension of the global response matrix R_g is $M_g \times (N_g + 4L)$, where M_g is the number of global BPMs, N_g is the number of global correctors and L is the number of local feedback systems. The global inverse matrix R_{ginv} is obtained by applying the technique of singular value decomposition (SVD) [3,5,8] to R_g . The local feedback corrector vector Δq_l is then the sum of contribution to the global orbit correction Δq_{lg} , and closed local bump Δq_{lc} , i.e.,

$$\Delta q_l = \Delta q_{lg} + \Delta q_{lc}. \quad (4)$$

Δq_{lg} is obtained by multiplying a corresponding submatrix of the global inverse matrix R_{ginv} and the global orbit error vector. Δq_{lc} is computed based on the inverse of the model local response matrix and contains the decoupling term for proper compensation for the effect of global feedback as explained in the following subsection.

The DSPs and interface boards are installed in twenty VME crates distributed around the ring. Each crate covers two sectors for both the horizontal and vertical planes, an odd sector in the upstream and an even sector in the downstream. A single DSP is assigned per four degrees of freedom. Since each DSP needs access to the global orbit data at 80 global BPMs, data collected from all 40 sectors needs to be available across the local VME bus. This is done via a dedicated fiber optic network of reflective memories with a data transfer rate of 26 Mbytes/sec. For each degree of freedom, the 80 BPM data are multiplied by a row of R_{ginv} . Operation of the DSPs is synchronized by an event signal broadcasted from a central 4-kHz clock. Global and local feedback systems run simultaneously with the same clock frequency.

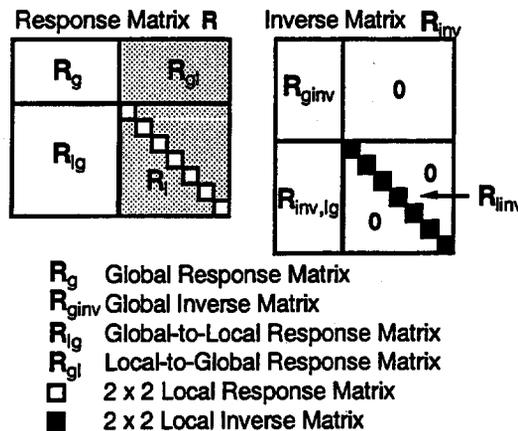


Fig. 3: Response matrix R and its inverse R_{inv} for the unified global-local orbit feedback system.

D. Decoupling of Global and Local Feedback Systems

In the ideal situation of zero local bump closure error, the local feedback systems are transparent to the rest of the ring. However, the global feedback unavoidably interferes with the operation of the local feedback systems unless the source is localized and does not cause global orbit distortion. This global-to-local coupling is represented by the matrix R_{lg} , which

relates global corrector kick and the beam motion on the local beamline. In general, orbit perturbation is seen by both the global and local feedback systems, and they will attempt to correct it independently, which leads to undesirable interaction between the global and local feedback systems. If this interaction is too strong, the feedback systems can become ineffective, oscillatory, or even unstable. This can be resolved by decoupling the global and local feedback systems [5]. This unified approach in effect combines the global and local feedback systems, renders the entire system into multiple non-interacting feedback systems and thus minimizes the interference and the corrector load.

In the case of independent operation of global and local systems, the submatrix $\mathbf{R}_{inv,lg}$ is set to 0, and the feedback systems are coupled. In order to decouple the feedback systems, the effect of global correctors on the local orbit based on the global orbit error is subtracted from local orbit error, i.e.,

$$\mathbf{R}_{inv,lg} = \begin{cases} 0 & \text{coupled} \\ -\mathbf{R}_{inv} \cdot \mathbf{R}_{lg} \cdot \mathbf{R}_{ginv} & \text{decoupled.} \end{cases} \quad (5)$$

This scheme requires that the local feedback systems have access to the global orbit data, which is readily met since the global orbit data is available in all VME crates through the reflective memories.

Using Eq. (5) for the decoupled case and assuming $\mathbf{R} \cdot \mathbf{R}_{inv} = \mathbf{1}$, it can be shown that the matrix product $\mathbf{R} \cdot \mathbf{R}_{inv}$ is given by

$$\mathbf{R} \cdot \mathbf{R}_{inv} = \mathbf{R}_g \cdot \mathbf{R}_{ginv} \oplus \mathbf{1}_1 \quad (6)$$

in the ideal case of no local bump closure error. The operator $\hat{\mathbf{A}}$ combines two matrices as depicted in Fig. 4.

$$\boxed{\mathbf{X}} \oplus \boxed{\mathbf{Y}} = \begin{array}{|c|c|} \hline \mathbf{X} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Y} \\ \hline \end{array}$$

Fig. 4: The matrix combination operator $\hat{\mathbf{A}}$.

Now, putting

$$\mathbf{G}(z) = \mathbf{G}_g(z)\mathbf{1}_g \oplus \mathbf{G}_l(z)\mathbf{1}_l \quad \text{and} \quad \mathbf{H}(z) = \mathbf{H}_g(z)\mathbf{1}_g \oplus \mathbf{H}_l(z)\mathbf{1}_l, \quad (7)$$

we obtain

$$\mathbf{F}(z) = \mathbf{U} \cdot \mathbf{F}_g(z) \cdot \mathbf{U}^T \oplus \mathbf{F}_l(z), \quad (8)$$

where \mathbf{U} is the unitary global BPM transform matrix derived from SVD, and $\mathbf{F}_g(z)$ and $\mathbf{F}_l(z)$ are diagonal. Equation (8) indicates that there exists a coordinate transformation that decouples the feedback channels, and single-channel feedback theory can be applied to each channel.

Using the relation $\mathbf{U} \cdot \mathbf{U}^T = \mathbf{U}^T \cdot \mathbf{U} = \mathbf{1}$, we obtain from Eq. (3) the diagonal elements of $\mathbf{F}_g(z)$ as

$$F_{g,ii}(z) = \begin{cases} \frac{H_g(z)}{1 + H_g(z)G_g(z)z^{-1}} & \text{coupled modes} \\ 0 & \text{decoupled modes} \end{cases} \quad (9)$$

and similarly for $\mathbf{F}_l(z)$. The noise filter matrix for the BPMs can be obtained from Eqs. (8) and (9). The expression for the coupled modes is identical to that of a single-channel feedback system [3]. The PID controller function $G(z)$ is given by

$$G(z) = K_p + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1}), \quad (10)$$

where K_p , K_I , and K_D are the proportional, integral, and derivative controller gains, respectively. These gain coefficients should be positive for negative feedback. When K_I is finite, the open loop DC gain is infinite, and therefore, the long-term drift can be completely corrected.

E. Hardware/Software Configuration and User Interface

Figure 5 shows the operations interface to an orbit feedback system crate. There are a total of 20 such crates, each of which has one VME CPU and multiple DSP boards. Considering the number of processors in the system, it would be impractical and error-prone to write, compile, and link individual codes for each processor. Instead, the codes are generated automatically from base source codes which contain instructions for code generation for different processors. The command parser is based on the GUS kernel, a script-based interpretive shell developed for streamlined data acquisition and analysis [9]. The compiled codes for the VME CPUs, DSPs, EPICS (Experimental Physics and Industrial Control System) data base, and controls interface are downloaded to the target processors through the Ethernet connection. Once the codes begin execution in the processors, the feedback operation, including data sampling and analysis, can be controlled via EPICS and the user interface.

The rf BPMs, x-ray BPMs, and corrector power supplies are interfaced through the feedback system interface controller (FSIC) and the sister board FSIC-B in the VME backplane. Reflective memories are used for global orbit data sharing and are networked through fiber optic cables. During feedback operation, the DSPs perform direct data I/O with FSIC and reflective memory through the local VME bus.

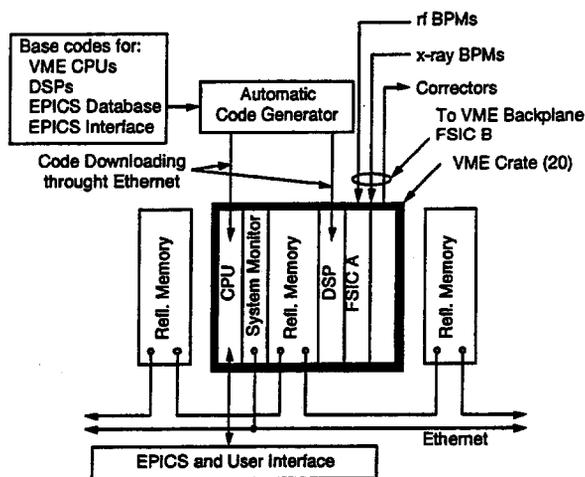


Fig. 5: Operations interface to the orbit feedback system.

III. SYSTEM PERFORMANCE

The performance of the global feedback system is shown in Fig. 6 in the time and frequency domains using the sampling frequency of 4 kHz, LPF bandwidth of 50Hz, and the PID gains $K_P = 10$, $K_I = 0.5$, and $K_D = 2$. For both tests, the global orbit perturbation was simulated using a single steering error in the ring. In the time domain, the steering error introduced a step impulse at $t = 10$ ms. Comparison of the orbit motion with feedback on and off is shown Fig. 6(a). The response time is approximately 500 μ s. The system performance in the frequency domain is shown in Fig. 6(b). The system bandwidth at -6dB noise rejection is 250 Hz. Similar performance results have been obtained from the local feedback system test on the ESRF storage ring [7].

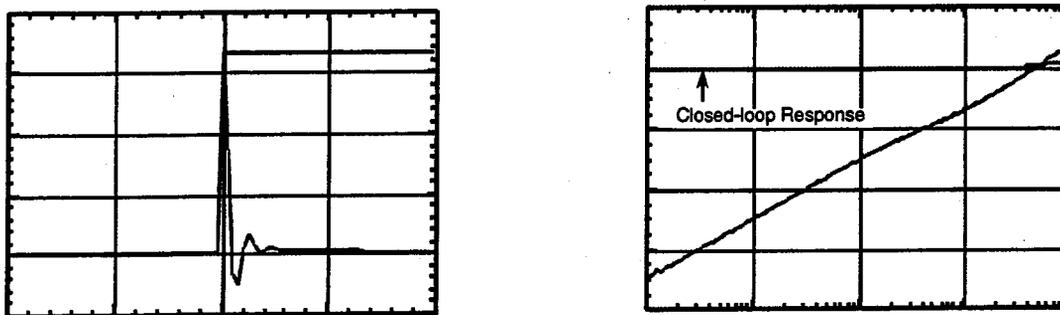


Fig. 6: (a) Impulse response of the global feedback system. (b) Frequency response. The sampling frequency was 4 kHz, LPF bandwidth was 50Hz, and the PID gains were: $K_P = 10$, $K_I = 0.5$, and $K_D = 2$.

IV. CURRENT STATUS

At this time, the backbone hardware, including the VME crates, VME CPUs, DSPs, reflective memories, and communication network, has been installed and tested. An automatic code generator has been developed to facilitate compiling and downloading of the software for the VME CPUs and DSPs. The performance of the integrated system indicates a response time of 500 μ s and a correction bandwidth of 250 Hz at -6 dB noise rejection.

In the near future, one or two local feedback systems will be installed on selected x-ray beamlines. With completion of production of the BPM and corrector power supply interface boards expected at the end of this year (1995), the global feedback system will be implemented early in 1996.

V. REFERENCES

- [1] R. Hettel, "Beam Steering at the Stanford Synchrotron Radiation Laboratory," *IEEE Trans. Nucl. Sci.*, NS-30, No. 4, p. 2228, 1983.
- [2] R. J. Nawrocky et al., "Automatic Beam Steering in the NSLS Storage Rings Using Closed Orbit Feedback," *Nucl. Instr. and Meth.*, A266, p. 164, 1988.
- [3] Y. Chung, et al., "Closed Orbit Feedback with Digital Signal Processing," *Proceedings of the 1994 European Particle Accelerator Conference*, London, U.K., p. 1592, 1994.
- [4] R. Hettel, et al., "Design and Characterization of the SSRL Orbit Feedback System," *Proceedings of the 1994 European Particle Accelerator Conference*, London, U.K., p. 1580, 1994.
- [5] Y. Chung, "A Unified Approach to Global and Local Beam Position Feedback," *Proceedings of the 1994 European Particle Accelerator Conference*, London, U.K., p. 1595, 1994.
- [6] A. Peled and B. Liu, *Digital Signal Processing*, (John Wiley & Sons, 1976).
- [7] Y. Chung, et al., "Local Beam Position Feedback Experiments on the ESRF Storage Ring," *Proceedings of the 1995 IEEE Particle Accelerator Conference*, Dallas, Texas, 1995 (to be published).
- [8] Y. Chung, G. Decker and K. Evans, Jr., "Closed Orbit Correction Using Singular Value Decomposition of the Response Matrix," *Proceedings of the 1993 IEEE Particle Accelerator Conference*, Washington, D.C., p. 2263, 1993.
- [9] Y. Chung, et al., "Development of GUS for Control Applications at the Advanced Photon Source," *Proceedings of the 1994 European Particle Accelerator Conference*, London, U.K., p. 1794, 1994.

Optimization of Synchrotron Beam Alignment using a Linguistic Control Scheme

Roberto Pugliese and Roberto Poboni*

Electronic Technology Laboratory, Scientific Division, Sincrotrone Trieste.

*Department of Electrotechnics, Electronics and Informatics, University of Trieste.

Abstract: The project deals with the design and the implementation of an adaptive controller which optimizes the beam alignment at the SuperESCA beamline, using a behavioural model based on fuzzy logic. The operating environment is an extension of the FuzzyCLIPS¹ expert system shell, integrated in the ELETTRA Beamline Control System.

