THE BATES PULSE STRETCHER RING CONTROL SYSTEM DESIGN

Tomas Russ, Anthony Carter, Zdenek Radouch, Coles Sibley
MIT-Bates Linear Accelerator Center
P.O. Box 846, Middleton, MA 01949

Abstract

The design of the control system for MIT's pulse stretcher ring is described. Heavy use is made of standard hardware and software elements. Particular emphasis is given to the design of a true distributed system, one that can expand both at the data acquisition/control and data processing end. For this purpose, a method is proposed to avoid increasing the traffic on the network when adding more computers. Issues related to real time response and user interface are also treated. Finally, measurements up to date are summarized.

Ring Parameters

MIT's Bates Accelerator Center has been operating a medium energy electron linear accelerator since 1972. The original maximum energy (480 MeV) was doubled with the addition of a recirculator in 1981. An RF upgrade program lifted the maximum available energy to the 1GeV mark in 1986. However, the maximum duty cycle (1%) remained a constraint for low data rate (coincidence) experiments. In 1987, DOE authorized the construction of a pulse stretcher ring, which is due to begin operations by 1991[1]. It will extend the beam duty factor to 85%, with a maximum extracted current of 50 μA. The design will also allow for internal target experiments, in which the ring operates in storage mode. The maximum circulating current in this mode will be 80 ma. Fig. 1 shows a layout of the ring, with the associated injection and extraction lines. Some of the parameters useful for understanding the magnitude of the control system are presented in Table 1.

Control System Goals

Ring operation will rely heavily on on-line modeling and correction, which puts a premium on the availability of beam instrumentation, particularly BPMs,current monitors and profile monitors. Enough computational and data acquisition power must be provided to allow for tune and misalignment measurements and calculations. Manual corrections, of the type provided by control panels, will also be used frequently, particularly in the first stages of ring commissioning.

Element	Inj	Ring	Extr
Magnets	29	203	48
P. Supplies	29	142	54
BPMs	5	40	5
RF systems	1	1	
Curr. monit.	2	40	1
Slits	1	2	1
Profile mon.	3	3	3
Synchr mon.		4	1

Table 1. Pulse Stretcher Ring Inventory.

Individual and ganged controls, data presentation in different physical units, and adequate response time in terms of "operator feel" are some of the important elements in the design of the user interface.

In addition, all remaining instrumentation, status and alarm signals, may want to be displayed permanently on adhoc monitors in the control room, or stored (logged) for later analysis and display.

Finally, we realize that not all the control system needs can be forecast at the time of the initial ring design. An important consideration is to allow for expansions both at the data acquistion and the data processing end.

Summarizing, the basic premises for the design of the control system are:

- Open ended architecture, both in hardware and software.
- Use of as many standard elements as possible.
- Provide response times for operator controls > 10 Hz.
- Provide the capability to update all the beam instrumentation at a rate > 3 Hz.
- Allow for several simultaneous displays and control panels.
- Allow for complete data logging and plotting.

Ring Control System Architecture

The control system functions can be clearly classified as either data acquistion/control (sometimes called front end functions, those that deal directly with the hardware), or data processing.

In trying to identify standard elements for these functions, it becomes apparent that workstations are the system of choice for data processing. The combination of great CPU power and

