ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

The Computer Control System for the CESR B Factory

C. R. Strohman, D. H. Rice and S. B. Peck
Laboratory of Nuclear Studies
Cornell University
Ithaca, New York 14853*

Abstract

B factories present unique requirements for controls and instrumentation systems. High reliability is critical to achieving the integrated luminosity goals. The CESR-B upgrade at Cornell University will have a control system based on the architecture of the successful CESR control system, which uses a centralized database/message routing system in a multi-ported memory, and VAXstations for all high-level control functions. The implementation of this architecture will address the deficiencies in the current implementation while providing the required performance and reliability.

I. INTRODUCTION

CESR-B is an upgrade to the existing CESR facility.[1] The major part of the upgrade is the addition of a second storage ring in the existing tunnel. The two rings will operate with asymmetric energies (3.5 GeV and 8 GeV) and will intersect within the CLEO detector. The design luminosity is $3x10^{33}cm^{-2}s^{-1}$ which will be achieved with 230 bunches in each ring.

The control system for CESR-B is also an upgrade of the existing control system.[2] The architecture is shown in figure 1. The MPM (Multi-Port Memory) contains the database and is accessible by the high-level computers and the BCCs (Bus Control Computers). The high-level computers are used to develop and run programs which interface with the operators and physicists to control and monitor the experiment. The BCCs manipulate and move data between the database and the accelerator hardware. Both the MPM and the interface hardware are mapped into the memory space of the BCCs. They only transfer data when requested to by the high-level computers.

It is important to remember the difference between the architecture of a system and how it is implemented. Within a well defined architecture, one can make hardware or software changes to improve some aspect of performance without affecting systems which are outside of the boundary of the control system.

II. DESIGN PROCEDURE

Having decided that the current CESR control system is a suitable model for the B factory, we proceeded to analyze the strengths and weaknesses of our system and the different needs of the new system. Part of this process was defining the scope of the control system.

A. Boundaries of the Control System

For a large design project, well defined boundaries are essential. At the boundaries, the needs of other people must be taken into consideration, and the design process requires communication between the designers and the users of the control system. Within the boundaries, the control system designers can do whatever is needed. We have defined the scope of the control system by defining interfaces for application programmers, instrumentation designers, and operators.

Application programmers must be provided a complete, well documented, set of functions which meet their needs. Programmers are not allowed to bypass these functions by using calls to lower-level routines. CESR uses approximately 35 functions.

Designers of instrumentation hardware are provided with a specification for constructing interfaces to the control system. This includes mechanical, electrical, and protocol details. Recommendations that simplify the control system are included, but not required. This encompasses things like avoiding write-only registers and not having read operations change the state of the system.

The actual implementation of the operator interface is a combination of efforts by both the application programmers and the instrumentation designers. However, it is essential to know the needs of the operators when designing the control system.

B. Special Requirements of the B Factory

We need to know what makes CESR-B different from CESR and how these differences affect the architecture and implementation of the control system.

The first question is how much larger will the new system be? At this early date, we do not have all of the details from the various design groups (eg. vacuum systems, magnet systems), but we do have general numbers. Combining this information with the fact that the amount of equipment in the tunnel will approximately double, we determined that the new control system will have roughly twice as many output control points as the current system.