1. INTRODUCTION

The Sincrotrone Trieste is a research laboratory which built ELETTRA, a "third generation" machine, with a 2GeV state-of-the-art storage ring. ELETTRA is a high brilliance synchrotron radiation facility covering a large spectrum of wavelength from the VUV to hard X-ray region. This radiation can be used for chemical, physical and biological experiments.

According to the requirement of the experiments, the light has to be collimated, monochromatized and focussed on the sample under test; this is realized by a highly sophisticated beamline working in ultra-high vacuum. The Super-ESCA beamline [1] has been designed for high-resolution core-level spectroscopy using soft x-ray synchrotron radiation. It receives the light from an undulator in the storage ring ELETTRA. The tunability of this insertion device and the connected SX700 monochromator allow experiments in the photon energy range 100-1400eV. This beamline, operational since 1994 has shown a resolving power varying from 10000 to 8000 for photon energies between 240 and 850 eV.

The performance of the Super-ESCA beamline is strongly affected by the alignment conditions i.e. all the optical components must be set to optimize the flux and resolving power at the experimental station.

The propagation process of the photon beam through the beamline has a complex structure. Many optimization schemes can be applied under the assumption of the existence of a mathematical model of the process [12]. However such a model may be too complex, it may not be precise or it may not even exist.

Soft Computing methodologies such as neural networks, genetic algorithms, and fuzzy sets theory have been applied to a broad class of such real-world problems. In this work, optimizing control is performed by using a rule-based look-up table which can be learned from operating experience without the need of a process model.

¹ FuzzyCLIPS was developed by the Knowledge System Laboratory, National Research Council of Canada

2. THE BASICS OF FUZZY LOGIC

Fuzzy logic introduced by Lofti Zadeh [2,3] brings reason to the vagueness found in human experiences or to the unclear boundaries in physical processes. For example, even a concept like "normal height" is difficult to express by standard "crisp" computing, but easily described by a characteristic function that has a continuous grade between 0 and 1. This "fuzzy set" or membership function can take various shapes (bell-curve, trapezoidal, triangular, and other).

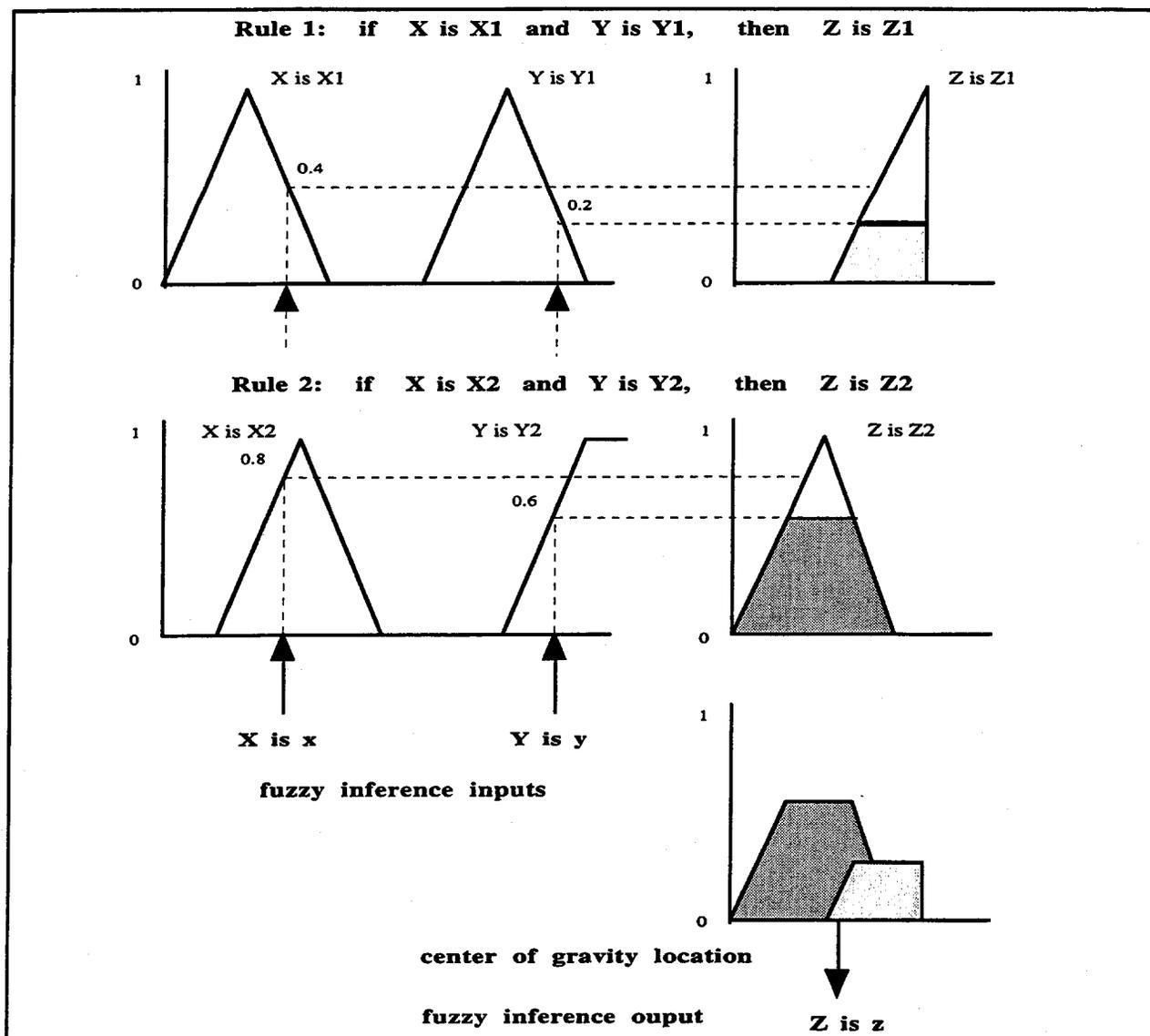


Fig.1: Stages of Fuzzy Processing

Fuzzy Logic theory allows one to work on a linguistic description of the reality, representing information by linguistic variables. Linguistic variables are entities whose values are fuzzy sets. Fuzzy Logic represents a new approach to system modeling. The first known applications of Zadeh's theory were reported by Mamdani [4,5].

The stages of fuzzy processing are illustrated in Fig.1 for a simple case of two linguistic variables (X,Y) combined by means of two rules to form the result Z. Inputs are compared to membership functions by a process called "fuzzification", so a membership grade is assigned to each linguistic term. Logic combinations of linguistic terms are then evaluated by application of Fuzzy Logic operators: usually the minimum grade of the conditions is selected as the fitness grade of the rule left-hand side. Fuzzy values of the rule right hand side are adjusted to reflect the rule fitness grade and then combined with the fuzzy values of the other rules. Center of gravity or other averaging methods are used for "defuzzification", that is, to obtain a single value for each output variable. Many implementations of this kind of inference can be found in literature [9].

So far Fuzzy Logic has been exploited mainly in Knowledge Based Systems and in Control Systems [6-8]. We say that a control system is "fuzzy" when the value of the controlled variables is determined by the input signals through a fuzzy inference. In the same way we talk about Fuzzy Optimization when the optimization process is driven by fuzzy inference.

3. THE PROBLEM STATEMENT

The schematic diagram in Fig.2 is a simplified description of the beamline Super-ESCA. In the drawing only the components relevant to the optical alignment are present; optical components are represented with rounded border. A detailed discussion of each component of the beamline is beyond the scope of this document, but to feel the complexity of the system requires a brief description of the beam path. As the synchrotron radiation beam leaves the undulator, it passes through two photon *Beam Position Monitors* and then through the *Double Slits* where the central part of the beam is selected; it then reaches the *Switching Mirror*, a plane mirror used to switch between the branch lines². Before reaching the *SX700 Monochromator*, the light beam passes the *Prefocusing Mirror*, a cylindrical mirror which focusses the beam in the vertical plane onto the *Entrance Slit*. The Monochromator is the most complex optical component of the beamline, consisting of a plane mirror, a plane grating and an elliptical focusing mirror. The plane grating spreads the incoming radiation in the vertical plane according to energy. Only one selected energy follows the right path to the elliptical mirror, which focusses the radiation on the *Exit Slit*. The *Refocussing Mirror*, an elliptical mirror, refocusses the monochromatized light onto the sample in the *Experimental Chamber*. A photo-diode supplies a signal proportional to the quantity of photons leaving the *Exit Slit*.

The complexity of the optical path makes it difficult to determine the equations relating the quality of the synchrotron light spot at the experimental chamber with the position of the optical components of the beamline. One may realize that the alignment optimization, that is the process of setting the positions of all the optical components to optimize the flux and resolving power at the experimental station, is a very difficult. In practice it is several hour job for a pair of trained operators.

² The Super-ESCA and ESCA-microscopy beamlines share the so-called "front-end", i.e. the part of the beamline from the source to the switching mirror.

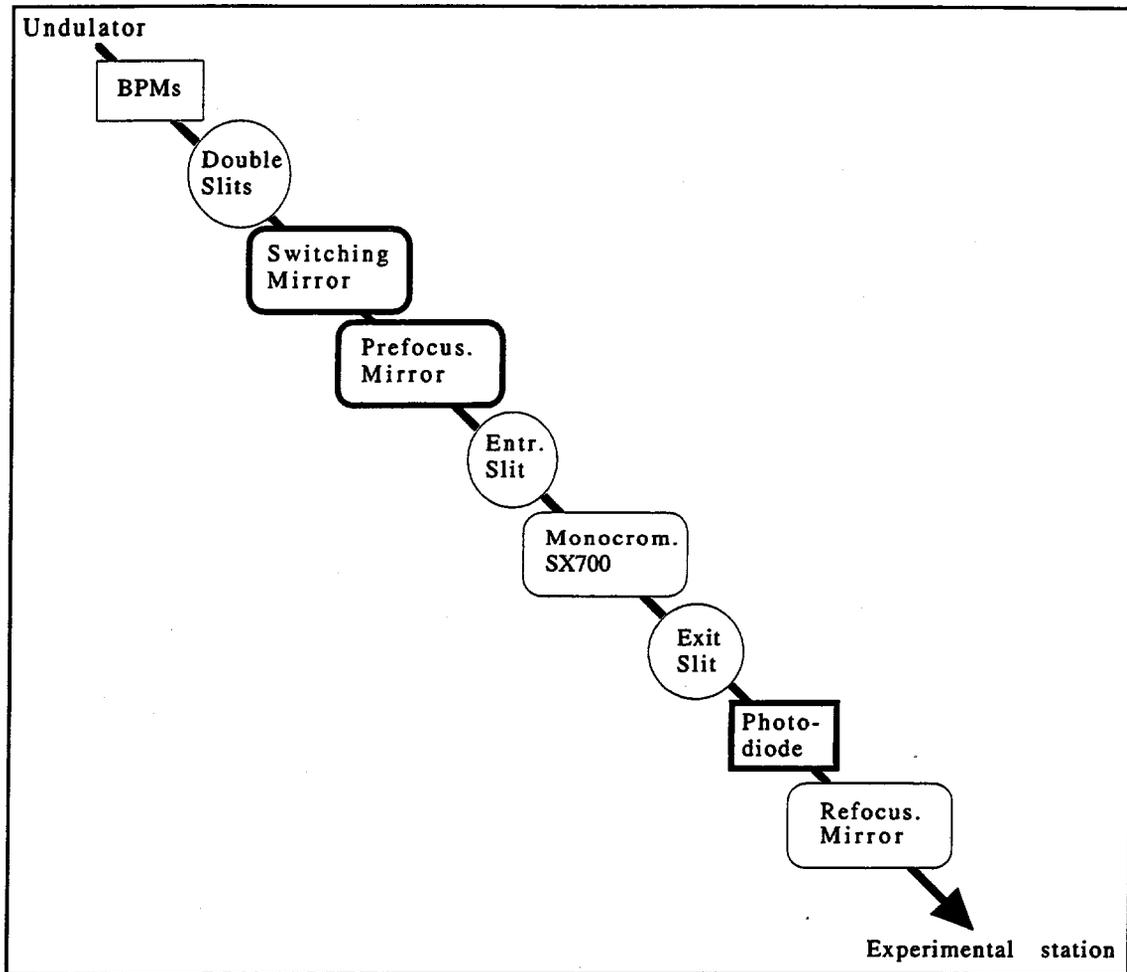


Fig.2: Simplified scheme of the Super-ESCA Beamline

The problem we have actually considered, a non-trivial subproblem of the global alignment optimization, can be formalized as follows: determine the positions of *Prefocussing* and *Switching* mirrors which maximize the flux of photons at the photo-diode, assuming that the *Monochromator* is aligned. Namely, we have considered only the degrees of freedom of the mirrors which can be considered optically critical: the *Pitch* and the *Roll* angular positions. Formally we have to find the maximum of the following function:

$$y = f(x_1, x_2, x_3, x_4)$$

where y is the photo-diode current, x_1 is the *Switching Mirror Pitch*, x_2 is the *Switching Mirror Roll*, x_3 is the *Prefocusing Mirror Pitch* and x_4 is the *Prefocussing Mirror Roll*.

4. THE OPERATING ENVIRONMENT

The Beamline Control System [14] provides supervision and control of the beamline equipment: it has the responsibilities to assure a "safe behaviour of the beamline" and to give the operator a uniform access to the

instrumentation.

The system has a distributed multi-layer architecture (Fig.3). The presentation level is based mainly on UNIX graphical workstations (such as DEC-ultrix, DEC-alpha, HP9000): a synoptic interface implemented with DataViews³ and Motif enables graphical navigation through the beamline instrumentation.

