CONF-870498--5

EVOLUTION AND DEVELOPMENT OF THE OAK RIDGE 25URC TANDEM ACCELERATOR CONTROL SYSTEM*

R. C. Juras, N. F. Ziegler, M. J. Meigs,
R. L. McPherson, D. E. Hoglund,** and J. A. Biggerstaff

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831, USA

CONF-870498--5

DE87 008830

ABSTRACT

Since acceptance of the 25URC accelerator in 1982, we have continued to develop and improve both the accelerator control system and associated software. In this paper, we describe these improvements and also discuss how our experience with the present system would influence the architecture and design of future, similar systems.

INTRODUCTION

The Holifield Heavy Ion Research Facility tandem accelerator has been in operation since 1982 and its operational characteristics have been previously reported.[1,2] Likewise, the control system, largely designed a decade ago, has been described elsewhere[3,4,5] and will be reviewed here briefly, along with recent developments, to illustrate evolutionary trends in a computer-based control system. Later in the paper we will discuss how our experience with this system would influence the design of new systems.

A block diagram of the control system is shown in Fig. 1. At the

*Research sponsored by the Division of Basic Energy Sciences, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

**Co-Op student, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.

hub of the control system are a pair of 32-bit minicomputers. These computers are "tightly" coupled by a shared memory. One of these computers, the Message Switching Computer (MSC), contains a direct memory access (DMA) interface to six CAMAC serial highways. On these highways, 16 CAMAC crates, some at elevated potential (in the injector and in the tandem accelerator column), contain the ADC's, DAC's and other I/O signal conditioning modules used to control and monitor the accelerator equipment. Two additional CAMAC crates contain the electronics used to provide operator controls and displays at two control consoles.

CONTROL CONSOLES

The operator interface consists of two identical consoles (Fig. 2), each equipped with a color alphanumeric display and an associated page selector keypad and trackball driven cursor. Also located on each console are three "assignable" analog meters, six dedicated meters, three assignable knobs, eight assignable "analog jacks," and an alarm panel. Panel-mounted oscilloscopes are used to monitor accelerator terminal capacitive pickups, analyzing magnet NMR's, beam profile monitors and capacitive pickups for bunched beams. Emergency off buttons are mounted on the console for use in case of computer failure or the need for a quick shutdown.

In operation, the cursor is moved to the item of interest on the CRT page. A "do-it" button mounted next to the trackball may then be used to control the status of the device (for example, a device may be turned on or off). The device may be monitored on an analog meter by pressing a button next to the meter. The device label appears above the meter and the range and units appear below the meter. Similarly, a shaft encoder may be assigned to adjust the device indicated by the cursor by pressing a button next to the shaft encoder. Each shaft encoder has associated "save" and "restore" buttons so that the current setting may be saved and subsequently retrieved if the change is not beneficial. A range switch associated with each shaft encoder permits selection of 2, 4, 8, ... 4096 turns full scale. One of eight digital-to-analog converter outputs on the back of the console may be assigned to monitor the device indicated by the cursor. This is accomplished by pressing one of eight "analog jack" assign buttons on the console. These outputs are normally connected to a strip chart recorder.

The control consoles as described above have proven to be a reasonably convenient means of tuning the tandem accelerator. However, experience has prompted us to make some enhancements.

One of our first improvements was to add alarm functions. With only one CRT page displayed, out of a possible 100, it is difficult to know when, for example, a power supply has tripped off or a vacuum level is deteriorating. The original alarm panel had lights only for transmission errors on each serial highway and for computer faults. We implemented alarm lights which indicate that the operator should select a specified CRT page. On that page, the alarm is highlighted by a red background. The operator may acknowledge the alarm by moving the cursor to the alarm. The alarm background then turns yellow and if there are no other active alarms on the page, the alarm panel light extinguishes. If the alarm condition goes away, the alarm is rearmed after a short delay.

The console originally contained a voice annunciator for alarms. The mechanism proved to be annoying and unreliable, and was removed. Although our operators are unanimous in rejecting the voice annunciator concept, it is acknowledged that an acceptable "attention grabbing" annunciator is needed.