*Work supported by the US National Science Foundation

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

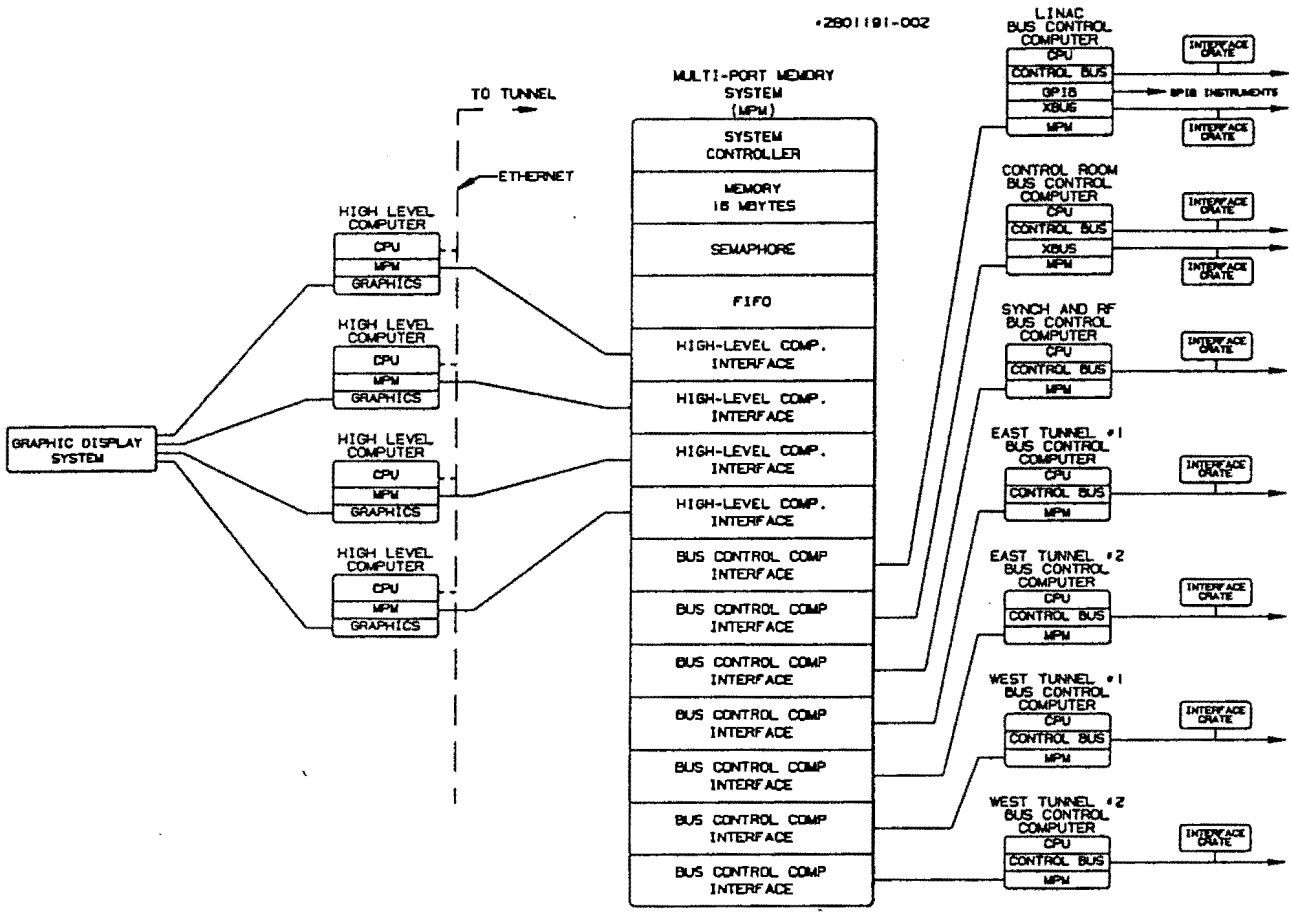

Figure 1. Control System Schematic

We are planning to monitor a great deal more than we currently do. This will help minimize downtime either by indicating when a problem is developing or pointing to the correct area when a malfunction occurs. With the increased monitoring and the additional equipment, we are assuming about five times as many input control points.

There will be several instrumentations systems which require local processors. These systems will need a control system interface for passing processed data. They will also need a connection for downloading and debugging.

C. Strengths of the Existing CESR Control System

Performance is a critical issue. When the operator turns a knob, there should not be a noticeable lag in the response. We run the CONSOLE program at 10 Hz. This is the program that accepts input from the operators and displays results to them. Each time this program runs, it scans the operator input devices, updates the controlled device, and updates the operator's display. It takes about 7 milli-seconds for each pass through this program.

The CESR control system makes very efficient use of the high-level computers. We currently use two VAXstation 3200 computers. The normal application load, consisting of 14 programs, uses 50% of one computer and 30% of the other. This allows special applications (orbit measurements, energy changes) to run in a timely fashion. The efficiency is achieved by minimizing the layers of function calls required to communicate with the hardware and by having processes hibernate when they have requested that the BCCs move a lot of data.

The system is simple and easily extended. We make minimal use of operating system and network functions. This allows us to avoid 'black boxes'; pieces of software over which we have no control. When we added a SUN computer for beam dynamics studies, it was trivial to move the subroutines that are provided to the application programmers.

There is a single database. This allows us to avoid the programs and overhead which would be needed to maintain the consistency of a distributed database.

The database and the interface hardware are memory mapped into the address space of the BCCs. Simple 'move' instructions are used to access the database and the hardware. The control bus can complete a data transfer to the farthest interface crate in less than 15 μsec. Database accesses typically take less than 1 μsec.