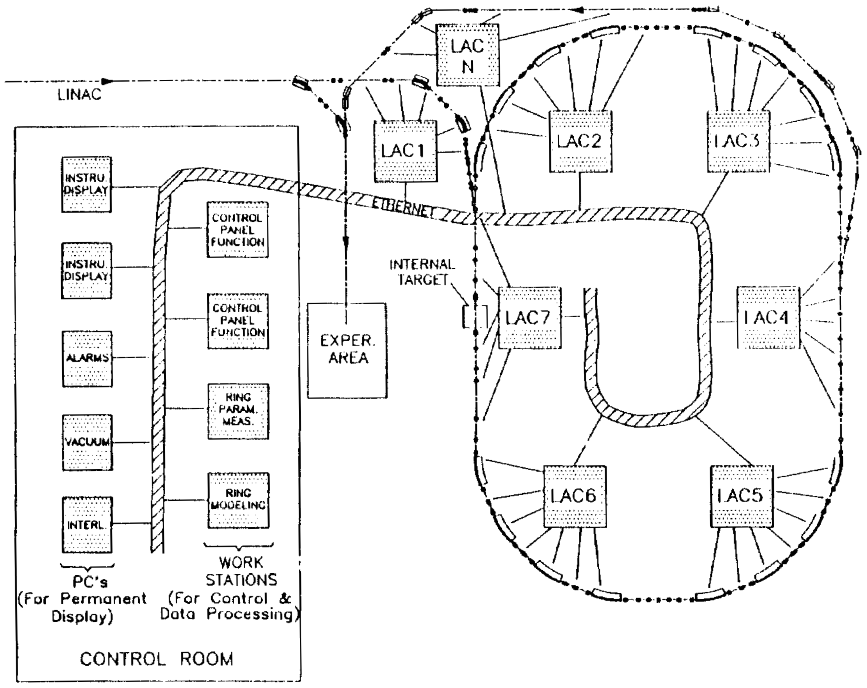

Fig. 1 Pulse Stretcher Ring Layout and Control System Architecture.

readily available graphics capability make them especially suitable for both control panel type functions and on-line modeling and correction. Several examples of the use of workstations for accelerator control systems have been described[2,3]. Even more, efforts are being made to develop standard software elements for control functions, promising to save many man-years of programming[4].

At the data acquisition end, the choice is not so clear cut. For one, most of the data acquisition/control is reduced to writing and reading from I/O ports. In those cases, very little is needed in terms of local CPU power. This function was traditionally performed with CAMAC modules, but cost and lack of integration with general purpose microcomputers are relegating this time honored standard, and replacing it with one version or another of industrial bus based computers. In our case, we found that even the low end of industrial buses, STD, would be suitable for this simple I/O function. In other cases, where local processing power is important (beam profile monitor data reduction, for example), VME or other 32 bit buses are a better choice. We designed the system so that more than one type of bus based computer could be used in the front end.

In terms of networking, there is little doubt that the only standard supported by all workstation manufacturers today is still Ethernet. As has been abundantly demonstrated[5], Ethernet is not the best candidate for real time control, because of its non-deterministic nature. However, we have taken steps to minimize the collision problem by adding a time multiplexing feature to the front end computers, as described below.

Fig. 1 shows one possible layout of the ring control system architecture. Local area computers (LACs), most of them STD bus based, connect to the hardware within a certain geographical area of the ring. Alternatively, LACs could be functionally —rather than geographically—- distributed (one LAC for BPMs, one or more for magnet control, etc.).

In the control room, general purpose workstations handle control panel, modeling, and parameter measurement functions. In addition, personal computers (PCs) are used as dedicated displays for beam and non-beam instrumentation and alarms.

As can be noted, the system is truly distributed, in that no computer is acting as a master nor handling all the network traffic. It is also open ended, in that other front end or data processing computers can be added without degrading the system (with limitations, of course).

Mechanisms for Control and Database Update

Operator or computer control of a given set of elements in the ring will originate in one of the workstations. A short packet will be sent to the corresponding LAC with the request. The LAC will execute it and return the status in a similar reply packet. This request/reply mechanism, because of its packet by packet acknowledgment, can be implemented at a very low level in the network software (typically link level access). This helps in maintaining a good response time, which has to be kept below 100 ms. to avoid "operator complaints".

More stringent traffic considerations apply to the acquisition of instrumentation data. As shown in Table 2., the network bandwidth usage is kept at a fairly low level, even if all the instrumentation database were updated at 4 Hz or more. However, this is true if only one copy of the database is sent through the network. This can be accomplished by using multicast addressing[6] and having the LACs place the instrumentation data periodically on the network, unsolicited. In this way, whenever a processing computer needs access to a certain data, it can connect to the corresponding multicast packets and retrieve the information. If a clever distribution of multicast addresses is assigned, processing computers would need to handle only the network traffic of interest for the currently running programs. This scheme can be viewed as keeping the instrumentation database on the network at all times, and overcomes the need to have duplicate copies of data for each computer that needs it.