Another improvement has involved a subtle enhancement in the assignment of shaft encoders; pressing the assign button for a shaft encoder when the CRT cursor does not point to a legal device results in retrieval of the previous assignment instead of a null assignment. Thus it is possible to toggle between two shaft encoder assignments without moving the cursor.

A similar improvement has been made in CRT page selection; pressing the "Enter" key on the page select keypad with no number entered recalls the previous page. Thus it is possible to toggle between two CRT pages simply by pressing the enter button.

The change in the method of shaft encoder assignment noted above lessens the operators' complaint that, since many accelerator controls naturally occur in pairs of pairs (such as the x and y axis controls for a quadrupole lens and the x and y axis controls for a steerer), four shaft encoders would be better than three. In fact, we do plan to implement four shaft encoders per console.

We now believe that the shaft encoders are serviced by the software at an unnecessarily high rate. At present, the value is read from the shaft encoder hardware and sent out to the selected device 50 times per second, except for bending magnet power supplies which are updated twice per second. We now believe that twice per second is an adequate update rate for all devices.

The CRT page itself has undergone several changes, principally to provide more information to the operator. A field was added on each line to show, in percent of full scale, the control output for the device on that line. The number of active lines was increased from 18 to 24. The top line of the CRT shows the assignments for the analog jacks. The date and time of day are displayed. The color of CRT lines with active shaft encoder assignments is changed so that operators at different consoles can avoid conflicts.

Possibly the most important addition is a CRT terminal located next to each console. The terminals are connected to the Control and Supervisory Computer (CSC). Using a terminal, the operator may invoke several of the software aids to operation which will be described later. The terminals are equipped with programmable function keys which minimize the amount of typing necessary; programs may be started with only one keystroke. These terminals are also used to communicate with the MSC software by means of a message-passing buffer in shared memory.

COMPUTERS

A major step in the evolution of the control system was replacement of the original Concurrent (the company previously was named Perkin-Elmer, and before that Interdata) model 7/32 computers with Concurrent model 8/32 computers.

The model 7/32 computer had only 128 kilobytes of memory and it did not have memory address relocation and protection hardware. This made it necessary to run a single-user, single-tasking operating system. It was also becoming difficult to find parts and service for such old computers. More memory and new address hardware could have been purchased (at considerable expense), but fortunately, an inexpensive solution presented itself. The HHIRF data acquisition system was upgraded from model 8/32 computers to Concurrent 3200 series computers. Thus, two 8/32 computers became available for use as the CSC and a spare parts kit (there was not enough hardware for two complete computers). Later, when another division at the laboratory had a surplus model 8/32 computer, we used it to upgrade the MSC. In the case of the MSC, the above considerations were not as important as the faster execution time of the model 8/32 and the fact that we now have only one type of computer to maintain. In making these changes we were fortunate to be using a computer that is part of an evolving "family." Thus, we were able to use most of the I/O boards from the model 7/32 computer and software changes were minimal.

After making the conversions described above we were able to configure the leftover parts into an operational computer for use as a test stand for new interfaces and, more importantly, as a "hot" spare. We have learned that an operational spare computer is vital, providing a source of boards which are known to function and a way to test suspect boards without disabling the accelerator control computers (which are in continuous operation).

The major advantage of the new CSC computer has been its multi-user capability. We are now able to use a CSC terminal at each control console, as mentioned above, and in several offices for program development. In addition, the CSC has two graphics terminals and is used for setup calculations for the Oak Ridge Isochronous Cyclotron (ORIC). The model 8/32 computer is well suited to such calculations. It has floating point hardware and "writable control store" which makes possible the optimization of often-used code sequences at a level even closer to the computer hardware than the machine code level. The major problem with the model 8/32 computer is its one megabyte address space, of which one quarter megabyte is taken up by the shared memory.

SOFTWARE

Software has undoubtedly been the area of greatest evolution in our control system, as it is likely to be in any computer-based system. Both MSC and CSC software have improved, but for different reasons.

MSC Software

The MSC software consists of two major input/output (I/O) data buffers; the smaller I/O buffer executes 50 times per second and the larger executes twice per second.

The 50 Hz I/O buffer contains input from console control devices such as shaft encoders and do-it buttons, and output to console devices such as analog meters. Accelerator components that are assigned to be controlled or monitored are placed in this buffer at the time of assignment. The task to manipulate this data buffer is started every 20 ms and runs to completion in less than 20 ms. Presently, with both consoles "fully loaded" (all devices assigned), the buffer's I/O is completed in 12.5 ms. The task requires 9 ms for execution (the I/O and task execution are simultaneous with the exception of some "interlocking").