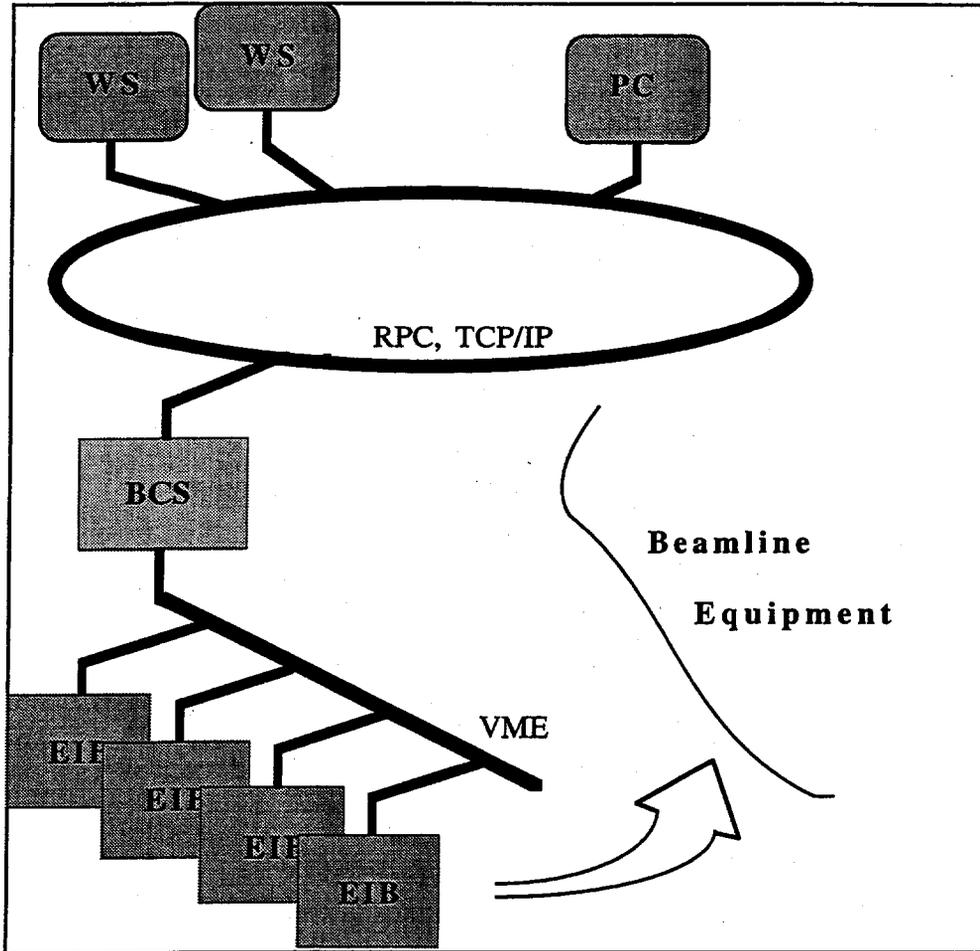


Fig.3: The Beamline Control System architecture

The core of the system, called BCS, is the process control level, based on VME embedded systems using the Motorola 68030 processor (MVME147) and the LynxOS real-time operating system. These two layers are connected by an ethernet LAN via Remote Procedure Calls⁴. The equipment interface level, connected with the BCS by the VME bus, is based on standard data acquisition boards⁵. The application software can be developed both at the presentation level and at the process control level by means of well defined libraries.

³ Is a graphical visualization package developed by V.I. Corporation

⁴ In the current release the CERN network compiler has been used.

⁵ We have named such boards Equipment Interface Boards (EIB).

FuzzyCLIPS is an extended version of the CLIPS rule-based shell⁶ for representing and manipulating fuzzy facts and rules. In addition to the CLIPS functionality, Fuzzy CLIPS can deal with exact, fuzzy, and combined reasoning, allowing fuzzy and normal terms to be mixed freely in the rules and facts of the knowledge based system. The system uses two basic inexact concepts, fuzziness and uncertainty.

We have developed an extended version of FuzzyCLIPS which allows one to perform the basic read and write actions on the BCS controlled variables. The optimization application was implemented at the process control layer.

5. THE FUZZY OPTIMIZER

The design of the optimizer was guided by the domain knowledge of the expert operator and by the analysis of the qualitative behaviour of the objective function; an operational model of the beamline was implemented with the Ray Tracing Shadow⁷ simulation program.

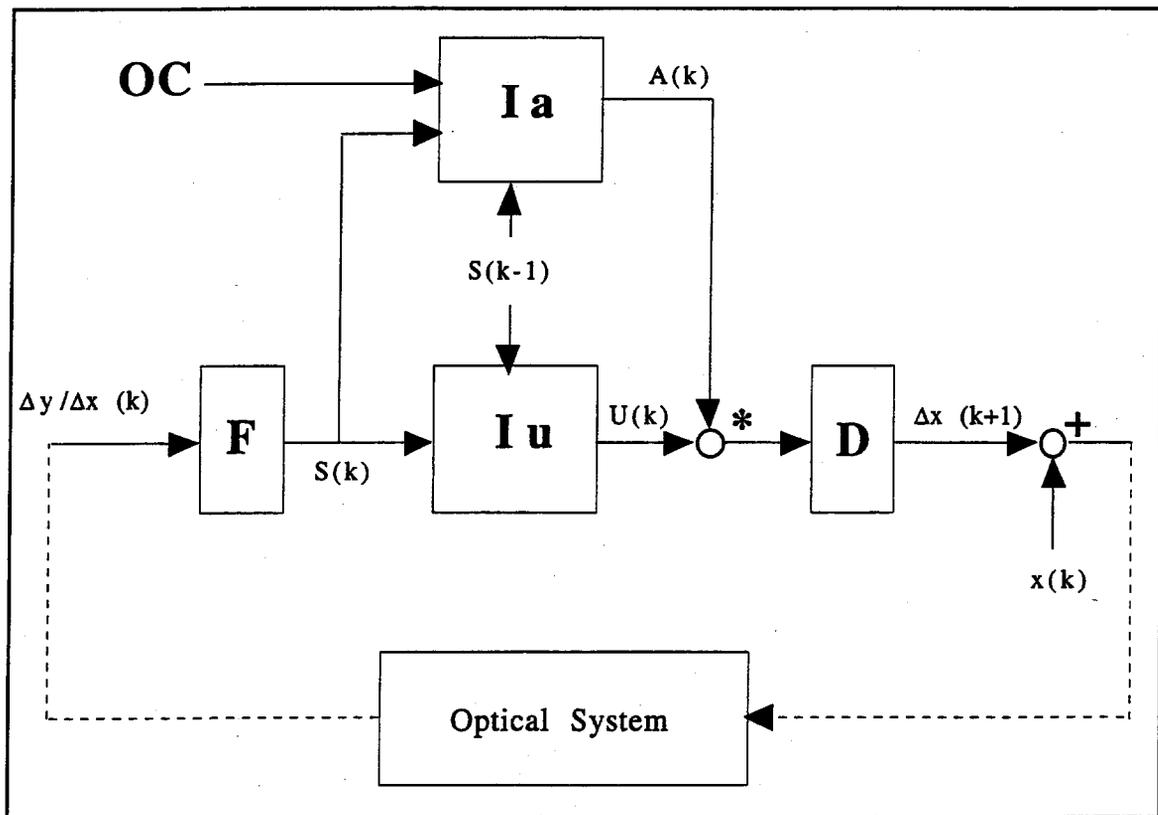


Fig.4: Simplified block diagram of the controller.

The optimization control scheme we have developed is a modified version of the linguistic control scheme developed

⁶ CLIPS was developed by the Artificial Intelligence Section, Lyndon B. Johnson Space Center, NASA.

⁷ Ray Tracing Shadow was developed by the University of Wisconsin.

by Procyk and Mamdani [10,11]. The independent variables, namely the mirror positions, are optimized in a sequence determined by a knowledge based planner. Each variable is optimized by a sophisticated hill climbing technique: a simplified block diagram is shown in Fig.4.

At each step the block *F* processes the slope of the objective function by performing normalization and fuzzification. The block *Iu* takes the normalized slope at the current and previous step and by fuzzy inference provides the magnitude of the correction of the mirror position (*angle change*) which is then defuzzified by block *D*. The direction of the correction is determined in the natural way by the sign of the current slope. If we call *S(k)* the normalized slope and *U(k)* the angle change, a typical linguistic rule can then be written in the following form:

$$\text{if } S(k) \text{ is } S_i \text{ and } S(k-1) \text{ is } S_j \text{ then } U(k) \text{ is } U_h.$$

The fuzzy sets of the normalized slope and angle change are shown in Fig.5. The linguistic rule set can be represented by the relation matrix or look-up table of Fig.6.

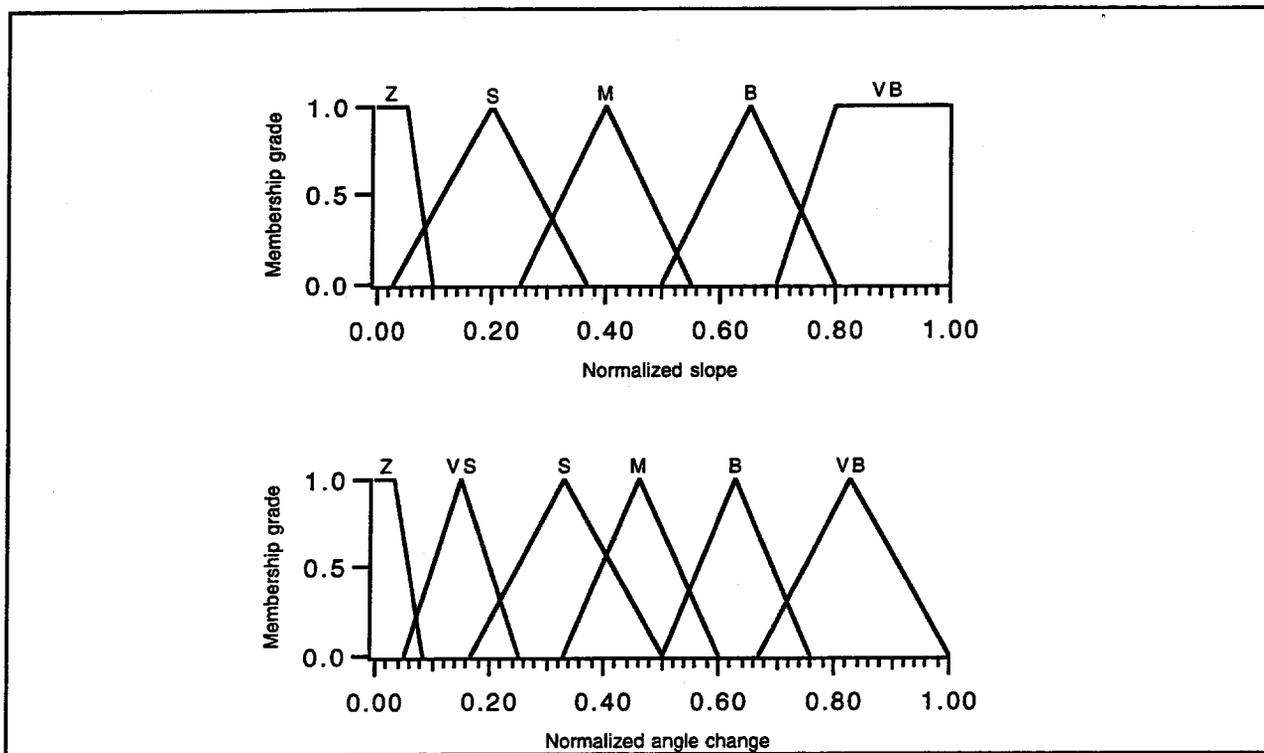


Fig.5: Normalized slope (*S*) and angle change (*U*) membership functions. The labels can be read as follows: *Z* as zero, *VS* as very small, *S* as small, *M* as medium, *B* as big and *VB* as very big.

We have also introduced an adaptive component in the fuzzy controller. The method described in Maeda-Murakami [13] consists of realizing an adjustment function which determines, depending on the operational conditions of the controller, a scaling factor for the *angle change*. As shown in the block diagram shown in Fig.4, the module *Ia* takes

the normalized slope at the current and previous step and depending on the operational conditions OC , determines by fuzzy inference the scaling factor $A(k)$. A knowledge based module sets OC by observing the optimization process.

U(k)		S(k-1)					
		n.a.	Z	S	M	B	VB
S(k)	Z	VS	Z	Z	VS	Z	VS
	S	VS	Z	VS	VS	VS	Z
	M	S	VS	S	M	VS	S
	B	M	S	VB	B	B	S
	VB	S	M	B	M	M	S

Fig.6: Linguistic control rules

An example of the behaviour of the controller with and without the adaptive component is shown in Fig.7; the performance gain provided by the adaptive module is generally valuable.

