Presented at 1988 European Particle Accelerator Conf., Rome Italy, June 7-11, 1988/

THE AGS BOOSTER CONTROL SYSTEM*

R. FRANKEL, E. AUERBACH, B. CULWICK, T. CLIFFORD,
S. MANDELL, R. MARIOTTI, C. SALWEN and N. SCHUMBURG

Brookhaven National Laboratory
Associated Universities, Inc.
Upton, New York 11973

BNL--41438

DE88 013990

Although moderate in size, the Booster construction project requires a comprehensive control system. There are three operational modes: as a high intensity proton injector for the AGS, as a heavy ion accelerator and injector supporting a wide range of ions and as a polarized proton storage injector. These requirements are met using a workstation based extension of the existing AGS control system. Since the Booster is joining a complex of existing accelerators, the new system will be capable of supporting multiuser operational scenarios.

The Booster timing system will function as part of an overall timing system designed to synchronize and trigger the multiple accelerators which form the AGS complex. Each accelerator in the complex will be supported by an assigned dedicated timing generator (Tandem Van de Graaff (TVDG)), LINAC, *BOOSTER* and AGS. Cycling of each of *these* generators will be initiated by a signal from a Supercycle generator defining the type of cycle which is to occur. A dedicated generator produces the sequence of timing signals required by that accelerator for a particular cycle.

The security system will rely on relays and switches hard-wired together to form an integral system. Computers will serve a monitor only function.

Booster controls will be a part of the AGS control system and share a common architecture. Not only is the same functionality assigned to the distributed components of the system but the components are expected to be software compatible with the existing system. The most significant change for the Booster is at our middle or STATION level where MULTIBUS systems will be replaced by APOLLO workstations.

This change offers cost and performance advantages over the design previously used. The operating system in the STATION will be the manufacturer's standard so that HOSTS (highest layer) and STATIONS will now share a common development environment. Code will be written in a high-level language (C) and *STATIONS* will use the same data structures as HOSTS.

Apollo HOST computers, embedded in consoles or elsewhere, will be interconnected via Apollo's "Domain token ring". This ring will be a logical extension of the existing AGS and TVDG console ring. Fiber optic cables will be employed to provide ground isolation between widely separated computers. The network interface units (cards) will be standard Apollo products. The STATION computers will also be interconnected via the token ring. Again the network interface units and software drivers will be the manufacturer's standard products. The HOST ring and the *STATION* network will be interconnected at two points so that a single point failure cannot break data flow. STATIONS will interconnect to our DEVICE CONTOLLERS (lowest layer), via the IEEE 488 bus. This choice is made so that existing AGS Device Controller designs may be used for the Booster Project with minimum rework.

Device controllers will typically be custom designed MULTIBUS 1 systems as used in the AGS because this system provides the necessary "real-time" and back-plane power to support the extensive hardware required to interface to accelerator devices. Standard interfaces to devices include DATACON (an AGS developed serial process which offers long distance transmission, transformer isolation and a 1 Mbit/s data rate) and IEEE 488. Some devices will be operated directly via hardware installed in the MULTIBUS crate.

APOLLO workstations are the main interactive tool for communication with the accelerator because of their multi-windowed screen, mouse and keyboard. All operation programs will run from any Apollo Workstation so that any workstation can be made to serve as a console including those assigned to be stations.

We have learned from our experience with the Heavy Ion Transfer Line that there is some level of problem supporting knob type control devices. The problem lies in the interface of the systems through which communications must pass. The knob was designed to send out setpoints at 10 per second to match the station which was designed to deliver the setpoints to the device controller as fast as 10 per second. Some devices are not able to receive commands this fast, however, so the device controller has delays built in to accommodate its particular devices. The device controller does not buffer its commands, so it only picks up commands from the station as it needs them.