When the 50-times-per-second task is completed, control is relinquished to a lower priority task that executes twice per second. This task does less time-critical work (such as assignments). The associated I/O buffer in shared memory is executed twice per second and contains all of the controlled and monitored devices in the control system. Presently, this buffer's I/O is completed in 56 ms (of the 187.5 ms available).

Apart from the changes in the operator interface described above, the upgrade of the MSC software involved improvements in execution speed and maintainability. Execution speed was greatly improved by the use of a new "optimizing" compiler. Also, wherever possible, the software was streamlined and algorithms improved.

Because of the inefficient machine code produced by our original FORTRAN compiler, extensive use was made of assembly language subroutines and FORTRAN subroutines with in-line assembly language. Since execution time is critical in the 50-times-per-second task, we were forced to laboriously examine the output of the compiler and replace lines of FORTRAN code that resulted in particularly inefficient machine code with in-line assembly language. About the time of the software upgrade, a FORTRAN optimizing compiler became available. This compiler produces machine code that is as good as that produced by a proficient assembly language programmer. We, therefore, removed the in-line assembly language statements from the FORTRAN subroutines (in fact, in-line assembly language code interferes with the optimization) and converted some assembly language subroutines to FORTRAN.

Maintainability of software depends to a large degree on how easy it is to read and understand. Here, the increased use of a high-level language is by itself a great improvement. But, as anyone who has had to maintain software knows, it is possible to write unintelligible programs, even in a high-level language; conscious effort is required to produce coherent software. Maintainability of the control system software has improved as we have worked to make both assembly language and FORTRAN subroutines more comprehensible. This task has been made easier by an important feature of the new FORTRAN, namely, the use of IF-THEN-ELSE constructs. We rewrote several of the FORTRAN subroutines using these constructs.

The use of a database is another way to improve maintainability. In the original implementation, I/O buffers and CRT page buffers were tediously generated by hand. We have implemented a database for each of these. Now the database is changed with an editor and a program is run to generate run-time data buffers. The saving in time is dramatic. For example, where the turnaround time for a change to a CRT page was once measured in hours, it can now be accomplished in minutes.

CSC Software

Together, the shared memory architecture and the philosophy of continuous update of the buffers in shared memory make it easy to write application programs. A program on the CSC doesn't have to perform CAMAC I/O; it simply writes to a data location in shared memory and that control information is sent to the controlled device within one-half second. When a CSC program reads a data location in shared memory the information is never more than one-half second old. We have found that staff members and visitors on short term assignment can quickly begin writing useful programs and that the work of such individuals has had a beneficial effect on the quality of accelerator operation.

The software aids to operation have been previously described.[5] They include programs to set up the accelerator from recorded or scaled parameters, identify ion species and avoid analog beams, maintain a continuous history of accelerator parameters, record and print operating parameters on command, cycle the energy-analyzing magnet, and scan the mass-analyzing magnet to determine the ion source output. Off-line aids include calculation of theoretical lifetimes of stripping foils, calculation of dispersion and energy loss due to stripping foils, and calculation of tandem parameters for manual setup - including predicted charge-state fractions after the strippers. In addition, programs on the CSC precalculate ORIC accelerator parameters for coupled operation.

NEW ACCELERATOR EQUIPMENT

Beam Lines

We have added controls for several beam lines. The key to reducing software cost is to make controls for new beam lines as similar to existing controls as possible. Even seemingly minor differences in the details of the hardware interface can result in large software cost. For example, in the original control system, CAMAC output registers for status control were segregated according to whether the type of control function was toggle action or momentary action. The first experimental beam lines for the accelerator were designed by a group that did not know this rule. The two types of control function were mixed and when control of the beam lines was added to the control system, we were faced with either changing the software or rewiring all the beam lines and changing the documentation. We chose to change the software. Recent beam lines have been added with little software cost because we have adhered to a standard hardware interface.

