THE CONTROL SYSTEM FOR THE FERMILAB ACCELERATORS

D. Bogert
Fermi National Accelerator Laboratory*
P.O. Box 500
Batavia, Illinois, 60510, USA

Summary

The five cascading accelerators which together provide the protons and anti-protons for the Fermilab Tevatron fixed target and colliding beams experimental program are controlled using a flexible, yet operationally unified, control system. The system, now functional, has been developed in a somewhat evolutionary fashion over the last six years. Very little of the control system developed more than fifteen years ago for the original 500 GeV accelerator complex remains.

The present system, which includes over 1000 microprocessors, is characterized by a very extensive use of distributed parallel processing, as well as global access to readings and control points by users at twenty identical consoles. Significant design effort was expended to insure rapid parallel access to all data by any console user, whether machine operator or systems specialist. Generality of access to devices, uniformity in the presentation of data, simplicity of system utilization, and ease of programming modification have all been guidelines during the development of the control system. A clock system synchronized to various important machine frequencies is used to schedule the operation of the accelerator.

Accelerator controls is a dynamic field, and experience gained from the control and operation of the superconducting Tevatron collider continues to indicate new areas for the improvement of control of the Fermilab accelerator complex, as well as being a 'laboratory' for assisting in the definition of requirements for the control systems of future accelerators.

Introduction

The Fermilab Accelerator Control system has been reported at several conferences and in several articles while it has been under development (1,2,3,4). These reports have tended to be organized around a traditional demonstration of a controls architecture. In detail, that type of presentation is in fact useful, and it will not be ignored in this presentation. The installation and conversion work to complete a restructuring of the Fermilab control system has now been accomplished. This work, which began in the period 1979/80, has included the following basic items: the addition of the superconducting accelerator, the addition of the anti-proton source, the reconstruction of the fixed target extraction system, and the replacement of the 8-GeV transport line. In addition to the above list of accelerator projects, the control system for the remaining existing pieces of the accelerator complex was replaced in varying degrees. The system now in place was not, however, completely designed and specified in advance. There are several reasons why this is so, and at least two are worthy of emphasis: 1) The project was quite large and probably exceeded the size of our staff to do such an all-inclusive detailed design study, and 2) Electronics and controls is a very rapidly developing technical field, and if one makes a very specific plan which then takes as long as six years to complete, and if one does not permit some degree of innovation in the work, then the completed project is very likely to be both outmoded and insufficient by the time of completion. Having admitted that a truly comprehensive plan of attack, with widely accepted intermediate technical specifications, did in fact not exist, it is perhaps both instructive and an aide to understanding of the system as it is now in operation to make the assumption that such planning did exist and to investigate the Fermilab Control System from the point of view of a uniform overview and philosophy. That is going to be the conceptual map for this discussion.

The most general description of the Fermilab Control System would concentrate on two important concepts, and it is fair to state that these concepts were recognized as important in the very early states of the planning, but many of the details of the implementation of the concepts were very much the result of evolution and necessity. These concepts are called "Devices" and "Distributed Processing." They are both at the heart of the present system, and are, in our opinion, very important to the operation of the now existing accelerator complex. Both of the phrases used in this presentation, "Devices" and "Distributed Processing" are a shorthand for much more than the dictionary definition of the phrases. Originally, it was not possible to control the entire Fermilab accelerator complex from a single console; there were four independent control systems with somewhere between absolutely no intercommunication and an extremely limited and very slow minimal intercommunication between them. To first order, the separation was complete, and an operator wishing to control an item on a system other than the one with which he was dealing at the moment was forced to physically move to a different console. The fact that there were four different, disjoint systems also did not tend to enforce uniformities of effort, and applications programs prepared for one system had little chance of being transported easily to another.