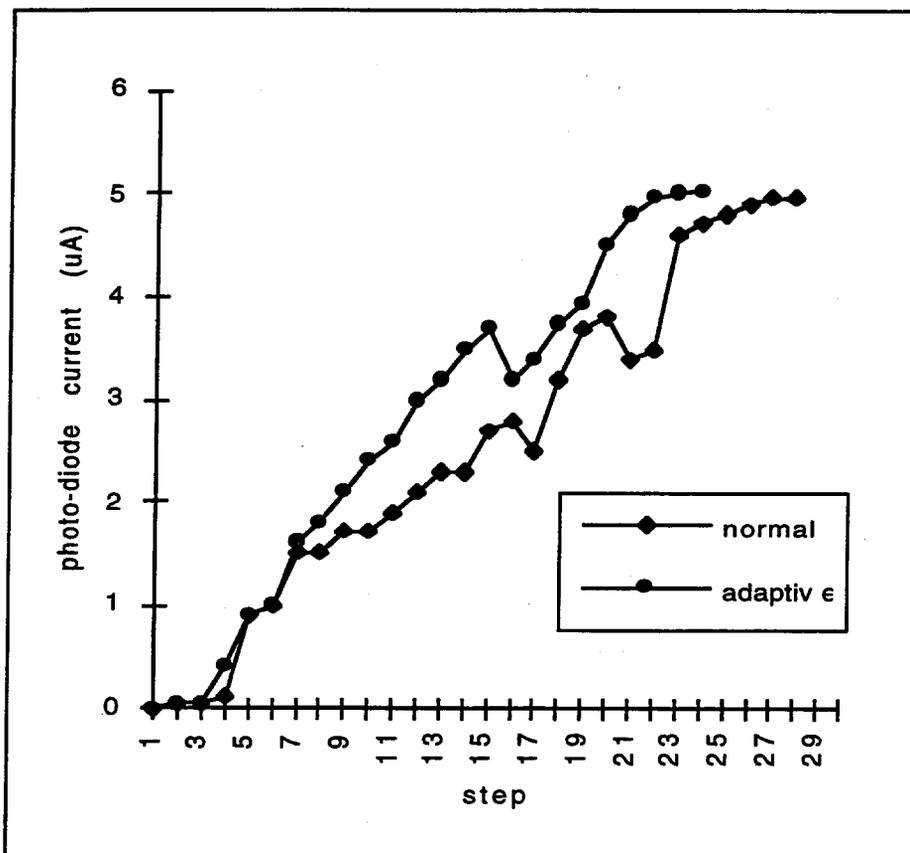


Fig.7 the behaviour of the optimizer

6. CONCLUSION

We have implemented an adaptive controller which optimizes the beam alignment at the beamline Super-ESCA by using a behavioural model based on the fuzzy logic. The system is now under test and its behaviour is quite satisfactory both on the simulator and in the field.

Further improvements could be obtained by using the beam status information provided by the Beam Position Monitors; fuzzy logic could be used as a tool to implement a qualitative optical reasoning component.

The introduction of soft computing techniques can be considered a meaningful step in the extension of the application areas of the Beamline Control System.

Acknowledgements

Special thanks are due to the staff of the beamline Super-ESCA, in particular to Daniele Cocco and to the staff of the Electronic Technology Laboratory, in particular to Rudi Sergio and Fulvio Billè.

References

- [1] W.Jark, *Soft X-ray monochromator configuration for the ELETTRA undulator* - Rev.Sci.Inst. 63(1) 1992.
- [2] L.A.Zadeh, *Fuzzy Sets* - Inform. Control 8, 1965, pag. 338,353.
- [3] L.A.Zadeh, *Outline of a new approach to a new analysis of complex system and decision processes* - IEEE trans. on Syst. Man and Cybernetics, 3/1973, pag. 28-34.
- [4] E.H.Mamdani, Assilian, *A fuzzy logic controller for dynamic plant* - Int. J. Man-Machine Stud. 7, 1975, pag.1-13.
- [5] E.H.Mamdani, *Application of fuzzy algorithms for control of simple dynamic plant* - Proc. IEEE 212, 1974, pag. 1585-1588.
- [6] R.Scattolini, N.Schiavioni, *Introduzione al Controllo fuzzy* - Automazione e Strumentazione, Aprile 1993, pag. 67 - 73.
- [7] G.A.Magnani, *Esempi di applicazione nei sistemi di controllo* - Automazione e Strumentazione, Giugno 1993, pag. 83 - 88.
- [8] L.X.Wang, M.Mendel, *Generating fuzzy rules by learning from examples* - IEEE transaction on System, Man, and Cybernetics, vol. 22, No. 6, november/december 1992, pag. 1414 - 1427.
- [9] J. and S.R. Jang, *ANFIS: Adaptive-Network-Based Fuzzy Inference System* - IEEE transaction on System, Man, and Cybernetics, vol. 23, No. 3,1993 pag. 665 - 685.
- [10] T.J.Procyk and E. H. Mamdani, *A linguistic self-organizing process controller* - Automatica, 15, (1979), 15.
- [11] S.S. Jang, D.S.H.Wong and C.K.Liau, *On-line/off-line optimization of complex processes using a linguistic self-organized optimizing control scheme* - Fuzzy sets and systems, 47, (1992), pag. 23 - 33.
- [12] G.V.Reklaitis, A.Ravindran and K.M.Ragsdell, *Engineering Optimization: Methods and Applications* - a Wiley-Interscience Publication, John Wiley and sons.
- [13] M.Maeda and S.Murakami, *A self-tuning fuzzy controller* - Fuzzy sets and systems 51, 1992, pag. 29 - 40.
- [14] A.Abrami, F.Gagliardi, A.Galimberti, A.Savoia, *The ELETTRA Beamlines Control System* - ST/S-TN-92/9, 1992.

Software Engineering Techniques in the CERN RD-38 (CICERO) Project

R McClatchey¹, J-M Le Goff², N.Baker¹, R Barillere², C Escrihuela², E.Bernard³

¹Dept Of Computing, Univ West of England

²ECP Division, CERN

³SPACEBEL Informatique

Abstract: The complexity of the software required in the forthcoming control systems for LHC experiments necessitates a re-think of the strategies employed in the design of such systems. Use of software engineering standards and systems development methodologies are required to ensure the success of High Energy Physics (HEP) control system design. The CERN RD-38 (CICERO) project aims to use modern software engineering methods such as Object Orientation to design the main building blocks of a generic control information system which will be based on the distributed object standard, CORBA [1]. This paper outlines conclusions that can be drawn from the use of Object Orientation and the ESA standards in the design of Cortex [2], an integrating environment which enables user control objects to be 'plugged and played' in CICERO.

1 The Cortex approach to Designing HEP Control Systems

The large number and variety of parameters that require monitoring in HEP experiments as well as the complexity of the software to perform that monitoring and control has forced systems designers to reconsider the entire design of HEP control systems [3]. The increase in the numbers of sensors and in functionality required by these experiments has led inevitably to the construction of very heterogeneous and distributed control systems where integration and communication between the different detector (sub-)systems has become an issue. It is only by reducing the apparent complexity of the control system that the operation of these systems and their overall maintenance can be efficiently handled by physicists. In addition, since these experiments evolve over time, any HEP control system must also be scalable and flexible to change and provide reusability of its constituent components.

To help reduce development costs and to simplify the apparent complexity of the HEP control systems, physicists are looking for partially reusable solutions to their technical problems. However, in the recent past it has proved difficult to integrate commercial products together (such as PLCs with front-end VME crates) and to integrate these products with existing CERN-made control (sub-)systems [4]. In essence, what is required by those responsible for HEP control systems is an overall framework to facilitate integration between control system components. Such a framework (or software integration platform) should go beyond defining standard interfaces; its design should guarantee that commercial products and 'home-grown' systems can exchange information and collaborate regardless of the organisation of the overall control system. The CICERO research and development project, involving both academic institutes and industrial partners, aims amongst other things to investigate the use of Object Oriented Design (OOD) techniques and distributed system standards in providing an integration framework (Cortex) to simplify HEP control systems design. The Cortex design concepts are described in another paper contributed to this conference [5].

Experience of the development of control systems for the LEP experiments [3] and anticipation of the increased demands which will be generated by the new and larger experiments for LHC, has identified the main constraints for the design of such an integrating scheme. Firstly, since the LHC experiments will be equipped with > 100,000 sensors and activators, the *management of control system complexity* is a major consideration. Object-oriented design techniques embodying abstraction, inheritance and the use of classes and objects will aid the design here. Secondly the problem of *concurrent and collaborative software engineering* requires addressing. This is particularly true when the development of the control software is carried out by engineers who are separated geographically. Thirdly, the software developed should provide *stability, flexibility and availability* of control system elements so that system downtime (both for repair and for upgrades) is minimised. Finally, the control system software should provide *balanced, distributed processing* in the heterogeneous environment of High Energy Physics (HEP) control systems.

The architecture of any distributed control system will inevitably change during the lifetime of the accelerator or the experiment it is operating. As a consequence, Cortex must support a mechanism to allow a new version of the distributed control system to be in preparation off-line while an older one is operated on-line. Furthermore this off-line/on-line facility is needed since tests and validation (and possibly simulations) of a new configuration will be needed before it is applied to the operating on-line system. These constraints have led to a so-called dual-face approach being taken in CICERO for the Cortex integrating framework:

* an off-line Cortex Repository is proposed to handle the logical descriptions of the architecture of the distributed

control system and to describe the various information and commands to be exchanged between the different components.

* an on-line Cortex Infrastructure is proposed comprising so-called Distributors through which the User (control) Components can exchange information and commands in a pseudo- 'plug-and-play' fashion. A generation mechanism is provided to facilitate updates of the on-line Infrastructure according to the Repository contents.

The physicist then builds a logical model of the required control system architecture off-line in the Cortex Repository, specifying detail at the configuration level to enable updates to be carried out on the existing architecture. These updates are validated against the existing architecture and the generation procedure instantiates the updates and creates the new updated Cortex Infrastructure. The details behind the operation of Cortex have been dealt with elsewhere [6] & [7]; this paper examines the design approach taken in implementing the building blocks behind Cortex. In particular, section 2 investigates constraints on the Cortex design approach and section 3 outlines the methods followed in delivering a Cortex prototype. A critique of this approach is presented in section 4 and conclusions are drawn in the final section.

2 The Need for Software Engineering in the Design of HEP Control Systems

Historically, high energy physicists have been slow in adopting the software engineering methods of computer scientists. The systems developed in HEP have often responded to the development needs on an ad-hoc basis, with many short term solutions to similar problems [3]. Structured software development methods have been used [8] for the analysis and design of control systems for HEP experiments at LEP. Since then, experience of structured software engineering methods and CASE tools has increased at CERN [9], [10]. As yet, however, there have only been a few examples of the use of OOD methods in the design of control systems at CERN. One notable exception is that of Myers et al [11] which investigated the use of OOD in a controls environment. In industry, OOD techniques are more prevalent and have been used in the design of control systems for Supervisory Control And Data Acquisition (SCADA). Ericsson et al [12], advocate both the use of object design techniques and the use of CORBA as the distributed communication platform for control systems. The CICERO project is using both OOD and CORBA to implement a generic control information system for LHC experiments.

As HEP control systems evolve for LHC, the need for rigor in software engineering increases in importance. In addition, as the development of these complex systems is increasingly carried out remotely from CERN or by developers on short-term contracts at CERN, the need for clearly defined deliverable code and interfaces between (sub-)systems also becomes crucial. As a consequence, software engineering standards have been investigated in the last few years by large groups of developers in HEP. Furthermore, work at the European Southern Observatory [13] into standards has recommended the use of the European Space Agency software engineering standards alongside those from the IEEE and IEE.

LHC experiments will involve collaborations of many tens of institutes and over 1,000 physicists, engineers and computer scientists from around the world. The knowledge required to construct and monitor the hearts of the experimental detectors will be distributed between these institutes making it difficult to impose standards. There will consequently be significant problems of information transfer in ensuring that each detector subsection retains autonomy of control but can work with other detector subsections for data-acquisition. In addition, the LHC detectors will be required to have a long lifecycle, since the experiments will take data for several years and as a consequence maintainability will be an important consideration. Clearly, to reduce problems of interoperability and integration the project needs to adopt a set of standards both for the development of the CICERO software and its operation across heterogeneous hardware platforms [5]. The methods used and the standards followed must enable collaborative working in a systematic manner and adherence to a strict discipline. Support tools are required to provide cross-checks of the design and to reduce the possibility of error in the construction of design models.

3 The Choice of Methods and Standards in CICERO

The European Space Agency Procedures, Specifications and Standards [14] method has been adapted as an essential feature of CICERO. This standard has been developed for the European Space Agency to ensure that any project has the best chance of being successful. There are two major parts: the product standards themselves and the procedures standards used to produce them. The products are the documents and software used to create, use and maintain software. The procedures guide the system developer in software project management, software configuration management, software verification and validation and software quality assurance. The documentation for this standard includes both mandatory and optional sections, divided into three levels. In order to meet the ESA standard, all mandatory operations must be carried out or documentation produced as appropriate. The process of production is divided into six phases, following the standard life-cycle model of User Requirements Definition, Software Requirements Definition, Architectural Design, Detailed Design and Production, Transfer and Operations & Maintenance. In addition there are documents on Structured Analysis, Fortran Coding, Ada Coding & C Coding standards.

Management of the software lifecycle is catered for in the ESA standards through the use of a Software Configuration Management Plan (SCMP), a Software Verification and Validation Plan (SVVP) and through Software Quality Assurance (SQA). These plans are detailed in [14] and provide the project manager with requirements for identifying, controlling, releasing and changing software releases and for recording their status. The SVVP provides for the review, testing and auditing of the delivered software products. Further, the project manager can ensure quality is being main-

tained in software delivery by following the recommendation of the SQA plan.

The ESA standards were originally based on the 'Waterfall Model' of the software lifecycle, following a phased approach to software development. Modified forms of this approach include the incremental delivery and evolutionary development models. In the development of Cortex an *evolutionary* approach has been adopted which overcomes some of the limitations of the waterfall model (eg implementation only in full and at project completion). The evolutionary approach allows for the planned delivery of multiple releases of Cortex, with each release incorporating the experience of earlier releases in a manner analogous to the 'Spiral Model' of software development suggested by Boehm [15]. Given the often speculative nature of the research in CICERO and the mechanisms of staged funding, following the evolutionary approach to systems development reduces the inherent risk of project failure.