New Beam Chopper/Buncher

The largest new device added recently was a new beam chopper/buncher.[6] Because of the unique nature of the chopper/buncher controls, it appeared that costly modifications to MSC software would be necessary. To minimize the costs and to add some local intelligence to the buncher controls, we decided to use a microcomputer in the buncher CAMAC crate. It appeared that the microcomputer would reduce costs by translating buncher parameters to a format compatible with the MSC software. The MSC could then read and write locations in a memory in the buncher CAMAC crate as if these locations were buncher CAMAC modules. The microcomputer would be responsible for maintaining a relationship between these memory locations and the actual CAMAC modules (Fig. 3).

A memory like the one described above is called a "mailbox" memory. Unfortunately, commercial mailbox memories are not well suited to our scheme. Because of the small address space of a CAMAC module (only 16 addresses per function code), most mailbox memories use an internal pointer. This pointer can be set to any value. When the memory is written, the pointer is incremented to the next memory location. Consequently, the memory appears as just a single address to the computer. This is a problem in a control system for several reasons. For example: if a serial highway error occurs in the middle of a block of data, the serial crate controller will reject the erroneous command(s). Therefore, data following the error will no longer be associated with the correct memory locations. One could envision drastic consequences of highway errors such as power being turned off and on, or power levels changing. The solution to this problem was to build a simple, small (only 32 words) mailbox memory. Each control or monitor may then have a unique memory location and highway errors are not catastrophic (recall that in our system, data are refreshed every one-half second). In fact, the microcomputer/mailbox memory worked so well that no MSC software modifications were necessary (aside from additions to the database). The software for the microcomputer represented a far smaller effort than modification of the MSC software.

New Interface

A new direct memory access (DMA) interface between the MSC and CAMAC was constructed. The primary motivation for this work was to build a spare for the existing interface. The new design makes extensive use of programmable logic devices (PLD's). Our experience with the PLD's is favorable. They result in a considerable savings of board space and wiring costs. Changes can often be made simply by programming a new device.

Fiber Optic Highway

The serial highway in the injector was upgraded to fiber optics for two reasons. First, serial highway errors were caused by injector sparking. This was not a serious problem (in fact the serial highway error light served as a useful indicator of injector sparking - usually caused by excessive cesium flow) and probably could have been solved by careful grounding and shielding. Secondly, the equipment for the previous scheme (light links through air across the gap between injector deck potential and ion source potential) took up too much valuable space and required several power supplies. The new link resides in a CAMAC module and thus requires no additional space or power supplies.

CONCLUSIONS

Here we try to answer the question "What have we learned from our experience with this control system?"

We have learned that it is vital to program as much as possible in a high-level language. First, a high-level language reduces the initial cost of software by making the programmer more productive. Second, maintenance costs are reduced because the programs are easier to understand and modify. Third, a high-level language facilitates computer upgrades. The use of an optimizing compiler permits the use of the high-level language even for time-critical software. In fact, the availability of good compilers is an important consideration in the selection of a computer for the control system.

We chose FORTRAN for our system. FORTRAN is still a good choice, but other high-level languages which satisfy the criteria of readability and maintainability would also serve. The important point is that the language should be widely known so that individuals on short-term assignment do not have to spend time learning a new language.

Another software consideration concerns the management of data. Data buffers should be generated from a database. Additions and changes are easier because, for example, it is not necessary to work in a coded binary format, but in an easily understood text format. Documentation is improved. Changes can be made more quickly and with smaller probability of error.

The control system should be designed so that programs to aid operations are easy to write. It should be possible for programmers not intimately familiar with the workings of the control system to write application programs. In other words, one should not have to be concerned with the details of transferring data to and from remote equipment when writing application programs.

Periodic refresh of all control system parameters is valuable. With periodic refresh, there is no need to be concerned about loss of data stored in remote locations with a spark or a remote power disturbance.

Provision should be made for logging accelerator parameters. In our experience, the combination of a logging program that takes a snapshot every three minutes and a strip chart recorder for monitoring more rapid changes has proved valuable.

Hardware for a "distributed processing" architecture was not available when our control system was first designed. Now, many single board computers (SBC's) are available in CAMAC and other formats. We have learned that SBC's in remote crates can save software costs for unique devices by converting data to a standard form for the central computer. This was true in the case of the beam chopper/buncher even though we were forced to program the SBC in assembly language. The control consoles would be a good place for SBC's to aid the central computer by performing format conversions. The same arguments for the use of high-level languages for the SBC's as for the central computers can be made, but this software development is lagging.