The Concept of a "Device": Selective Acquisition; Uniformity;

The concept of a "Device" is that it is possible to specify in the broadest possible terms a rather uniform entity, with sufficiently general properties and characteristics that both servicing requests for information to be acquired from or delivered to it may be handled in a somewhat uniform fashion. In the most extreme case, all possible information to be transmitted to or delivered from the entire accelerator complex would be specified as a property of some type of "device."

An additional feature of a "Device" would be that it should be possible to make multiple parallel accesses to the device without detailed attention to the behavior of the device at the originating requestor's level. At this point it is probably necessary to introduce a figure outlining the actual organization of the Fermilab Accelerator Control system, because it will be useful to refer to parts of the actual system, and system support features, in terms of the real, rather than a theoretical, system. Figure 1 is such an overview schematic. There are four distinct levels of equipment represented. The top row indicates twenty DEC PDP-11/34 or 11/73 "Console" computers. The next level shows four DEC VAXes performing various "central" jobs. The level below indicates a row of "Front End" computers which are mostly DEC PDP-11/44's or 11/84's. The bottom area of the sketch illustrates the physical links and electronics found in the various component parts of the accelerator complex. A second useful schematic is found in Figure 2, which illustrates in expanded form the organization of distributed processing at a typical Tevatron service building.

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

The twenty consoles are both physically and conceptually identical. They are located at widely separated locations about the site. (The physical hardware is all supported at some distance from the centrally clustered computers, and communications between the distributed hardware and the computers is via individual serial CAMAC links.) The system is structured so that a user may go to any console and request any "applications" program to execute readings, settings, etc. to any part of the accelerator complex. In principle, such a program should operate "transparently" to the parallel execution of any other program, or even of itself on another console. In practice, this extreme generality may not be achieved for two reasons; 1) It may be desired for security reasons to limit certain changes in accelerator operating conditions to consoles in the Main Control Room, and 2) In some cases it may not be possible at a lower level to support an arbitrarily large number of requests for service at an arbitrarily high frequency.

A device as defined for the Fermilab Control system may have an arbitrary combination of supported "properties." Some of the most general properties are readings, settings, basic status, basic control, digital alarm, analog alarm, save/restore. It is not necessary that a device have any particular subset of these properties, and the details of the particular properties may vary widely. An example of a rather simple device might be a power supply which has a DAC connected to it to provide the control voltage, some basic ON/OFF and Reset controls, a status register, and an output voltage (or current read through a transducer) that is then monitored by an ADC as part of an MADC (multiplexed ADC) facility. In this particular example there might actually be a single control card containing the DAC and status and control registers, while the analog readback might be from a physically distinct unit.

All the appropriate addressing and accessing information needs to be provided at some level in the system to permit a general purpose request for service of a particular device property. It is in fact possible to conceive of a variety of methods for servicing particular device properties. It is illustrative to consider the reading property. The original Fermilab Control System contained, relatively speaking, few physical devices. If one summed over all readable variables, readable setting registers, etc., the number was of the order of 6000 such items, spread over the four independent computer systems. It was possible, and efficient, to construct a data pool of all readable information which was constantly refreshed with the latest possible reading. It was possible to maintain a refresh rate of 15Hz. Any user program simply accessed a "mailbox' in memory which contained the given reading