The ESA standards do not prescribe a method to follow in designing software systems, although they promote a functional decomposition approach to systems design. However, the nature of HEP control systems - their complexity, the distribution of components across heterogeneous systems and the set of fundamental system services required in these systems (such as replication, migration and failure and error recovery) point to the use of an object encapsulation approach as an appropriate mechanism of hiding and managing system complexity. Such objects can improve inter-operability due to their semantic richness and can be combined in novel ways to promote information processing and minimise code modification. Furthermore, adopting an Object Oriented approach in design facilitates the encapsulation of existing non-object based (legacy) systems so providing software reuse and a clear upgrade path for control systems designers. Consequently, an Object Oriented approach has been adopted in CICERO.

In the development of CICERO, Object Oriented programming languages (eg C++), Object Oriented Databases (eg Gemstone, O2) and object-based distributed systems standards (CORBA [1]) are being investigated. In integrating such technologies, an object oriented design approach is required to be followed which is sufficiently mature, academically sound and supported by available Computer Aided Software Engineering (CASE) tools. Foremost amongst these is the method developed by James Rumbaugh and colleagues, OMT [16]. The OMT method comprises a number of models which are developed and enhanced as the project moves from requirements analysis through design to implementation. OMT comprises three models and four software development stages: the Object, Dynamic and Functional Models and the phases of Analysis, System Design, Object Design and Implementation.

Of widest use on the CICERO project have been the Object/Class model of OMT (a form of Extended Entity Relationship model showing static data relationships in the software), event traces (showing sequences of messages sent between objects to accomplish a given function) and the Dynamic Model (State Charts to define the temporal ordering of events impinging on a given object). The Functional Model which encompasses data-flow diagrams and function definitions has not been used much in CICERO. The Object and Dynamic models have been supported by the use of a diagramming tool, Select initially, and latterly Westmount Case. In addition, the CICERO project has used a code generation tool, OBLOG, which supports the OMT methodology, to generate User Components (in C++) from the OMT Object/Class model. The OBLOG tool integrates concepts from semantic data modelling and concurrent processing, employing objects as a unifying concept and aims at a conceptually seamless methodology from requirements to implementation [17].

4 A Critique of the Use of OMT and the ESA Standards

The software quality assurance process of ESA enforces the classic waterfall model of software engineering by requiring that no phase can begin until the previous one is complete. While Object Oriented Design is referred to in the Guide to the Software Architectural Design Phase [14] and Object Oriented Programming is mentioned in the Guide to the Software Detailed Design and Production Phase [14], the specified method for architectural design is largely a top down approach. Therefore in using OMT with the ESA standards some mapping from the four software development phases of OMT onto the six phases of the ESA standards is required.

CICERO has mapped OMT on to ESA in the following manner: The User Requirements phase of ESA is mapped onto a text-based definition of the problem statement in OMT. The Software Requirements then maps directly onto the Analysis phase in OMT where an understanding of the application domain is achieved and a model of the system is developed. The OMT phase of System Design then matches that of Architectural Design in ESA when the overall architecture of the system to be developed is determined. Next, the Object Design phase of OMT is mapped onto the Detailed Design and Production phase of ESA when the models are consolidated and the final system design is specified in detail. The Implementation phase of OMT corresponds to the Transfer and Operations and Maintenance phases of ESA (see figure 1).

Following the ESA standards and an evolutionary lifecycle model, the Cortex element of CICERO has gone through the first spiral of User and Software Requirements specification followed by Architectural and Detailed Design resulting in a demonstrable prototype. The disciplined use of the ESA methods during the prototype development undoubtedly helped to clarify the design of the prototype system and to focus the effort of the designers who were geographically remote (in the UK, Belgium, India, Hungary and CERN). Such a use of the ESA standards does, however, require care, especially in identifying the boundaries between the different phases of the lifecycle. In principle, ESA standards do not enforce any particular methodology. In practice, however, the ESA guidelines clearly support a functional breakdown of the problem statement and hence imply non-object-oriented methodologies. Some trade-offs had to be made to support the OMT methodology, especially in the Architectural and Detailed Design phases of the

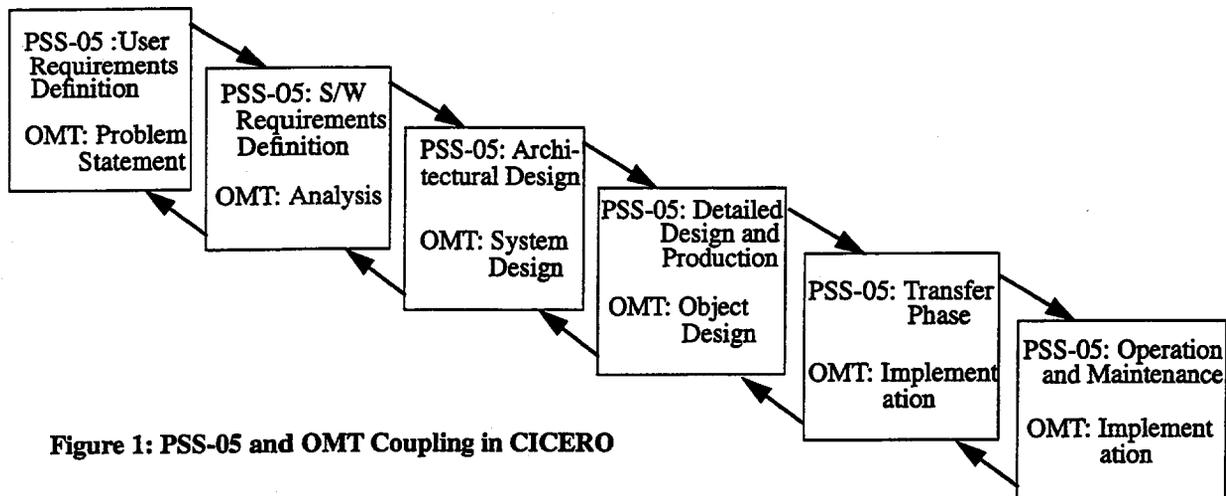


Figure 1: PSS-05 and OMT Coupling in CICERO

ESA standards. One example is in resolving the relationship between the demands on the documentation produced in accordance with the ESA standards, particularly in the prominence given to natural language in the User and Software Requirements documents, and the need for more precise semantics demanded by the OMT modelling approach.

Software produced for any pilot project must be similar to production software with respect to robustness and reliability. Therefore management guidelines supporting the software lifecycle as enhanced by the ESA-PSS 05 standard should be strictly followed. In particular, as stated earlier, a complete set of Software Project Management, Software Configuration Management, Software Verification and Validation and Software Quality Assurance plans should be produced in parallel with the software lifecycle documents. This is often overlooked in the non-software engineering world of HEP and in particular when costs are a major constraining factor. It is the experience of CICERO that these important software management procedures should be addressed in any application of the ESA standards.

The OMT notation itself is strong in describing the abstract models used in object design, but is lacking when describing the fully implemented system. This failing was particularly noticeable during the Architectural and Detailed Design phases of the prototype. Yet it is here, when the volume of design information increases substantially, that notations to capture the design and tools for their manipulation are essential. Work on design notations and the capture and re-use of designs is the focus of some researchers in the OO field and the results of this work should be consulted by groups using the ESA standards with object-oriented design techniques. OMT is not particularly targeted at the design of real-time systems and is not sufficient to handle concurrency. As a result, the treatment of some issues of concern in the development of Cortex have been inadequately addressed by OMT. Some of these issues are better handled by later developments of the OMT approach, notably the Syntropy method [18]. Syntropy provides additional rigour to the OMT method, particularly in the design of cooperating processes. This is a potential area of future research in CICERO.

It was clear during the prototype development phase that in normal discussion about Cortex, designers and users make great use of concrete examples of usage of the system and specific, often important, examples of message sequencing within the implemented system. It is felt that the efficiency of the design process and of communication of the design to the users could be improved by the incorporation of scenarios and use cases into the formal documentation and design process following the Jacobson approach [19]. The most appropriate phase for incorporation of such use cases is clearly the User Requirements specification of ESA. However, these familiar scenarios are likely to become part of the vocabulary of the project team and be referred to at subsequent stages of the lifecycle thereby adding clarity to the design discussion.

The quality of tools support for methods remains an issue for CICERO. Platform specific tools, difficult access to design repositories and the lack of support for multi-user and multi-site development, lead to a loss of efficiency. In particular, this can generate additional conversion and cross-checking tasks. Methods for managing versions of documents, supporting relationships between models from different viewpoints on the design and tracking the process of design decision making are all desirable in the selected toolset. As yet there is no obvious single tool for supporting the use of OMT in developing control systems. Tool evaluation, selection and guidelines in their use in an integrated manner remain tasks for the next phase of the CICERO project. The use of CASE tools like OBLOG for the automated generation of CORBA compliant User Component code will be investigated further, especially to increase the performance of the generated code, to deal with more complex data types and to support concurrent access etc.

5 Conclusions

Since being approved in February 1994, the RD-38 project has grown and continues to attract further commercial and academic research interest. A demonstration prototype has been constructed by the CICERO group to validate the ideas proposed by the collaboration and to illustrate new concepts. This prototype is a small scale illustration of a dis-

tributed control system for a real HEP experiment. This prototype is operational and has been used to carry out some basic performance tests. Preliminary results have been obtained, using different user components and distributor configurations and the quantitative results have been reported elsewhere [7].

A clear conclusion to be drawn from the prototyping phase is that standards, while necessary for the structured development of a project, especially involving widely dispersed collaborators, can introduce problems unless they are well integrated with methods and tools. In CICERO their disciplined use has demonstrated that a widely dispersed collaboration can work together even under tight time and resource constraints to produce a functioning system. However, the deliverable-driven approach encouraged by the use of the ESA standards is not entirely compatible with an Object Oriented approach to design and with the prototypical nature of the CICERO research. The prototype has nevertheless been successful in, amongst other things, demonstrating the limits to which the ESA standards and OMT can be used in CICERO and in investigating automatic code generation of Cortex User Components using the OBLOG CASE tool. It has successfully followed a mapping between the waterfall model of ESA and the object oriented design of OMT. Recently the next phase of CICERO has been approved. By mid 1996 the project will build a pilot project in an existing HEP experiment at CERN (L3). This will enable further investigation of the ESA standards and OOD techniques (following the evolutionary development approach [20]) and greater use of support tools such as Select, Westmount and OBLOG CASE tools. In addition, the techniques of Jacobson and Syntropy will provide a fruitful area of further research as identified in the previous section.

Within a year the CICERO collaboration has demonstrated that it is possible to build an heterogeneous distributed control application using OOD techniques, commercial products and to deliver high level functionality like alarm filtering, user assistance and on-line documentation to help the user operate the control system. Much work remains in CICERO Phase II to provide a system which could be used in practice. It is expected that CICERO Phase II will reach this goal and will prepare the ground for a final consolidation phase yielding a set of software building blocks to enable the implementation of both LHC experiment control systems and industrial complex control systems.

6 Acknowledgments

The CICERO project involves the following groups to which the authors extend thanks both for financial support and for technical assistance: BARC (Bombay, India), CERN (Geneva, Switzerland), CIEMAT (Madrid, Spain), IVO International (Helsinki, Finland), KFKI (Budapest, Hungary), OBLOG (Lisbon, Portugal), SEFT (Helsinki, Finland), SPACEBEL (Brussels, Belgium), UID AB (Linkoping, Sweden), USDATA (Dallas, USA), UWE (Bristol, UK), VALMET Automation (Tampere, Finland) and VTT (Oulu, Finland).

7 References

- [1] CORBA, The Common Object Request Broker: Architecture and Specifications, OMG Pubs 1992
- [2] Barillere R et al., "The Cortex Project: A Quasi- Real-Time Information System to Build Control Systems for High Energy Physics Experiments". Nuclear Instruments & Methods A 352 (1994) pp 492-496.
- [3] Barillere R et al., "Ideas on a Generic Control System Based on the Experience of the Four LEP Experiments Control Systems". Presented to the ICALPECS '91 Conference, Tsukuba, Japan, Nov 11-15 1991, pp 246-253
- [4] Dalesio L R et al., "The Experimental Physics and Industrial Control System Architecture: Past, Present and Future", Nuclear Instruments & Methods A 352 (1994) pp 179-184
- [5] Barillere R et al., "CICERO: Control Information system Concepts based on Encapsulated Real-time Objects". Invited talk at the ICALPECS '95 Conference, Chicago, USA, October 30th - November 3rd 1995.
- [6] Govindarajan G et al., CICERO: 1994 Status Report. CERN RD-38/LHCC/95-15.
- [7] Le Goff J-M & McClatchey R., "CICERO RD38: A Distributed Information System for HEP Controls". Proc of the CERN School of Computing, Arles August 1995.
- [8] Charity T., McClatchey R. & Harvey J., "Use of Software Engineering Techniques in the Design of The ALEPH Data Acquisition System". Computer Physics Communications Vol 45 (1987) No 1-3 pp 433-437
- [9] Buono S et al "Software Engineering Techniques and CASE Tools in RD13", Nuclear Instruments & Methods A 352 (1994) pp 383-385
- [10] Mato P et al., "The New Slow Control System for the ALEPH Experiment at LEP", Nuclear Instruments & Methods A 352 (1994) pp 247-249
- [11] Myers D. et al "Use of Object Oriented Techniques in a Beam-Line Control System", Proc of Int Conf on Computing in High Energy Physics, 21-27 Apr 1994, San Francisco, USA.
- [12] Ericsson G et al., "Structured Development of Industrial Control Systems for Power Networks", 2nd IEEE Conference on Control Applications, Sept 13-16 1993, Vancouver, Canada
- [13] Filippi G., "Software Engineering for ESO's VLT Project", Nuclear Instruments & Methods A 352 (1994) pp 286-389
- [14] ESA PSS-05-02. ESA Board for Software Standardisation & Control (BSSC), 1991
- [15] B. W. Boehm, 'A Spiral Model of Software Development and Enhancement', IEEE Computer 21 (5), pp 61-72 (1988).