ACKNOWLEDGEMENTS

We wish to express our appreciation to staff members of the National Electrostatics Corporation for their work on the control system. We also wish to thank P. K. Kloeppel and K. A. Erb for their work on software aids to accelerator operation and T. P. Cleary who wrote the programs to precalculate ORIC parameters.

REFERENCES

1. J. K. Bair, J. A. Biggerstaff, C. M. Jones, J. D. Larson, J. W. McConnell, W. T. Milner, and N. F. Ziegler, IEEE Trans. Nucl. Sci. NS-22, No. 3 (1975) 1655.

2. C. M. Jones, G. D. Alton, J. B. Ball, J. A. Biggerstaff, D. T. Dowling, K. A. Erb, D. L. Haynes, D. E. Hoglund, E. D. Hudson, R. C. Juras, S. N. Lane, C. A. Ludemann, J. A. Martin, S. W. Mosko, D. K. Olsen, E. G. Richardson, P. H. Stelson, and N. F. Ziegler, Nucl. Instr. and Meth. A244 (1985) 7.

3. J. A. Biggerstaff, Proc. 3rd Int. Conf. on Electrostatic Accelerator Technology, Oak Ridge, Tennessee (1981), IEEE Catalog No. 81C41639-4, p. 121.

4. N. F. Ziegler, Proc. Symp. of Northeastern Accelerator Personnel, Oak Ridge, Tennessee (1978) ORNL (1) CONF-781051, p. 379.

5. R. C. Juras, J. A. Biggerstaff, and D. E. Hoglund, Nucl. Instr. and Meth. A247 (1986) 25.

6. N. F. Ziegler, G. K. Shulze, J. Rochelle, W. T. Milner, M. J. Meigs, R. C. Juras, "Recent Electronic Improvements to the Oak Ridge 25URC Accelerator," this conference.