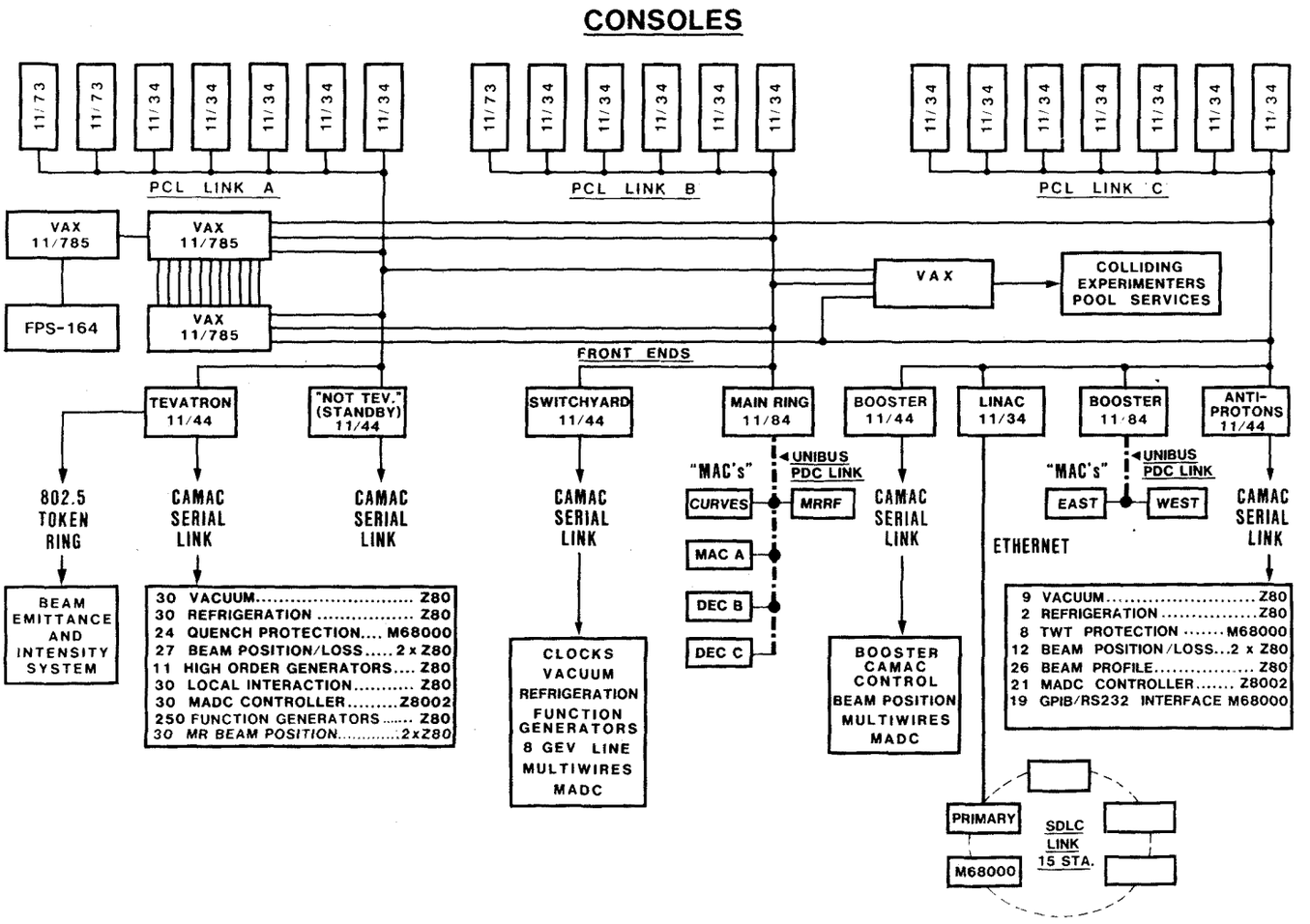

Fig. 1. The Fermilab accelerator controls computer network.

which would be no older than 1/15 second. As the technical requirements for the superconducting accelerator became evident, it was realized that the number of devices and readings implicit in the system was going to increase dramatically. The comparable number of 6000 has grown to over 100,000 and this does not reflect the fact that many of the readings themselves may be large vectors or arrays such that the distributed information may exceed several tens of megabytes. With technology available when the present system was envisioned, it was not possible to maintain a central data pool to be refreshed at a rate such as 15Hz, over a system containing data at this magnitude. Therefore, it was necessary to design a system based upon selectivity, which only gathered or distributed information on demand, and only for as long as it was required, following which the ad hoc data transmission paths would be torn down.