- [16] Rumbaugh J et al, Object Oriented Modeling & Design. Prentice Hall 1991
- [17] Ehrich H-D., "Fundamentals of Object Oriented Information Systems Specification & Design: the OBLOG / TROLL Approach", Nuclear Instruments & Methods A 352 (1994) pp 375-378.
- [18] Jacobson I., Object Oriented Software Engineering - A Use Case Approach. Addison-Wesley 1992.
- [19] Cook S. & Daniels J., Designing Object Systems - Object Oriented Modeling with Syntropy. Prentice Hall 1994.
- [20] Mazza C., "Controlling Software Development"., Nuclear Instruments & Methods A 352 (1994) pp 370-374

Managing an Internationally Distributed Software Project

Richard J. McGonegal and Stephen B. Wampler

The Gemini 8-m Telescopes Project
950 N.Cherry Ave., Tucson AZ 85726

ABSTRACT

The Gemini 8-m Telescopes Project is faced with the challenges of demanding scientific requirements, a fixed budget, and an aggressive schedule. In addition Gemini is an international project and the majority of the work itself is being done by groups within the partner countries. In order to meet the requirements on budget and on schedule Gemini has had to adopt a development methodology tailored to its specific circumstances. This paper will detail the engineering practices put in place to ensure quality, software and hardware standards adopted, and the management techniques and tools use to promote success. In addition this paper will describe some of the lessons learned and, most importantly, what the author would change if this were repeated.

Keywords: telescopes, development methodology, distributed management, software project

INTRODUCTION

The Gemini Project is an international partnership to build two 8-meter telescopes, one on Mauna Kea, Hawaii, and one on Cerro Pachon, Chile . The telescopes and auxiliary instrumentation will be international facilities open to the scientific communities of the member countries. The international partnership is made up of the United States, the United Kingdom, Canada, Chile, Argentina, and Brazil. The telescopes will be high performance, 8-meter aperture optical/infrared telescopes and have a planned completion date of 1998-2000.

CHALLENGES

Scientific Requirements

The goal of the telescopes is to exploit the best natural observing conditions and to undertake a broad range of astronomical research programs within the national communities of the partner countries. In order to reach this goal Gemini has set demanding requirements in terms of image quality, tracking, pointing, and availability (Table 1)

TABLE 1.

Specification	Requirement	Description
Image Quality	0.1 arcsec	increase in 50% encircled energy diameter
Tracking	0.044 arcsec	rms jitter in line of sight
Pointing	3.0 arcsec	in service pointing
Availability	98%	time collecting science photons

In addition Gemini has requirements to support a number of different observing modes.

- *Classical Observing* - user manually sequences the different telescope subsystems to acquire data
- *Preplanned Observing* - user plans detailed use of facility in advance and submits this to facility
- *Queue Scheduling* - parts of one observer's program are interspersed with those of others in order to make more efficient use of facility
- *Flexible Scheduling* - programs to be executed are selected depending on environmental conditions in order to make efficient use of existing conditions
- *Service Observing* - program is executed by a member of observatory staff - not the Principal Investigator

These different modes create a requirement for a programmable upper layer to the software - which Gemini calls the Observatory Control System. This system acts to synchronize and sequence the actions of the different subsystems which make up the observatory system.

The support for these different observing modes has been a very real factor in the planning for Gemini - the Gemini Science Committee has recommended that Gemini set as a goal that 50% of all observing be done in a "non Classical" manner in order to take advantage of the unique characteristics of the site.

Aggressive Schedule

The Software project started in 1992 and has the following milestones:

- Sep'93 - Critical Design Review of Software System Design
- Apr'95 - Delivery of User Interface Prototype
- May'96 - Delivery of Alpha 1 Prototype
- Aug'96 - Delivery of Alpha 2 Prototype
- Jan'97 - Delivery of Alpha 3 Prototype
- Aug'97 - Delivery of Beta System
- Jan'98 - First Released Version of Software
- Dec'98 - First Light; Mauna Kea
- Apr'98 - Acceptance of Mauna Kea Software System
- Jun'00 - First Light; Cerro Pachon
- Oct'01 - Acceptance of Cerro Pachon Software System

One of the major differences between a new telescope project and adding a subsystem to an existing telescope is that everything is being done in parallel. This means that, quite often, it is only the interfaces which can be defined, the software on either side of the interface does not exist yet.

Fixed Budget

Gemini is a fixed price project - both of the telescope systems must be delivered within a price envelope of \$176 million, including inflation. This means that the specification, the schedule, and the budget are all tightly interlinked and constrained. What this means in practice is that schedule is just as important as budget.

As Gemini must pay for labor from the same pool of funds from which it pays for capital expenses purchasing commercial products makes good financial sense as they are, in general, quite inexpensive compared to the effort required to duplicate them and, especially, the effort required to maintain the duplicated software. It also means that it can cost more to be late than it does to hire more staff and be early - especially when inflation and currency exchange fluctuations are taken in to account.

INTERNATIONAL COLLABORATION

The Gemini system is being produced as a combination of internationally bid contracts and allocated work packages. In general the allocated work packages represent work in which the partners possess intellectual interest and existing skills. In the area of software/controls, the work is being done currently at five sites; Tucson, Arizona; Victoria, Canada; Edinburgh, Scotland; Chilton, England; and Cambridge, England. In addition the software/controls for the Scientific Instruments will be done at other, yet to be determined sites.

WHY DO THIS

A valid question is, "Why not just form a central project team and do the software in a conventional fashion at a single site?". At first look this would seem to be sensible way to proceed and follows conventional wisdom.

Just as the telescopes are designed to exploit the best that the sites can offer the project structure is designed to exploit the best skills that the individual partners can bring to bear on the challenges represented by Gemini. The Astronomy community has a number of individuals with unique skills, expertise, and experience. The project benefits immensely from being able to tap these resources - indeed, access to these resources is required in order to build a telescope which satisfies the specifications, budget, and schedule requirements of Gemini.

The focus of the Gemini Project will migrate from Tucson (1992-1997), to Hawaii (1998-2000), and then to Chile (1999-2001) as will the project staff. Also, Gemini is a fixed length project with no guaranteed project positions past the year 2001.

Under these conditions it would be extremely difficult to convince a large number of these highly skilled individuals in the partner countries to give up their current positions to move to the Gemini Project. Indeed, it may be better for the long term health of the community if these people remain with their home institutions.

The Gemini solution to this in the area of software/controls was to form a distributed group. In order to make this successful it was necessary for Gemini to evolve a methodology for how to do this.

THE GEMINI DEVELOPMENT METHODOLOGY

Work Breakdown Structure

Gemini has followed two guiding principles, not always rigorously, in how the system was broken down into its component subsystems:

- try and partition the system to minimize the subsystem interfaces, and
- try to locate the software/controls at the same site which is building the mechanism.

The partitioning of the subsystems is complicated by the fact that it is a tightly coupled system. Unlike more conventional telescopes everything, including the orientation of the wind flow relative to the telescope, can affect the delivered image quality. This means that there exists a large degree of interconnection between the subsystems.

Locating the software/controls work at the mechanism fabrication site is not possible for some of the items such as the telescope structure and enclosure. In general the vendors with the skills to do this type of fabrication do not have ready access to the skills to do the software/controls.

TABLE 2.

Work Package Title	Organization	Site
Observatory Control	NOAO/Gemini	Tucson
Data Handling	DAO	Victoria
Communications	CONICYT + JACH	Chile + Hawaii
Core Instrument Controller	ROE	Edinburgh
Standard Control System	RGO	Cambridge
Telescope Control	RGO / DRAL	Cambridge + Oxford
Mount Control	RGO	Cambridge
Primary Mirror Control	RGO	Cambridge
Secondary Mirror Control	ROE	Edinburgh
Enclosure Control	DAO	Victoria

Engineering Practices

Gemini decided early on to follow three guiding principles:

- People who write software for big computer projects like to describe themselves as engineers. To deserve the name, they must behave as other engineers do.
- Successful software project require two things:
 - customers who can explain what sort of job needs doing
 - engineers who can deliver a system that will do the job on time and at a price that makes doing the job worthwhile
- When buying technology, it always pays to be second.

Gemini has adopted parts of the IEEE and ESA standards covering software development. These are used as guidelines for software developers following the Gemini Software Development model.

Gemini depends on a number of key elements in its model in order to keep a distributed group going. These are:

- Software Management Plan
- Overall System Design
- Formal Development Process
- Formal Review Process
- Established Standards

Each of these will be discussed in detail below.

Software Management Plan

An additional key item to distributed management is an overall plan. The project has found that the IEEE Standard 1058.1, *Software Project Management Plans*, is a useful mechanism for putting together this plan.

Overall System Design

The development of the Overall System Design was carried out by the Gemini Project Office using engineers drawn from the partner countries. This design went through a formal review process consisting of system design, preliminary design, and critical design where the review committee was made up of members of the partner countries and other large telescope projects.

The result of the System Design process was a formal Software Description Document¹ (SDD) which laid out the conceptual design of all of the subsystems as well as Interface Control Documents for the inter-subsystem and intra-subsystem interfaces in common. The SDD followed appropriate IEEE and ESA standards.

Formal Development Process

A Work Package is defined by a formal Work Scope document which is signed by all parties concerned. This Work Scope describes the details of each phase of the development process, the deliverables from each phase, and the details of budget, schedule, and payment.

The individual phases and their deliverables are detailed below. The first phase, Work Package Development, takes place as part of the development of the Work Scope.

Work Package Development

The output of this phase of the development consists of a Work Scope which contains the following deliverables:

- Work Breakdown Structure
- Schedule
- Cost Breakdown by WBS element
- Capital Costs
- Travel
- Key and Supporting Personnel

System Design

The output of this phase consists of the requirements and a conceptual design of the system.

- Package Requirements Specification.
- Environmental and Behavioral Models.
- Plans for Design Process
- Plans for Simulations / Prototypes / Trade Studies
- Documentation Plan
- Proposed Equipment

Preliminary Design

The output of this phase is a

- Preliminary Package Software Design Description
- Results of Simulations / Prototypes / Trade Studies
- Initial Work Product documents

Prototype

For most of the software packages we do not produce a formal critical design. A critical design review would review all of the software design in advance of coding - the intent being to avoid significant rework if the design does not meet the requirements. Our belief is that the only reason to do significant levels of detailed software design is if the cost of producing the software, and hence the cost of redoing the software, is significant relative to the cost of doing the detailed design. With the tools which Gemini uses for development we believe that a series of rapid prototypes is a much better investment than reviewing a formal detailed design.

- Prototype Control System
- Package Software Design Description
- Update of Simulations / Prototypes / Trade Studies
- Preliminary Work Product Documents

Alpha Version

Alpha Control System

- provide the user and programmatic interface to the work package as seen by higher level users and systems
- respond correctly to these higher level inputs
- respond correctly to peer systems (such as the Data Handling, Instrument Control, and Interlock system).

Preliminary Package Test Procedure

Beta Version

Beta Control System

- no Catastrophic Bugs
- may contain Severe Bugs with or without Work Arounds.
- may contain Cosmetic Bugs

Package Test Procedure

Preliminary Acceptance Test Plan

Final Version

Acceptance Test Plan

Final Control System

- no Catastrophic Bugs
- may contain Severe Bugs if Work Arounds exist.
- no Cosmetic Bugs

Acceptance Testing

Acceptance Test Report

Specification Control System

- hardware and software
- documentation

Review Process

Each of the major phases of the Work Package have an associated review (Table 3). The goal of the reviews is not to have a large amount of presentation material and spend long periods of time listening to presenters. The current review format depends on whether it is a *real* or *virtual* review committee. The difference is that *real* review committees meet in person a single time with the person responsible for the work being reviewed while only a subset of a *virtual* committee ever meets in person.

TABLE 3.

Major Phase	Associated Design Review
System Design	System Design Review
Preliminary Design	Preliminary Design Review
Prototype / Critical Design	Prototype / Critical Design Review
Alpha Version	Alpha Review

TABLE 3.

Beta Version	Beta Review
Final Version	Acceptance Specification Review
Acceptance Testing	Acceptance Testing Review

The process that virtual review committees follow is:

- receive the documents ahead of time via email or ftp,
- send in comments ahead of time via email,
- reviewee prepares responses to comments
- subset of committee may/may not meet with reviewee to go over responses
- comments/responses sent via email to committee for comment
- final report issued to all concerned

Real review committees are very valuable during the early phases of a work package, especially for areas which have poorly defined requirements, difficult interfaces, or demanding functionality. *Virtual* review committees make much better use of everyone's time for areas which are well defined and reasonably non-controversial.

Software and Hardware Standards

Gemini has followed a policy of adopting or adapting to commercial and community standards wherever possible. The philosophy behind this is to spend the time building the application, not the infrastructure for the application. Towards this end the following are considered Gemini standards.

TABLE 4.