FIGURE CAPTIONS

1. Block diagram of the tandem accelerator control system.

2. The tandem accelerator control console.

3. Chopper/buncher controls.

ORNL-DWG 87-8855

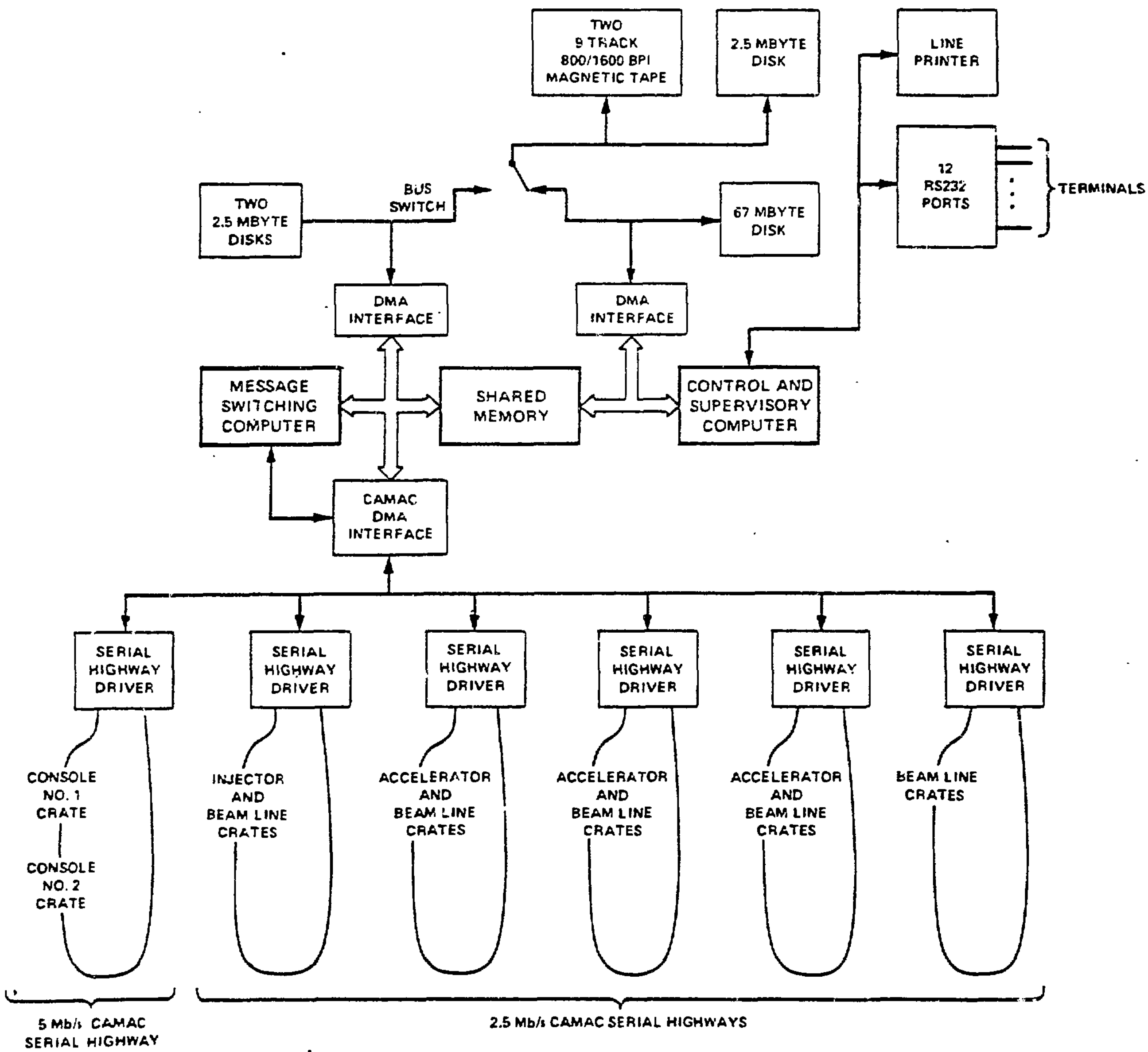
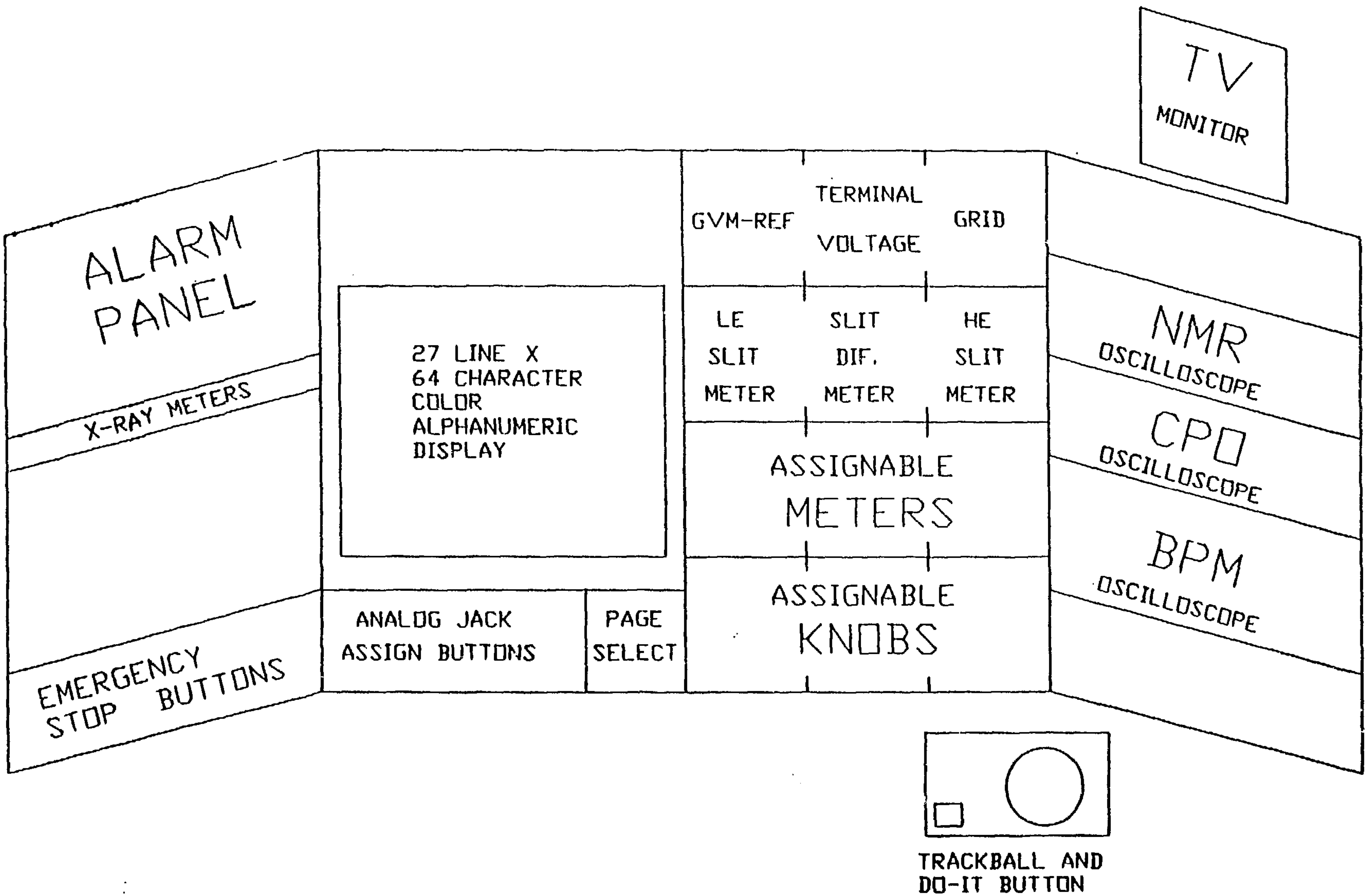