Device	Qty.	Bytes	Updated:	% BW.
P.Supplies	225	16	250 ms.	1.152
BPMs(single)	50	8	250 ms.	.128
BPMs(plot)	8	500	250 ms.	1.280
Profile mon.	9	3000	3000 ms.	.720
Vacuum	64	8	250 ms.	.164
Synch.mon.	5	500	250 ms.	.800
Current mon.	43	4	250 ms.	.055
Subtotal				4.299
Overhead(20%)				.860
Total				5.159

Table 2. Traffic calculation for "database update", as a percent of total available Ethernet bandwidth (10 Mbit/s.).

Avoiding Collisions

Of course, even though the network utilization might be low (around 5% of the available bandwidth), collisions on the Ethernet can seriously impair the effective throughput. This is further aggravated by the synchronic nature of the data acquisition, which is triggered by the ring injection/extraction cycle.

One way of avoiding most of the collision problems would be to assign a time window to each LAC for multicast transmission of the instrumentation database. As shown in Fig. 2, a common trigger pulse would be applied to all LACs, at the "database update" rate. An internal timer within each LAC would define a different "time slot" for packet transmission, avoiding overlaps. Request/reply packets would remain asynchronous, but collisions would only be produced if more than one processing computer would originate them at the same instant. Because of the short nature of the packets, and the rather low bandwidth usage, this is a low probability event.

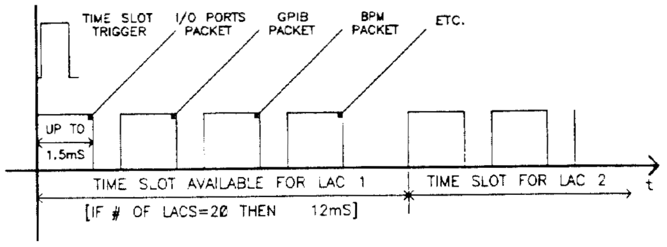

Fig. 2 Database Update. Every LAC multicasts all its instrumentation data (or the data that changed from last update) once every trigger pulse. A time slot is assigned to each LAC to avoid collisions on the Ethernet.

Workstation Software

As mentioned earlier, "standard" software for workstations is being produced at other laboratories. In particular, CEBAF[4] has come up with a set of graphics tools (LOGIC) to define the logical links between the parameters transferred from/to the front end computers and their presentation on the operator screen. Coupled with this, another set of tools (DISPLAY) enables the user to configure the workstation screen for a desired presentation and control of these parameters. This software is particularly useful for implementing control panels and alarm systems. Even though the original CEBAF architecture calls for CAMAC as the only allowable hardware interface, we have adapted it to access LACs, both at the request/reply and multicast levels. In this way, all the software effort done at the application level will be additive to that done at CEBAF and other places that are adopting it. We feel strongly about the need to combine efforts in this direction, saving this laboratory —and hopefully others— time and money.

LAC Architecture

The STD bus based Local Area Computer (LAC) is described in detail elsewhere[7]. We will refer here to aspects that relate to its hardware and software interface with the rest of the control system. Having stated in our goals the possibility of using other bus based architectures at the front end for CPU demanding tasks, we decided to incorporate the "time window" feature described above not in the LAC itself, but in it's Ethernet controller card. For that purpose, an inexpensive controller /transceiver was developed in-house[8], interfacing not with the the STD bus, but rather with the Small Computer Systems Interface (SCSI). We know of no bus based or workstation manufacturer to date that doesn't support the SCSI interface. Use of other LAC architectures are therefore guaranteed (Fig. 3).

Software for the LACs is written in C to run under VRTX, a commercial real time executive. All that is needed is a SCSI driver, a demultiplexer for the received packets, and the hardware interface applications, which run as separate tasks [7].