Several BCCs can work in parallel on a given transfer request. For example, when reading magnet currents two computers are moving data, one for the east half of the ring

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

and the other for the west half. Three computers control the operator interface.

The graphics display system provides fast and efficient access to many graphics screens, both color and monochrome. It is accessable to all of the high-level computers. Data is sent to it through a FIFO, so the high-level computers simply send data when they have it available.

Geographical addressing is used in the interface crates. Technicians do not need to set address switches on each card. We also insert and remove cards with the power on.

In the tunnel, the entire control system is contained with a metallic exoskeleton for shielding. Feedthroughs are used for connections to the controlled equipment. We are expecting the electrical environment to be more severe in CESR-B.

D. Limitations of the Existing System

The CESR control system has worked extremely well for over a decade. However, it does have limitations, most of which stem from design decisions which were appropriate in the late 1970s.

The address space on the control bus, which connects the interface crates in the tunnel to the bus control computers, is too small. Each control bus has an address space of 65 kwords. This is divided between 16 interface crates, with 16 slots per crate, yielding only 256 addresses per card slot.

We do not have a simple local extension of the control bus. We need to allow designers to build control system interfaces into their equipment and have an easy way to connect to the control bus. Our current system requires two bus operations with a delay of 20 μsec between each operation. A protocol similar to MIL-STD-1553 running in the 5-10 Mhz range would be useful. The length of these extensions would be less than 15 meters.

The interface crates were designed and built by us in the late 1970s. The backplane uses a byte-wide multiplexed protocol on which we transfer one address byte and two data bytes. Since it is a non-standard bus, there is no way to use commercial circuit boards. Producing more crates is very expensive and labor intensive.

There are no facilities for connecting a terminal or a computer in the tunnel. This makes testing and troubleshooting very difficult. We either walk back and forth to the nearest terminal or use the building public address system to communicate with someone at a computer.

The interface from the VAX computers to the MPM is relatively slow. This is due to the fact that the VAX does not have the ability to directly map the entire address space of the MPM. The system uses a set of address and data registers which are located in the Qbus I/O space. To access the MPM, one must first load the desired address into the address register, then transfer the data by reading from or writing to the data register. Since the Qbus has only a 16 bit data path, four Qbus operations are required for each MPM operation. An MPM access requires 12 μsec.

III. HARDWARE

A. High-Level Computers

CESR-B will probably use VAX computers for the major high-level functions. The features of the VAX are that they provide a reasonable development environment, they support priority scheduling of processes, and we are very familiar with them. We have shown that the control system operates comfortably on a VAX, and we expect that the heavier demands of CESR-B can be met by the newer generation VAXes.

As VAX performance improves, the inability to make a memory-mapped interface to the MPM becomes more of a bottle-neck. There is a two-stage plan to address this. First, we will make a 32 bit interface so that setting up the address register and moving the data becomes two operations, instead of four. If even more performance is needed, we will copy the read-only portion of the database into the VAX memory. The VAXes will be clustered to facilitate disk sharing.

There may be other types of high-level computers for special functions. Any computer that we can plug circuit boards into can be interfaced to the control system.

B. Multi-Port Memory System

The multi-port memory (MPM) will most likely be a VMEbus based system, although the Futurebus and any other contenders will be investigated. The MPM contains the RAM used for the database and for message passing. It also contains special hardware which is used to enhance the multi-processor aspects of the system. It can support up to 16 interfaces to high-level or bus control computers.

The system controller in the MPM is supposed to guarantee that under normal conditions no processor has to wait more than 4 μsec for a transfer to complete. This is to satisfy the bus-timeout requirements of the high-level computers, but it means that if the full complement of 16 processors are connected to the MPM, each bus operation must finish in 250 nsec. Several things are done to achieve this. No processor can own the bus for more than one operation. There is neither read-modify-write nor block-mode capability. The system controller contains a special 16-way round-robin arbiter. This uses wiring added to the backplane which provides individual bus request and bus grant lines for each processor interface. The timeout circuit is adjusted according to the needs of the slowest slave device, which is the memory, and must account for the actual access time plus any dead-time from error correction or memory refresh. In CESR, it is set for a 2 μsec period.