Figure 1

ORNL–DWG 87-9295



ORNL–DWG 87-8858

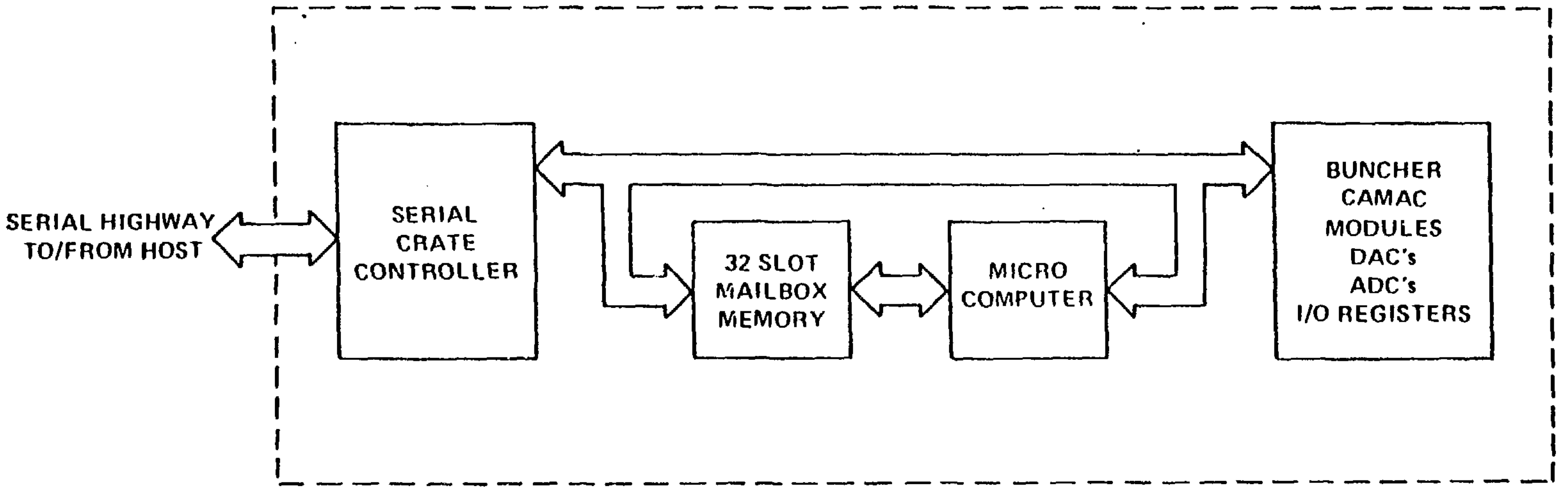


Figure 3