The Motivation for Distributed Processing; Examples

The above considerations combine with a few other primary facts to lead to the introduction of the second principal philosphic topic that is characteristic of our control system: distributed processing. If it is not possible to bring a very wide bandwidth of data to a central location for (possibly) very rapid and high frequency processing, it is necessary to distribute the processing power to the locally resident data. This is only one of perhaps three primary concerns which are: 1) The availability of large amounts of data for processing at high frequency, 2) The availability of relatively large amounts of CPU cycles, and 3) Local operation reliability and availability even in the absence of access to a central facility.

At this time let us consider some examples of the systems which have been distributed in parallel about the Saver accelerator complex, and give some attention to the relative weights of the considerations outlined above. These systems may be located schematically in Figure 2. There are at least two systems which are unique to superconducting accelerator complexes which are good examples of such distributed parallel systems; the Quench Protection Monitors (QPMs) and the satellite refrigerators (FRIGs). Superconducting magnets may be quickly destroyed by the undetected growth of a "normal" region of conductor during a period when current is flowing in the superconductor. If a normality is not detected promptly, and additional current bypassed around a suspect magnet, while removing existing current in the coil to a "dump" and raising the temperature of the entire coil to "normal", then the Joule heating will melt the conductor within approximately two 60Hz line cycles. The necessary heating of the entire magnet is accomplished by discharging a capacitor bank into