Area	Standard
Host Workstation	Sun
Host OS	Solaris
GUI Builder	not decided
Windows	X11
Command Language	TCL
GUI	Tk
Visualization	PvWave
Bulk Data transfer	Self Defining Structures (SDS)
Observing Database	not decided
Archival Database	Sybase
Data Processing	IDL
Data Reduction	IRAF
Version Control	CVS
Trouble Reporting	TkGnats
Real Time Target	68040
Real Time Target OS	VxWorks
Real Time Database	EPICS
RT Database Programming	CapFast

One of the truisms of software development is that it is often the infrastructure rather than the application which is the more interesting project. In a number of areas Gemini encountered substantial resistance to the adoption of standards, especially when the standards were used to displace existing infrastructure developed by that site.

Gemini's position is that it is only by adopting standards that Gemini can stay ahead of the user driven application demand - after all, users want applications. Users also want applications that work the way their other applications work, and that have the same bells and whistles. By leveraging our applications off of widely used infrastructure standards Gemini can take advantage of the ongoing development efforts as well as the large programming pool available.

MANAGEMENT TECHNIQUES AND TOOLS

Work Scopes

Once the overall System Design and the Standards are in place it is necessary to have an agreement in place about what

work needs to be done, how that work will be done and reviewed, and how much effort and money is to be used to perform the work. In Gemini's system all of the Partner Countries have agreements in place which cover most of the boiler plate of a more traditional contract. The details of each Work Package are contained in a Work Scope developed by the staff responsible for performing the work.

The management of a work package is via a matrix management approach. The individual at the site who is responsible for the work remains an employee of his/her existing organization and reports to and takes direction from his/her normal supervisor. However, in all areas to do with the work package, the individual is responsible to the Gemini Controls manager. This means in practice that no changes to the work package which affect specification, budget, or schedule can be taken without prior agreement of Gemini .

National Project Offices

Each of the Partners have a National Gemini Project Office which is generally staffed with a Project Manager, Project Engineer, and Project Scientist. The Project Offices perform a very valuable function of being the major point of contact between the national communities and the Gemini Project. They also act to form a national consensus on issues that will be discussed by Gemini.

The National Project Manager is one of the signatories, in addition to the Work Package Manager, who signs the Work Scope.

LESSONS LEARNED

A number of lessons have been learned over the preceding three years and there will certainly be new ones. Some of the more important ones are:

Ownership

It is extremely important that the groups responsible for the work have a real sense of ownership of the work they are doing. This can only be developed by involving the groups in as much of the process as is possible. In the Gemini case this required starting a System Design phase with members of the partner countries involved, having members of the partner countries sit on the review committees, and having the work package responsables develop the work breakdown structure, schedule and costing on their own.

Personal Contact

It is very much easier to do productive work with people whom you know well on a personal as well as professional level. It is imperative, especially during the startup of the project, to have regular face to face meetings. Gemini had someone from the controls group at the remote sites for one week per month for the initial year.

Delegation

You have to let go - once you involve outside groups you have to be willing to give them the freedom to make design decisions and choices. This means that the system you are building will not be done the way you would have done it. If you manage carefully enough though, it will probably be done better than you were capable of on your own.

Moving Target

The requirements have to change - although it would be easy to insist on having all the requirements iron clad at the start this is unrealistic. To be successful the project must meet the user's expectations, of which the requirements are only an attempt at expressing. The waterfall model of software/control development is not appropriate in this environment.

Communications

The most important aspect of keeping a distributed group on schedule would appear to be good communications. Gemini uses a mix of the following communication methods.

TABLE 5.

Method	Pros	Cons
email	effective for frequent in writing instructions and communications	not always read, still gets lost or delayed more than would like real problem with standards for attaching binary files between Windows and Unix
email exploders only way to make sure that everyone sees everything	very good for making sure no one is left out of the loop on key topics	volume of traffic means some people will arbitrarily delete email if title has no hook
site visits	only way to make personal contact must precede all other forms of communication required for testing and some reviews	very expensive (\$500/day) for foreign sites hard on schedules, staff and families
ftp mirrors	keep all documents at all sites	file naming conventions differ between Windows and Unix
teleconference	good for weekly meetings	quality of voice and audio levels make multiple person teleconferences difficult end point equipment expensive
video conference	very good for small meetings with good agenda allows access to body language	compression/decompression delays in audio difficult some of partners have very expensive charges still not quite "plug and play"
fax	sometimes only way to get document there you know it arrived	quality / missing pages differences between letter and A4 sizes
FedEx/UPS	still are people with no net connection	expensive
Postal Mail	still best way to guarantee that it was received	slow and expensive for large documents

THINGS TO CHANGE

The one area which the author would change significantly in a similar project would be to involve more of the partner countries principal software/controls staff in a central team for the overall System Design phase. The best option would be to get one or two staff from each of the institutions to spend a period of 6 months at a central site to do the system design as a coordinated team. This would have made things much easier at the point in time at which the subsystems are sent out as work packages as there would already be staff on-site with in-depth experience with the design. Although this might seem to contradict the usefulness of distributed management there is a learning curve associated with bringing the group up to speed on a design - perhaps improvements in video conferencing will remove this problem in the future.

In Gemini's case any such effort would have been difficult due to the span of time over which the partners joined the project - some 3 years. This meant that it was difficult to get commitments from all the potential partners to send staff to the central site.

Although it might seem easy to insist on all the partners being signed up before starting the requirements process, and then

to have all the requirements agreed to before proceeding to do a system design, this is not the way projects like this happen. It is necessary that large projects proceed as much in parallel as possible in order to shorten the time scale to delivery. I believe that this is just an extension of the concept rapid prototyping to the level of including all the phases of a large project.

CONCLUSIONS

The Gemini Project's Software Development Methodology effectively meets the challenges posed by Gemini's distributed nature. Although this methodology is more challenging and more expensive than centralized management this does not necessarily translate into more difficulty and more expense for the project as a whole. Indeed, it is the author's experience that distributed management allows one to access much more qualified people to work on the project than a hiring process would result in. This is not to say that the project cannot recruit and hire highly skilled staff - just that the pool is much larger if one is willing to do it in a distributed fashion. This results in cost savings, schedule shortening, and improved system performance relative to a centralized staff - however such improvements are difficult to quantify up front. Our current metrics show that staff, in general, are meeting their milestones with only 70-80% of the effort they estimated.

A methodology makes it possible to successfully manage a large, distributed software/controls development effort - although Gemini has a few more years to go before we can lay claim to ultimate success. However the Gemini software/controls task is currently on budget, to schedule, and has yet to miss a specification.

A distributed project is the only way large science projects will be done in the future so we need to find methodologies that work and not be afraid to try new models. The advent of high bandwidth communications at reasonable rates will bring more and more of these types of group interactions into the forefront. The ability to quickly form and disperse inter-organizational project teams in a distributed fashion will be key to keeping costs and schedules in check while at the same time meeting the challenging specifications with low risk solutions.

ACKNOWLEDGMENTS

The author would like to thank all of the Gemini work package responsables for their effort and forbearance as Gemini worked towards a model which would work.

REFERENCES

- [1] S.Wampler, K.Gillies, S.Beard, A.Johnson, R.Laing, C.Mayer, P.McGehee, R.McGonegal, P.Wallace, Software Design Description, Gemini Project Doc. No. SPE-C-G0037

The MECCA Source Code Capture Utility

Alex M. Waller
Fermilab
Accelerator Division Controls Department
MS 347
P.O. Box 500
Batavia, IL. 60510 USA

HISTORICAL BACKGROUND

In the early 1980s the Accelerator Division Control System underwent a major upgrade[1]. Until that time almost all the software for the control system was written by the controls group staff. This was largely because the means by which programs were entered into the control system were not readily accessible to most people outside the controls department. Programs previously were entered through magnetic tape and punched cards. Much care needed to be taken in entering an application in this fashion, as even editing of source was done with commands on punched cards and source on magnetic tape. This mode of operation was to be eliminated as the controls development environment became interactive. The move was from a system with a batch monitor operating system to one with a multitasking operating system that allowed multiple users access through CRT terminals. It was anticipated that with such an environment non-controls programmers would be writing applications for the newer Accelerator Division control system.

Until that point in time in the early 1980s, there was little need to keep track of application source code for the various programs that ran the accelerator. The source was all vaulted in rows of tape racks and trays of punched cards. Now source would be edited interactively and exist dispersed on rotating magnetic disk storage. It became desirable to archive all the vital code for the accelerator in some centralized fashion rather than having code reside within the private accounts and disk allocations of the various users.

There was yet another strong argument to keep all the source code centrally located. Since the control system was undergoing conversion and expansion both in hardware and software, it was necessary to be able recompile and relink all applications under certain circumstances. This could only be successfully accomplished if the source code was all centrally located.

As a result of the concern for source code management with the arrival of the newer control system, an in-house code capture system called APL was written[2]. It was based on the observation that most applications were not written by teams of programmers but rather by one or possibly two collaborating individuals. To this day this is largely true of all applications that run the accelerator. The utility simply copied all source code from a user's directory to an APL directory. Compiling and linking were automated but the necessary commands still were needed to be composed by the person writing the application.

The source code capture utility was very successful but had some limitations that became more annoying with time. Initially it was written in the VAX VMS script language DCL[3]. Since the commands were interpreted, APL would run rather slowly. Users could not write their own library procedures and hence a wealth of useful procedures could not be shared by all. The source code capture utility only applied to console applications. All task building, linking and overlaying instructions had to be defined by the user. For a large application this last point took quite a bit of expertise.

In the late 1980s an APLII was created to address some of the growing problems. The actual code capture utility was written in FORTRAN rather than DCL to speed up execution time. This was also the first try at integrating a commercial product in helping manage the source code. The VMS product CMS (Code Management System) was used[4]. Individual modules within the CMS database were compared and source code differences were logged in the CMS database. This allowed the reconstruction of any previous version of code. Unfortunately the price paid for this flexibility was execution speed. APLII never went far beyond beta testing because source code updating was too slow for users[5].

THE NEXT GENERATION

History and experience were the teachers for a new generation of source code capture systems. It was noted that archiving all sources associated with an application was adequate throughout the lifetime of the earlier source code capture systems. Include files from various system resources needed to be allowed while still other include files (other than those of the programmer) needed to be disallowed to contain applications within a scope of available, reconstructable code. Also, cataloging the owner of an application and maintaining a description of it proved invaluable to operations.

As more users programmed over a period of time, useful features became obvious. The next generation source code capture system had to address the current one's limitations. User libraries needed to be allowed and captured. These libraries had to be available for all other users also. Programming tasks other than just console applications should also have their source code captured. The user had to be freed from having to specify the program compilation, include file dependencies and module linking order. This process had to be fully automated so that the task building process was very simple as far as the user was concerned.

What was also desired was more functionality and flexibility for the source code capture utility. The utility should allow program development to go on within a user's default directory but without always archiving the modified source. Also the source capture utility itself should be designed in such a way as to be able to extend its functionality without having to recompile, link and install a new image thus disrupting user work every time. Modification or fixes "on the fly" to the source code capture utility was a desirable maintenance feature.

THE FEATURES OF MECCA

All the above mentioned desires were implemented into the new source code capture utility called MECCA (Management Environment for Controls Console Applications)[6]. MECCA did not stop with archiving only console applications, although this was its initial goal. User libraries and service applications also are captured. A service application makes use of accelerator console libraries but typically does not do any console I/O. These applications are typically servers or monitor applications.

MECCA incrementally builds applications by using the VMS commercial product MMS (Module Management System)[7]. This approach rebuilds an application project much more quickly. An entire application can be rebuilt if necessary by specifying a parameter to MECCA. Also one can retreat back to a previous version if required.

The application project directory can be listed by another MECCA parameter. Extra diagnostics can be turned on and inserted into the resulting log file of the application build. These diagnostics have proved invaluable in tracking down MECCA and user problems. Users can "announce" changes in their application through a mail list that is built from other users who "subscribe" to any number of such MECCA announcements for console applications and service applications.

Users can build VMS help files for the console or service application as well as libraries. Any user can get help on any application as needed. Author and history information are available on each application. If the maintainer of an application should change, modifications can be made to the author information that is kept in MECCA.

HOW MECCA WORKS

Having dispensed with the historical background that led to the development of MECCA and expounding on the features that were attained with it, a description of the workings of this program is in order. A conceptual diagram of MECCA appears in Figure 1. A more detailed and philosophical account appears in the final section of this paper.

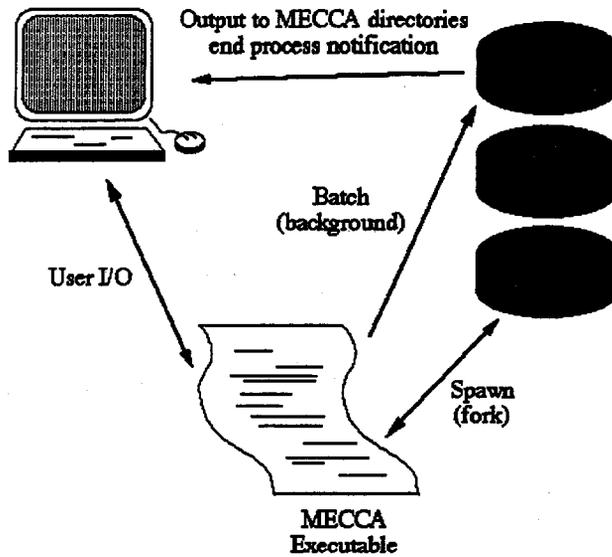


Figure 1
Conceptual Flow of MECCA Control

Any source code modification process begins when a user "checks out" the MECCA source of an application or library or service by performing a MECCA COPY command. All pertinent files for the application are copied into the user's default directory. Since MECCA is running under a VMS environment, the command would appear as a VMS command. Thus the syntax would be: MECCA/COPY nnnnnnnn; where nnnnnnnn is the program, library or service name. All other MECCA commands take a similar format. Remember from the historical note that individual applications within the Accelerator Division controls department are mostly developed by one or two people. A more elaborate check-out mechanism beyond the simple source copy is not necessary.