The two main applications are an I/O port and GPIB drivers. These will cover 90% of the control system needs. It has to be noted that every application will require different fields in their request/reply packets. This means that the processing computers have to generate (slightly) different packets depending on who they communicate with. This is a further incentive to keep the number of applications running in the front end to a minimum. Code is presently developed on a PC and downloaded to the LACs for debugging. Even though VRTX (as well as other similar products) is an optimum vehicle for real time software design, debugging tools are not very sophisticated, and can be time consuming. An alternate approach, given the simplicity of the software, would be to run the LAC under DOS. Consideration is being given to this option.

Measurements

We have made some measurements on a simulated environment consisting of a HP9000-330 workstation (2 MIPS) used in control panel applications running the CEBAF software, a PC/AT computer displaying BPM data (display computer), and one LAC controlling a power supply setup. Table 3. shows a summary of these measurements[9].

Workstation	
Request/reply	33 ms.
Multicast rate	> 50 Hz.
Logic cycle (simple)	5 ms.
Logic cycle (complex)	110 ms.
Disp. cycle (simple)	43 ms.
Disp. cycle (complex)	177 ms.
Display PC	
BPM Display (15)	> 4 Hz.
LAC	
SCSI data transfer	>1 MByte/s.
STD data transfer	>150 KB./s.
Jitter in time slot	<200 μs.

Table 3. Some Timing Measurements.

Two aspects which need more attention are the capability of workstations to handle multicast packets and the speed at which the CEBAF software can process the logic and display cycles, when the control panel complexity is raised. Earlier measurements using a Microvax II workstation[10] indicate that at least 200 multicast packets can be read per second before losing data. In our present HP workstation, this number seems to be reduced to around 50. We are not sure if this is related to the power of the workstation, or with more subtle aspects of the communication controllers.

The question of the update speed in the CEBAF software is surely related to CPU and graphics power. However, due to the "infinite loop" nature of the software, not much else can be done to improve the benchmarks except for "throwing CPU power" at the problem. One nice feature would be to port the display software to a non-proprietary standard, so one can take advantage of the competition existing nowadays in the UNIX workstation marketplace.

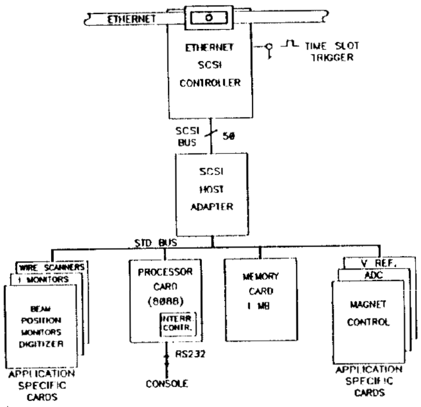

Fig. 3 The LAC architecture.

References

1. J.Flanz et al., "The MIT-Bates South Hall Ring", talk B5 in this conference.
2. V.Paxson, V.Jacobson, E.Theil, M.Lee, and S.Clearwater, "A Scientific Workstation Operator-Interface for Accelerator Control" Proc. of the 1987 IEEE Particle Accelerator Conference, 556–558.
3. A.Abola et al., "Experience in Using Workstations as Hosts in an Accelerator Control Environment.", Proc. of the 1987 IEEE Particle Accelerator Conference, 594–596.
4. R.Bork, C.Grubb, G.Lahti, E.Navarro and J.Sage, "CEBAF Control System.", in Proc. of the 1988 Linac Accelerator Conference, Williamsburg, VA.
5. see, for example, R.Parker and S.Shapiro, "Untangling Local Area Networks", Computer Design, Mar. 1983, 159—172.
6. Digital Equipment Corp., Intel Corp., and Xerox Corp., "The ETHERNET, A Local Area Network, Data Link Layer and Physical Layer Specifications. V2.0", 1984.
7. T.Russ, C.Sibley and Z.Radouch, "A Local Area Computer (LAC) for Control and Data Acquisition", contributed paper U29 in this conference.
8. T.Russ, "Ethernet Controller Adds Communications to SCSI Bus.", Electronic Design, **36**, 20, Sept.8, 1988, 91-96.
9. A.Carter "Performance Measurements of the CEBAF Software", Internal Bates Communication., Mar.13, 1989.
10. T.Russ, Z.Radouch and C.Sibley, "LAC report #3", Internal Bates Communication., Oct.28, 1988.