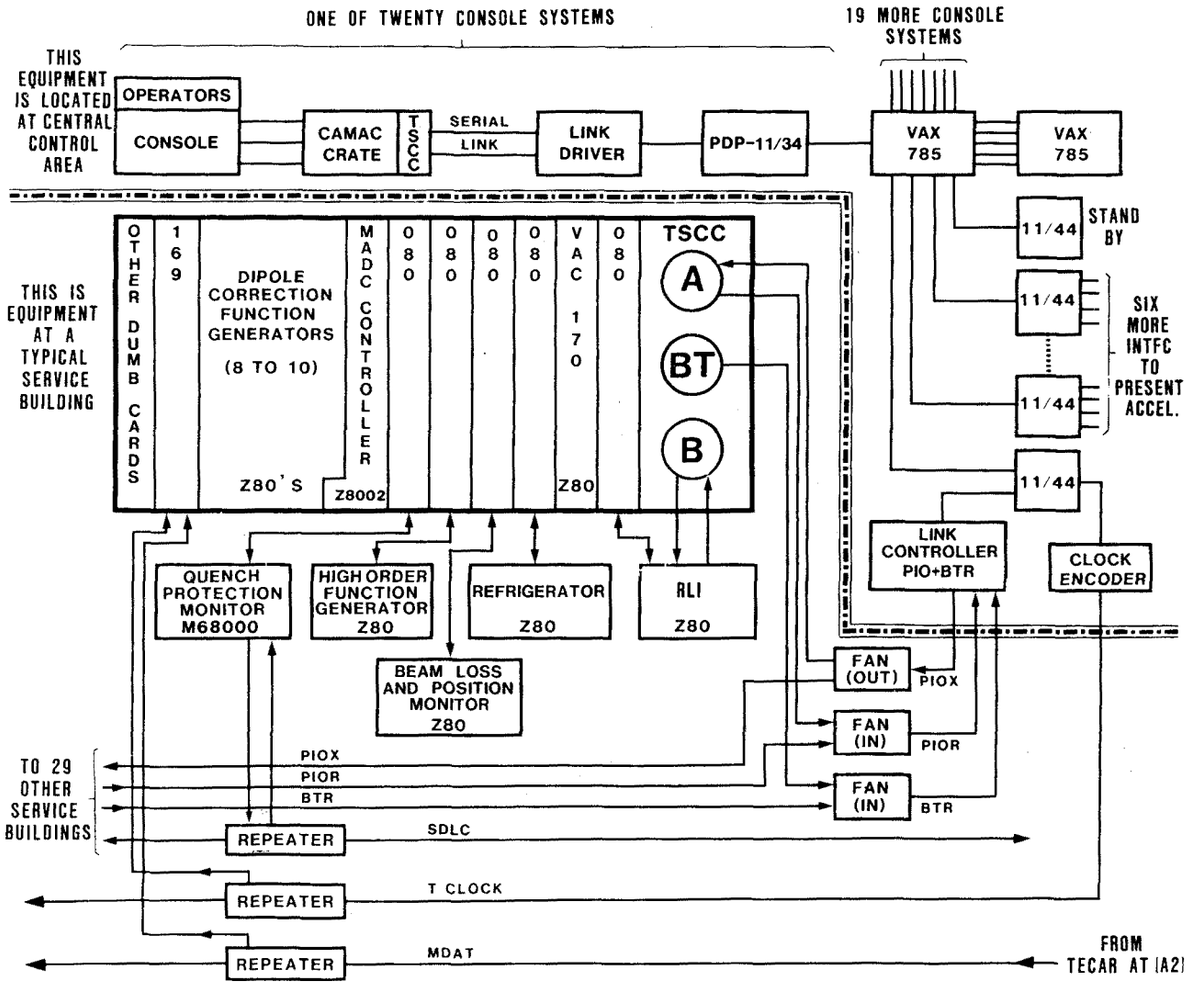

Fig. 2. The Tevatron controls system showing serial link, CAMAC crate, serial crate controller (TSCC), distributed microprocessors, local interaction (RLI), and clock.

special resistive "heater strips" built into the magnet inside the cryostat. Fermilab decided that the safest and most reliable means of achieving the desired protection was "active quench protection" which means that a processor is constantly monitoring the cryogenic element and watching for the development of a voltage drop across the superconducting element in question. It has been argued that it should be possible to perform this protection "passively", by which it is meant that some type of diode circuit could be used to detect the growth of an abnormal voltage across an element, and upon detection, trigger the protective action. Fermilab did not choose this route because it was wished to schedule the protective activity in a controlled fashion (e.g. not fire the heater system until beam was removed) and keep a buffer of magnet conditions immediately prior to quench development and during the protective response activity. These buffers have been crucial to the understanding of magnet performance during quenches, and it is Fermilab opinion that even in the absence of "active" protection triggered by a computer the resultant "passive" system would want the information from them. There remains the question of insuring protection in the event that the processor stops or ceases to loop through its calculation and protection code; Fermilab has addressed this point with the introduction of a "heartbeat" requirement wherein if the processor does not execute a particular piece of code within a sufficiently narrow time window, the heaters will be fired and the magnets made normal.

In principle, the above protection could be provided at a central location provided that the data could be brought there sufficiently quickly and that the central processor could execute the monitor code sufficiently rapidly. The problem is made much easier, however, by the decision to perform the protection function at 24 locations in parallel around the superconducting accelerator. The advantages are that a given processor need only perform 1/24th of the comparable CPU cycles, the processor need only sample 1/24th of the data, and in the event of difficulty resulting in a processor failure, only 1/24th of the ring is warmed up, making recovery much faster.

A second example of a distributed system necessary for the Fermilab superconducting accelerator is the Satellite Refrigeration Control System (FRIG). The Fermilab refrigeration system is based upon one large Central Helium Liquifier Plant with at least eight distributed compressor stations and twenty-four satellite refrigeration plants distributed around the ring. Other refrigeration plant designs are possible which relay on far fewer distributed satellite refrigerators; it is possible at Fermilab, however, to maintain a 1/24th arc sector at cryogenic temperatures (as long as one is not executing fast repetitive ramping) solely with the satellite refrigerator in a "stand along" environment. Since it was not possible to fit sufficient return header volume in the existing tunnel, and the magnets operate at one atmosphere, approximately 80% of the Helium inventory can potentially be lost within 20 minutes of the onset of a site-wide power failure. Generally, such power failures have been of short duration, but the recovery of the central host facility for the control system can take times approximating twenty minutes (at best) to restore full human contact with the entire complex. The satellite refrigerator/compressor distributed processors, however, have been designed to operate with default parameters after an automatic restart when power is restored. This can, and does, result in substantial savings of cryogens. The distributed FRIG processors do far more, however, than provide emergency support. They operate approximately fifteen closed loops and gather a local data pool of all digital and analog channels connected to the local refrigeration plant and the cold magnets. Most of this information is only used locally, being transmitted only on request upwards to the "host" system for applications pages currently executing or if requested by the central logging facility.