If commands are being delivered to the station faster than the device controller picks them up the buffer fills up. The station throws away commands received after some period of time if it has no room in its buffer and some commands are lost. This problem can get worse. If "n" power supplies are controlled by one knob, nx10 setpoints per second are sent out, causing this problem to manifest itself faster. The Architecture of control knobs will be improved, however, this type of control is not recommended. Knobs will be supported, but are expected to be of small importance for the high level of control envisaged.

This leads us to the general question of system latency, or the time it takes from when a device command is sent to when the device actually responds is a complex question. Studies were done on the Apollo to find the average time it takes to send a report request to a station and wait for the report to come back. It was found that the average time was approximately .1 seconds with the fastest time approximately .05 seconds and the slowest time

* Work performed under the auspices of the U. S. Department of Energy.

approximately 2.5 seconds. Further delays occur between station and device controller and between device controller and device. These delays have yet to be measured, for planning purposes we are assuming that each move between architectural layers (Host to Station, Station to Device Controller and Device Controller to Device) costs .1 seconds per jump, or .3 seconds as the average latency in the system. We expect that using Apollos in place of Multibus stations, may improve performance to five updates per second.

The design for the user interface will take into consideration the different kinds of people who will use the system. (Engineers, Technicians, Operators and Physicists). By keeping these views in mind, a core of tools for Booster control can be designed and implemented which will satisfy a wide range of users and provide a solid foundation on which to build future development.

Menu tools provide a uniform medium for control programs to interact with users. The menu package will be made available to all programmers to ensure that programs present a consistent and familiar "look and feel". Types of menu tools available are: a means of listing items from which the user will choose; pop-up menus; valuators, standard tools for numerical input which show a pictorial representation of the minimum current and maximum allowable values.

A spreadsheet style program for controlling simple logical devices and parameters is provided for single device control and monitoring. Device lists are provided in a logical tree structure for those who are not machine proficient.

Graphics provide a visually appealing and more intuitive view of accelerator controls than conventional programs. There are commercially available packages which can be used as a basis for providing tools in the accelerator controls environment. Common graphic applications include bar charts, plot packages, histograms, etc.

An example of another type of application is a system put into use at Fermilab where a graphics monitor displays a live schematic of an accelerator system providing static symbols showing components and dynamic symbols displaying real time data.

An alarm will be generated whenever the malfunction of an alarmable system component is detected.

Conditions in a system component that can generate an alarm will be maintained in that component's static database. A few privileged personnel will be given the ability to mask out alarmable conditions, enable others, etc.

A method will exist to query the system for outstanding alarms. Alarmable system components will be organized in a hierarchical manner. At the highest level, one must ensure that the Alarm Receiver Server and the host on which it runs have not themselves faulted. Watchdog programs which live in stations and hosts will have the responsibility for generating alarms. All alarms will be sent to the Alarm Receiver Server.

The Watch Task in the station detects and sends alarms whenever a device or device controller has faulted. A Network Monitor Program running in one of the hosts will poll the network and generate an alarm whenever a station, host, or network component does not respond. A Node Master Program running in a server node will generate an alarm whenever a server program faults. When a primary system component alarms, secondary system components also alarm, and so on. This cascading effect can be remedied by adding some intelligence about component dependencies into either the Watchdog programs or the Alarm Receiver Server.

Alarms will appear on a dedicated display, color coded by severity, with those which require the more immediate attention appearing at the top of the list. A menu driven alarm display program will display and log alarms received from the alarm receiver server. Alarms which overflow the display will be automatically deferred. Operators will be able to interact with the menu to list deferred alarms, erase alarms, undefer alarms, etc.

Due to the inherent difficulties of obtaining information in a distributed system, some mechanism will exist to query the stations and server programs, to obtain a ''current'' snapshot of those system components which are in a state of alarm.

Applications can be layered on top of this kernel which will process alarms for the purpose of short and long term analysis.