CESR uses about 3 Mbytes of a 4 Mbyte RAM board. It is error-correcting memory with a cycle time of 400 nsec. We discovered that the cycle time is more important than the access time in a multi-processor system. One memory board that was supposed to be fast malfunctioned if consecutive accesses occurred too close together. We expect to use

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

between 16 to 32 Mbytes of RAM for CESR-B. The performance of most modern RAM boards will be adequate.

Semaphores are provided for controlling access to shared data areas. The semaphore is a hardware test-and-set register. If the semaphore is not owned, a read operation returns a zero and sets the semaphore to one. If the semaphore is owned, a read operation returns a one. Writing resets the semaphore to zero. There is one semaphore for each longword (4 bytes) of RAM. This simplifies assigning semaphores. One just adds a fixed offset to a RAM address to access the semaphore associated with that address. The semaphores are implemented with a PAL and static RAMs. A 1 Mbit SRAM can make enough semaphores to cover 4 Mbytes of RAM, so it is easy to provide a sufficient number. Semaphore cycle time is 100 nsec.

The FIFO board is provided for passing messages between processors. A single operation can send a message to any number of processors. The FIFO also provides a queue to manage multiple messages to the same processor. The message passing protocol is defined so that the FIFO cannot overflow. Special wiring is provided so that an interrupt can be generated when a processor's FIFO contains a message. A processor can also poll a status register to determine if there is a message. FIFO cycle time is 150 nsec.

C. Bus Control Computers

The BCCs transfer data between the MPM and the accelerator hardware over the control bus, performing a variety of operations on the way. They are supposed to complete each transfer request as quickly as possible, then wait for another request. The computers contain interfaces to the MPM, the control bus, and, in some cases, the CESR Xbus. CESR uses 68020 CPUs and CESR-B will probably use the same family. Each computer runs a single common program; there is not even an operating system on them. The program is written on the VAX in 'C', compiled by a cross-compiler, and downloaded into the MPM. Bootstrap code in the BCCs moves the code from the MPM to local RAM and starts execution.

Under normal load, the CESR bus control computers are idle 85% of the time, which means requests are usually handled immediately. CESR-B will add two more computers to handle the additional equipment, for a total of seven. Boards with 68040 processors should be able to provide the CESR-B control system with enhanced performance appropriate to the heavier load.

The MPM interface passes memory references that fall within a particular address range on to the MPM. The BCCs are in the same physical location as the MPM. In CESR, we use a 32 bit, multiplexed, single ended connection between the BCCs and the MPM. It has an overhead of 300 nsec. For CESR-B, we will provide the same functionality, probably with the same type of interface.

The Xbus interface drives the control bus used in CESR. There are some places where CESR-B might use the same hardware as CESR, for instance in the LINAC or the Control Room, so an Xbus interface is required.

The control bus interface communicates with the interface crates around the lab. The details of this interface will depend on the design of the control bus itself.

D. Interface Crates

Interface crates will be distributed throughout the tunnel and in the control room. In the tunnel, there will be 4 control busses, each with 16 crates. The crates will be configured so as to maximize the effect of the parallel operation of the BCCs. This will involve two busses in each half of the tunnel, with crates alternating between the two busses.

The interface crates will use commercial VMEbus backplanes running the standard VMEbus protocol. At a minimum, they will support short (16 bit) and standard (24 bit) addressing with 1 Mbyte of address space per crate. Data width will be 16 bits. We may choose to support long (32 bit) addressing and 32 bit data.

The interface crates contain I/O devices which we would like to map into the address space of the bus control computers. The crate controller, which interfaces to the control bus, should be a simple design. We do not plan to have a general purpose processor board in each crate.

By using a commercial bus we will be able to buy circuit boards for many functions. However, we will most likely design and build our own interfaces for the more common functions. This will allow us to get exactly what we need, without too few or too many features. Maintenance will be simplified since we will use a consistent design for the bus interface logic.

We want to support geographical addressing, but this requires an extension to the VMEbus specification. Our plan is to divide the short address space between 16 backplane slots, yielding 4 kBytes of address space per slot. We will use four pins on the VMEbus P2 connector. On the boards that we design, these pins will be used to match address bits [A15..A12]. Commercial boards and designs that require more than 4 Kbytes of address space will use the standard (24 bit) addressing mode, with switches or jumpers on the board.

We will also investigate the issues involved with live insertion.