In addition, the refrigerators can operate logical devices called "finite state machines" that can be used to program cooldown sequences, or quench recovery sequences, or cold operation, for example.

Both Quench Protection and Refrigeration are examples of distributed parallel processing systems which are capable of very general communications between the host system and the individual subsystem. A "byte stream protocol" called "GAS" was written at Fermilab to support very general transfers of data, requests for data, and even the downloading of executable code segments. The protocol involves a seven byte header, which specifies in reasonably free form the following byte stream. This protocol has proven to be of rather general applicability and has been widely used at Fermilab for the more involved distributed parallel processing systems. In detail, the byte stream is constructed in either processor, routed through CAMAC under the control of the Front End computer, and buffered in FIFO's in the CAMAC 080 module shown in figure 2.

Some distributed systems, however, are not essentially "free form" in the data flow between the host and subsystem. In these systems, definite "mailboxes" are established in distributed shared memory, and any number inserted in such a mailbox will have a unique interpretation. An example of such a system is the Tevatron Dipole Correction Element Function Generator. The Tevatron has about 216 horizontal or vertical correction steering dipoles packaged in "Spool Pieces" around the ring. These must be programmed to achieve a closed orbit at injection, using information from a beam Position Monitor system to calculate correction angles, and then the correction currents must be scaled (in a non-linear fashion as it happens) as the beam is accelerated. The correction needs to be calculated at approximately 1kHz at each location to achieve the desired accuracy. This is again an example wherein if the same calculation were to be attempted at a central location in real time, then both an extremely powerful CPU and quite a high communications band width, with excellent sychronization, would be mandatory. It is certainly easier, both conceptually and in practice, to distribute the job to 216 individual processors which are externally synchronized with the beam acceleration via the Tevatron clock system.

Compromises with Reality; Comment on the Actual Elements of the Control System

The discussion thus far was based upon the artifact that one could imagine several general characteristics of the Fermilab Control System and investigate the operation in a very general sense. At this point, it is necessary to consider the actual realization of the Fermilab Control System, and to comment on areas where an idealized representation or view of the operation disguises many hidden special considerations.

In Figure 1 it may be observed that the lower level of the host computer system consists of approximately seven or eight "Front End" computers which interface a uniform system of Digital Equipment Corporation computers to widely disparate technologies which actually in turn interface to the individual accelerator components. One notes the existance of at least four or five communication technologies between Front Ends and the accelerator. These include CAMAC, Ethernet, SDLC, some recently added IEEE 802.5 Token Ring support, and some remaining support based upon the original Lockheed MAC-16 computers and I/O links emanating from the MAC-16's. It is not too extreme to state that this mix is not ideal and definitely the result of evolution with limited technical support, thereby preventing the complete replacement of older systems at all levels at times when newer systems are introduced.