The next step is to build the application (after the appropriate editing has been performed). The form of this command is generally: MECCA nnnnnnnn; again where nnnnnnnn is the application, library, or service name. The user generally needs not specify more VMS-like parameters unless (s)he wishes to do something special. For example, one may wish to force a recompile of all the source with the /COMPILE_ALL parameter. One may wish to have extra diagnostics placed within his/her log file with the /DIAGNOSTICS parameter.

A simple file date compare is performed on the current MECCA modules and the modules located within the user's default directory. New and modified files have their include files scan to check for any include files that are not allowed and to build up a list of dependencies for each new or modified module. The dependencies that are found are used to automatically generate an MMS (make) file for the application.

While MECCA is moving files into the MECCA directories and the source is being compiled and the image linked, a simple lock mechanism prevents others from trying to perform any MECCA operations on this application. A simple lock is created with a file. This mechanism is sufficient to prevent conflicts during this critical section within the MECCA operation on an application.

At this point MECCA copies all current working source from a user's default directory to an appropriate sub-directory located within the MECCA root directory. If this is a development phase (specified with a /DEVELOPMENT parameter) then no MECCA source will be copied to the appropriate directories within MECCA. The build operation is then started as a batch (background) process so the user may go on to do other things while the application, library, or service is being built. Within the batch process, it is determined if this is a console application or service or a library. If this is an application a linking process is involved. If this is a library the necessary library module is created. A log file, possibly with diagnostics, is produced by the batch process and is placed in the user directory where the initial MECCA operation began.

If no errors have occurred, the old source code maintained within the MECCA directories is moved to a directory with the text OLDn appended to it and the successful build is left in the current MECCA directory for the application. The n in the OLD text is incremented for each revision number. If there is an error, the old source maintained by MECCA is kept within the current MECCA directory for the application, and the version with errors is kept in a directory named TEMP for the user to peruse. Figure 2 details the MECCA directory structure. At the end of the MECCA batch process the user is informed of the state of the build operation.

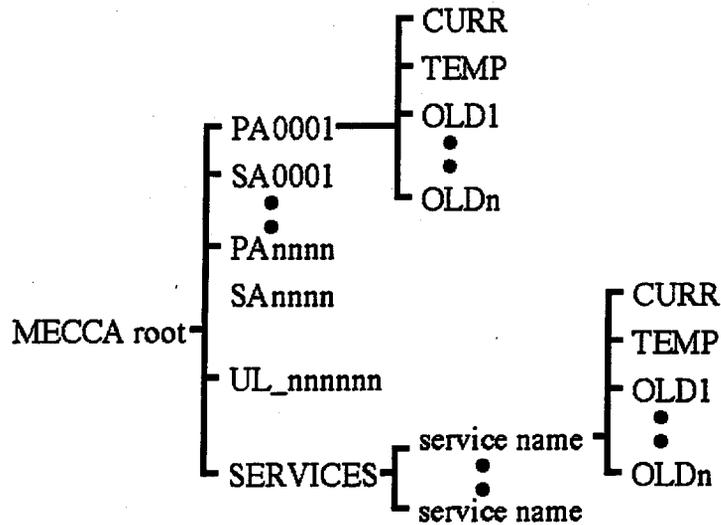


Figure 2
MECCA Directory Structure

THE MECCA PHILOSOPHY

User interface designed to be simple

Since the possibility of using character cell terminals existed, the MECCA interface was kept to be textual and command driven. Also, on a careful analysis of what was required, MECCA was better suited to be command driven. There is very little involved that would require a more sophisticated graphical user interface for selecting input options, especially if the default options were chosen wisely. As mentioned earlier, MECCA defaults are such that what the user would "normally" want to do requires no extra MECCA parameters to be specified.

While the user is within the MECCA monitor process very little user feed-back (prompting) is required. For entering a new library or application one needs to enter a line of text giving a description of the library or application. If one is retreating to an older version MECCA needs a version number to retreat to. If previous source code for an application was captured from a default directory different from the current default directory or modified by a user other than the current user this information is displayed and one is prompted to continue or abort. Finally, if all goes well, the user is asked if (s)he wishes to submit the application for the batch build operation.

Monitor process co-ordinates flow of control

Command input parameters are processed by a separate module of the MECCA monitor. This allows an easier port to other platforms where a non-VMS style of command parameters would be more acceptable. As mentioned in the previous section, the monitor will solicit any necessary user input. For new libraries and applications the monitor creates the necessary directories from the MECCA root directory; for console applications the directory name is derived from a sequentially assigned number; for libraries the directory name is prefixed with the string "UL_"; for service applications the service name is used directly.

The monitor process is responsible for spawning (forking) processes. Some of the processes that get spawned are: Help, which executes the VMS DCL HELP command; Directory, which executes the VMS DCL DIRECTORY command; Copy, which executes the COPY command; and HISTORY, which executes a VMS TPU[8] section which evokes an editor to browse the history file of a given application. When a process is spawned control returns to the MECCA monitor process. A spawned process is generally one where a single VMS command line can be executed, a brief VMS DCL script can be executed, or a brief piece of auxiliary code such as C or TPU can be executed.

The monitor process is also responsible for submitting batch jobs (background tasks). The background jobs that get submitted are: Build, which is the primary functionality of MECCA; Help, which builds the necessary VMS help from the extracted comments within a source program; Retreat, which retreats to a previous version of an application in MECCA. A Batch process is almost always the end result of a MECCA operation. Batch processes are tasks that are lengthy, complicated, or time consuming.

Auxiliary operations are kept separate from monitor process

Since many MECCA tasks are relegated to spawned or batch jobs, MECCA is highly modular. Only new functionality requires adding information to the monitor process so that the monitor can co-ordinate the execution of a new task. MECCA behavior can be altered or extended without compiling, linking, or installing an executable image with every instance of required change. This modular approach allows a tight granularity on maintenance. Generally only a small section of code or script has to be modified or debugged and fixed since all functionality that needs to be changed (or fixed) is entirely self-contained within a single module or script. Also, new functionality that is added can be developed and tested independently from the current running version of MECCA.

The many MECCA modules are written in a variety of ways. What is used largely depends on the situation. The MMS (make) generator is written in a powerful string processing language called TPU on our VMS system. This was most appropriate since much string processing is needed. The included scanner is written in C. Many of the other procedures are DCL scripts which take advantage of the command facilities of the VAX VMS operating system.

CONCLUSIONS

MECCA has adhered to the simplicity of earlier source code capture systems but yet was able to achieve the flexibility necessary for the growing needs of the programming community of console applications and services. It has proved fast and efficient for the given environment of the Accelerator Division control system. The decision to keep MECCA code itself modular has greatly aided in debugging and developing MECCA rapidly thus keeping maintenance controlled and to a minimum.

ACKNOWLEDGMENT

I would like to thank Jim Smedinghoff for his valuable suggestions on the general design direction MECCA should follow. Jim also provided the TPU procedure that generates the make (MMS) file for any given application, library or service.

REFERENCES

- [1] D. Bogert, The Fermilab Accelerator Control System, Nuclear Instruments and Methods in Physics Research, Volume A247 (1986), pp. 8-24.
- [2] A. Thomas, D. Baddorf, K. Cahill, D. Rohde, J. Smedinghoff, L. Winterowd, User's Guide to ACNET Console Systems, Fermilab Internal Report, Software Documentation Memo 62.3 (1985), Chapter 6.
- [3] OpenVMS DCL Dictionary, Digital Equipment Corporation, 1995.
- [4] DECset, DEC Code Management System Reference Manual, Digital Equipment Corporation, 1995.
- [5] Private communications with Robert Joshel and Brian Hendricks.
- [6] A. Waller, MECCA; the Management Environment for Controls Console Applications, Fermilab Internal Report, Software Documentation Memo 208 (1994).
- [7] DEC3GL Implementation Toolkit for VMS, MMS Description Generator User's Guide, Digital Equipment Corporation, 1992.
- [8] DECTPU, DEC Text Processing Utility Reference Manual, Digital Equipment Corporation, 1993.

Writing Easily Portable Code

by *M.D.Geib Vista Control Systems, Inc.*

Introduction

This paper discusses some of the issues related to producing portable software.

When a software system is ported from one platform to another the following logical steps are normally taken: the source files for the system are moved or made accessible on the new target system; any platform dependent modules are modified to support the new target platform; any required data files are moved to the new target system; and finally the complete system is built and tested on the target. Portable software makes this process as easy and reliable as possible.

Some of the important issues that can affect portability include the differences between the compilers being used on the different platforms, how to interface to required OS facilities, byte ordering, floating-point format, availability of standard APIs and facilities, and possibly the transfer of data between supported platforms.

This paper discusses these issues as they relate to source code and data portability. The paper is the result of experience gained during a project at Vista Control Systems, Inc. to re-engineer Vaccess. Vaccess is a real-time database with an API that transparently supports network access to remotely hosted databases.

Compilers

Compilers and languages in general are important issues with regard to developing portable systems. It is advantageous to choose a language that is available and mature on all the target platforms and in addition one that is covered by an adopted standard definition. There are a number of languages with adopted standards, such as FORTRAN and Pascal. However, languages such as FORTRAN are plagued by the availability of numerous extensions that make each implementation a non-portable super-set of the language. The FORTRAN-90 language holds the promise of being suitable as a portable language. However, FORTRAN-90 does not yet enjoy widespread support. Languages that do not provide a 'complete' set of facilities and runtime functions require extensions which are usually not consistent between vendors.

The popularity of C reflects its rich set of facilities, useful data types, operators, control structures, and runtime library functions. In addition to the characteristics of the language, C has another advantage over other languages, not specific to portability: There are a very large number of C programmers.

C is not free from many of the problems discussed above. Depending on the compilers involved, there can be a wide range in how consistent the different features of the language are implemented. Many older implementations include constructs and features not supported by, and incompatible with, the newer implementations. ISO compliant C compilers minimize the differences between implementations to a point where they are all but gone. This paper assumes the use of ISO C as the language used to develop portable systems.

What is not defined by ISO

Even if the compiler is ISO compliant, problems can still arise. The ISO standard leaves certain details up to the compiler implementers and these details can affect portability. For example, the ISO standard does not specify the default data type of a variable defined as *char*. Whether the variable is signed or unsigned is left up to the implementer.

In a similar way, some common C programming practices are specified as undefined in the ISO standard. For example, conversion between a function pointer type and a data pointer type is undefined, and the ISO

standard does not guarantee that a function pointer can be safely converted to or from a type (*void **). Code which includes such conversions most likely will not be portable.

Extensions provided by a compiler are also problem areas when it comes to portability. Be aware that some popular compilers may provide extensions to the ISO standard that may not be supported on all the target platforms. If a compiler supports a switch to disable ISO extensions or nonstandard constructs, the compiler can be used to help to find these problems early on in development, or when planning to port an existing system.

The size and range of data types is not completely specified by the ISO C standard. The standard simply states that an *int* can not be smaller than a *short* and that a *long* may not be smaller than an *int*. Similarly a *double* can not be smaller than a *float* and a *long double* may not be smaller than a *double*. The ISO standard does specify the minimum range for each type. For example, the legal values of an *int* fall in the range of -32,767 to 32,767. This requirement means that an ISO conforming compiler can not represent an *int* in only 8 bits. A compiler implementer is free to increase the range of a data type. If a data item must be of a known size, then by using a user-defined data type or new symbol that type can be supported on platforms that may have different sizes and ranges for the built-in C types.

User defined data types

Some platforms and compilers support data types not supported on other platforms and some are not fully specified in the ISO standard. Both of these problems can be handled with user defined data types, or simply new symbols defined for each type a system will need. Along with the user defined data types, it is important to keep in mind how these data types will be used and manipulated. If a data type may not be supported on a target, the chances are good that the normal C operators and runtime functions will not support the data type. In these cases a new function or macro should be defined and used to manipulate the data type. Configure the compiler to do as much argument checking as possible. This helps to reduce the possible incompatible use of user defined data types.

Platform independence

It is best when developing portable systems to isolate all the platform dependent code in separate modules. These modules can then be recoded on each target platform. This method is preferred over that using conditional compiles for a given section of code. Using separate compilation modules for isolating platform dependent code explicitly identifies future porting work. When maintaining the code, it is much less error prone to work on the platform dependent modules when a change is necessary, rather than working on a section of code in a platform independent module that could have side effects on other platforms the code runs on. Isolating platform dependent code also helps keeps the platform independent code much more portable when a new target is selected, since only the platform dependent modules need to be worked on. Platform dependent code includes any code that assumes the representation of data types, makes use of OS provided services, etc.

OS facilities

Any facility not part of the ISO standard for the C language must be isolated in platform dependent modules. Even though the newly adopted POSIX standard has tried to address some of the areas not included in the language specification, its support is not widespread enough to make it dependable. However, the use of POSIX can make the platform dependent modules much easier to port, so its use can still be beneficial. If more than one of the target platforms supports POSIX, porting the platform dependent modules could be trivial. They could be equivalent, thus reducing the work required to support a given platform.

Bi-endian support

When developing a portable system, it is impossible to ignore the issue of byte ordering. Intel and Digital both support little endian byte ordering while Motorola, SUN, and HP support big endian byte ordering. Some new processors can run in either or mixed modes. Any code that assumes the byte ordering of data is dependent on the platform and should be isolated in platform dependent modules.