E. Control Bus

The control bus is a data highway between the interface crates and the bus control computers. We have not decided how to implement this connection. The speed and performance should be as good as or better than the Xbus used in CESR. This bus has about 4 μsec of protocol overhead plus a round-trip propagation delay of about 3.5 nsec per foot. It uses differential data transmission for noise control and parity for error detection. At a minimum, we need to transfer 24 bits of address, 16 bits of data, plus some control signals. The three options under consideration are a fully parallel system, an address/data multiplexed system, and a serial system.

The parallel system is logically the simplest. We would just need to buffer the address and data lines of the BCC.

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

Protocol time could be less than a μsecond and throughput would be limited by propagation delays. The drawbacks are the amount of cabling and the number of connectors and bus receivers. Forty wire pairs are required just for the address and data signals.

The multiplexed system, where the address and data signals share the cable, is almost as simple as the parallel one. The receiving boards would need an address latch, but fewer receivers. Another μsecond would be added to the protocol time for the address transmission phase. It still needs in excess of 30 wire pairs, so the cabling is not trivial, but it is a one-time process. There are commercial products that could satisfy our needs.

A serial system requires only a single conductor. It could be a fibre which would provide noise immunity. Cabling would be simple. The disadvantages are speed and electronics complexity. Moving more than forty bits of data in 4 μsec indicates that the data rate would need to be in excess of 10 Mhz. Serial-to-parallel conversion would be needed at both ends. We are looking at some FDDI chipsets which would simplify this type design.

F. Control Bus Extension

The control bus extension allows designers to build a control system interface into their systems and eliminates the need to bring many analog and digital signals to the interface crate. The extension will provide up to a 4 kbyte address space, which may be shared between several remote devices.

The choices for a bus extension are the same as for the main control bus. Minimizing congestion in the tunnel by minimizing the number and size of the cables and connectors makes a serial protocol highly desirable. We are looking at implementations using MIL-STD-1553, 16 Mbit per second token ring, high-speed UARTS, and TAXI chips.

G. Graphic Display System

The graphics display system will provide each high-level computer with direct access to video displays. There will be at least 4 color display channels and 16 monochrome channels. The channels will be distributed throughout the lab through our video distribution system.

There is a large FIFO connected to each computer. When a program needs to update a display, it allocates the FIFO and sends its data. Aside from the appearance of the graphics, there is no acknowledgement that the data has been transferred. The high-level computer doesn't have to wait.

We will be investigating X-terminals. In particular, we will look at their response time (can we turn a knob and have a reading on the screen track the changes?) and the amount of computer resources that they use.

H. Special Function Hardware

Some applications require a dedicated processor. These systems either handle large amounts of data or need higher update rates (>10Hz) than can be handled by the bus control computers. These include beam position monitoring, beam lifetime monitoring, the collision assurance system, and feedback control equipment. These systems interface to the control system via a shared memory and only transfer data that is needed by the high-level computers. An ethernet will be provided in the tunnel for maintenance functions, such as downloading and debugging.

The GPIB is supported by an interface in one of the BCCs. We do not have any CAMAC plans, but if needed, it too can be driven by a bus control computer.

IV. DATABASE

The database contains all of the information required to define the accelerator hardware and to communicate with it.

The heart of the database is the Name Table. It contains an entry for each node in the control system, where a node is a grouping of related hardware or software entities. The name table entry for each node contains a 12 character mnemonic name, information about how many elements and properties the node has, and pointers into the data area for each property. Properties include control bus addresses, scale factors, and raw and processed data. There is also information about which BCCs are used for a given node and the type of processing that is performed on the data.

Additional data structures are the hash table, the link table, the request packet area, the request packet address table, and the data area. These structures will be described by way of looking at a typical operation.

V. TYPICAL OPERATION

An example of a common operation is reading the quadrupole magnet currents. The application program requests data by making the subroutine call:

```
call vxgetn('CSR QUAD CUR',num1,num2,readout_vec)
```

where 'CSR QUAD CUR' is the mnemonic name for the CESR quadrupole magnet current node, 'num1' and 'num2' are the first and last elements of this node that the user wants to read out, and 'readout_vec' is an array where the data will be returned. This subroutine, like most of the control system routines, will not make any subroutine calls. It will directly communicate with the MPM. We will go through the sequence of operations performed by this subroutine and show how the hardware, software, and database function together. Performance measurements based on CESR will be provided.

A. Initialization

The VAX computers cannot directly map the MPM, but instead use interface registers. The VAX to MPM interface provides 32 sets of registers. When the VAX is booted, the VMS program 'SYSGEN' is used to create 32 dummy devices,

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S03SRD03

one for each register set. With this technique, the operating system handles allocation and cleanup. A program allocates a set of registers and owns them for the duration of its execution.