Archiving will be a methodology for saving the setpoints and commands of controllable devices with the intent of restoring these values at a later time. Controllable devices include simple devices which have but one setpoint and complex devices like function generators with many setpoints. What will actually be archived is a collection of device lists. The accelerator can be modeled as a tree structure with the major subsystems represented as branches of this tree. The lowest branches of this tree, the leaves, will contain the device lists. In this model, the device lists appear as static entities. Device lists can also be created by query and subsequently archived.

Every Archive will contain header information such as the date, species, etc. Some of this information will be read from a configuration file which will be kept up to date. Operators will also be prompted to enter pertinent comments about the contents and reason for the archive. At some frequency, a program will awaken and archive those components of the accelerator required for restorations. This will insure that the restorable state of the machine is preserved within some minimum window of time. Archives may also be created by operator request using either a query or the tree model.

Archives created at a particular branch of the accelerator tree model can be restored at any lower branch. All devices in device lists in leaves below the branch will be restored.

Accelerator Modeling: We consider first the Booster-Controls interaction requirements in terms of the three areas the beam passes through: (a) injection, (b) acceleration and (c) extraction and matching to the AGS. Injection from either the LINAC or the tranefer line is fairly conventional. Standard accelerator programs, which in BNL usage include SYNCH, MAD, etc., are required for machine studies.

A standard lattice input needs to be part of any data base so that all modeling programs compute the same machine; the ability to transfer results

between such programs is also a requirement. During operation, since the basic machine lattice remains unchanged, a set of outputs (for a ''bare'' machine) from these programs should be available for use in any ''perturbation'' treatment. The basic requirements here are to adjust the machine tunes (horizontal and vertical), or rather to maintain their required separation while avoiding the customary resonances. This modeling-operations interactions should not be a ''closed loop'' procedure; instead, these programs will make available the region of operating parameters to be tried. As experience with the machine, and its matching to the AGS, proceeds, some degree of loop-closing will be developed. Extraction and matching to the AGS follows the programs used by CERN for the ISR. Computation Capability Requirements: A processor for floating point calculations with power of the order of 10 MIPS is needed. This power need not reside in a single unit, but may be distributed among two or three units (since some low degree of parallelism is possible for MAD-like calculations).

Because the Booster control system will require significant support from accelerator modeling programs embedded within the control functions, the Booster database must support a sophisticated view of all aspects of the accelerator, including control data and physics data. There are three types of data which are addressed in the database 1) the dynamic state of the machine data; 2) static slow access data; 3) static fast access control data.

Dynamic Database: the state of the machine data is truly distributed. The simple logical devices have their database in the stations while the complex logical devices have their data in the device controller. Queries of the database are made via host to station requests, with the station examining its own memory and returning to the host the requested information. There is a great deal of data in the accelerator data base which is not used for control and need be accessed only in "operator time".

Items like the name of the person responsible for the repair of equipment, the location of the equipment and the phone number of the phone close to the device are in the data base but do not require a fast access time. Also there are functional relationships among the devices in the accelerator which relationships are useful to high level control programs.

Rather than have private data for applications programs it is much better to have the data in a common database even though there might be just one program which uses the data. The model for data use here is an application program using the database for this type of data at set up time when it is first run. Fast access to this data is not required. An Interbase system will be used for static data entry in the database and for off line retrieval of data. The Interbase database will have a menu driven interface which can be used for data entry and retrieval. In order to have the data in the ''off line'' database accessible to programs we will have a network wide database server program which can be used by applications to retrieve data (slowly) from the database. We expect access times on the order of 0.2 seconds to access the database. This is fast enough for retrieving data which has a low rate of use.

There is a subset of the static data needed for control of devices. This data is extracted from the slow database (Interbase database) and put into a format which allows very fast access of the data. This is the device definition file (DDF) of the current AGS. This file contains addressing information for all devices, conversion functions, set point limits, and other information needed to send commands to devices. It is important that this file does not get filled with information which is not needed for fast access.

Space does not allow a full discussion of our control system, thus we have chosen to highlight several major subsystems rather than over simplifying all systems. Any of this paper's authors would be happy to be contacted about their work.