The SUN computer can directly address the entire MPM address space, so its programs simply use pointers. The mapping registers must be initialized at boot time.

The program also needs to allocate a Request Packet. It does this by reading the semaphores associated with the Request Packet Address Table. When an unowned semaphore is encountered, the program records the number and address of the Request Packet.

B. Search the Name Table

The mnemonic name is hashed by exclusive ORing the three longwords of the name and dividing the results by the size of the Hash Table. This produces an index into the Hash Table. The Hash Table entry, which is a pointer into the Name Table, is retrieved. The requested name is compared with the name found in the name table. If they match, then the search has been successful. If they do not match, then the Link Table, at the same offset, has the index of a new Hash Table entry. The name comparison is repeated.

In CESR, with over 900 nodes in the Name Table, the names will match on the first try 90% of the time. No lookups require more than three tries. A name table lookup requires 100 μsec.

C. Set Up and Deliver Request

The Request Packet owned by this process is filled in. The program inserts the Name Table pointer, the number of the first and last elements desired, and the type of operation that the BCC should perform. It also inserts 'BCCs used' bit field, which identifies which BCCs may be involved. This piece of data is inserted in four entries of the Request Packet; the 'used', 'start', 'done', and 'error' entries.

The 'BCCs used' bit field is combined (ORed) with the number of the Request Packet. The result is written to the FIFO board, which signals the appropriate BCCs that there is work for them. The logic on the FIFO board causes the number of the request packet to be written into all of the FIFOs which have a bit set in the bit field.

At this point, the program waits for the bus control computers to move the data. Programs can either go into a wait loop or they can hibernate. The choice depends upon the programmer and how many elements are involved. Most programs hibernate, which contributes to the efficiency of the high-level computers.

D. Bus Control Computer Operation

The BCCs are normally executing in a loop, checking their FIFO to see if there is a message. When there is one, the BCC reads the Request Packet number from the FIFO. In the Request Packet, the computer clears the bit which identifies it in the 'start' entry. Since there may be several bits set, a semaphore must be used to guarantee that only one computer at a time is changing a bit.

From the Request Packet, the computer gets the name table pointer, the element numbers, and the operation mode. It verifies that all database pointers required for the operation are valid. It uses an entry in the database which specifies the first and last elements that this computer handles to modify the element numbers from the request Packet. This keeps the computer from spending time trying to transfer data related to elements controlled by another computer.

Once all of the checking is complete, data movement begins. The control bus address of the next element is retrieved from the database. If the address indicates that the element is controlled by this computer, the raw value of current is read over the control bus from the magnet and stored in the database. If the element is controlled by another BCC, then this computer goes on to the next element.

Using parameters from the database, the raw value is then scaled and offset. The final result is written to the database. The time and status of the operation are also stored. Error information, if any, is saved. The process continues until the last element is done. Notice that other BCCs may be working on this request at the same time.

After all of the elements have been processed, the computer clears its bit in the 'done' entry of the Request Packet and in the 'error' entry (if there were no errors). Again, semaphores are used when changing bits.

E. Retrieve Status and Data

The high-level computer reads the 'done' entry in the request packet. When all of the bits are zero, then all of the bus control computers have finished. If the 'error' entry is all zeroes, then there were no errors. The data is read from the database and moved into the user's array.

In CESR, elements 1 through 49 of the 'CSR QUAD CUR' node are on one control bus and elements 50 through 98 are on another. The execution time of the 'vxgetn' subroutine to read the current from one magnet takes 700 μsec. Reading 49 currents requires 4780 μsec, or 80 μsec per element. The bus control computers use 50 μsec and the high-level computer uses 30 μsec. When reading all 98 elements, the advantage of the parallel operation of the BCCs becomes obvious. It takes 6200 μsec. The extra 1420 μsec is just the time that the high-level computer needs to move the data for the additional elements into the user's array.

VI. REFERENCES

[1] CESR/CLEO Staff,"Conceptual Design for a B Factory Based on CESR," CLNS 91-1050, 1991.

[2] C. R. Strohman and S. B. Peck, "Architecture and Performance of the New CESR Control System," in the *Proc. of the 1989 IEEE Particle Accelerator Conference*, Chicago, IL, March 1989. pp. 1687-1689.