Several of the technologies employed, but most notably CAMAC (because of its wide use in the Fermilab system) are not ideal for byte-stream interprocessor communication. This is one example of an area that puts extreme demands on the CAMAC Front Ends, since byte stream transmissions must be structured and then "micro-managed" over the CAMAC link. A Block Transfer Return capability mitigates, but does not eliminate, the awkwardness. In addition, the Front Ends must all be prepared to handle multiple simultaneous requests for service from several console or central applications programs. Thus, even for relatively simple "dumb" CAMAC modules the Front End may have to execute some fairly involved management of a request for service; for example a CAMAC module might consist of a pointer set by one action and then service a channel offset by that pointer. Thus, the action consists of two sequential CAMAC commands. It is necessary that one not permit one user to set a pointer and have a second user alter the pointer before the first user executes the second step of the service request. This implies that the Front End must manage all service requests in detail, and not leave to the end user the task of, for instance, choosing a channel in the CAMAC module and setting the pointer. No resource in the accelerator system can be allocated on a fixed basis to a console user in this environment.

The Fermilab Control System relies, exclusively, on the execution of compiled code at the "applications level" within the twenty console PDP-11's and within central applications processes as well. There is no interpretive service (such as interpretive BASIC) offered at all. This is possible because three software services are provided; the first is a central data base containing all the appropriate networking information and addressing, the second is the existence in each Console of a local Data Pool Manager which handles all network requests for the acquisition of data, and the third is the extensive use of an application program called the Parameter Page with a Plotting facility. We will consider all of these facilities in some detail.

The PDP-11 console computers operate under the RSX system. Thus, it is possible to execute several task images simultaneously at a console, and these tasks may in turn share software modules and resources. At present, the Console systems permit up to four applications task images to be executing at a single console, sharing (by allocation) the physical resources of the console. One of these tasks is designated a Primary Application or PA while all of the (up to three) others are Secondary Applications or SA's. Typically, an SA is started from a PA and then retains a limited set of resources, such as a digital display scope. The SA will execute until another task requests the physical resource, at which time the SA is terminated. An extensive set of software subroutines are maintained in a user accessible library so that most data acquisition and display activity at a console need not be rewritten for each newly created PA or SA. Let us consider how an end user at a console might fetch a simple reading in the Fermilab system; let us assume for the sake of discussion that this is a voltage read through an MADC unit somewhere in the Tevatron system. Also, let us assume that the user will employ the particular PA service called the Parameter Page mentioned above. The user is then required to know only one piece of information, the eight character name of the "Device" for which the voltage of interest is the "Reading Property." The user enters the name: for example T:HA15. The Parameter Page (as any other applications page would do) presents this name to a LOCAL Data Pool Manager program that is resident in the particular console PDP-11. An identical image of the DPM is running in each console. The function of the DPM is to collect data requests, actually obtain the requested data, and present the results to the PA and all SA's operating in the particular PDP-11. In order to process the request, the DPM must access the Central Data Base Manager program (DBM) which is resident in the Central (or Operational) VAX. This data base is the heart of the "Device" architecture being described. The data base does not contain ANY current data; it is rather the repository of all the accessing (or addressing) information for all properties of all devices, as well as descriptors and alarm and status text.

The physical connection between the Console PDP-11's, the Operational VAX, and all Front Ends is via DEC's Parallel Communication Link (PCL) hardware. There are three such shared links in the system and they are indicated in Figure 1. Each is a fixed time division multiplexing bus, and the Operational VAX serves as a "message switch" if one node attempts to access a node on another PCL system. A very specific networking protocol called ACNET has been written at Fermilab to support multipacketed message communications from one node to another over the PCL, including the possibility of "open-ended" multiple (or repetative) replies. The DPM creates a message packet which is transmitted to the Operational VAX and the DBM task. To decrease potential demands on the PCL bandwidth, the DPM will coalesce multiple requests for information into single messages. Upon receipt by the DBM, the DBM uses the name (which is one of only three keys to the data base) to access the data base for the necessary information for addressing the Property requested (Reading in our example). The DBM will create a reply packet which is sent back to the particular Console's DPM containing the following general information: which Front End must be accessed, the Device "discriptive text" (about 24 characters), the word length of the reply data when received, the default frequency time descriptor (e.g. 15 Hz, 1Hz, or perhaps what "clock event" the data are read at), and a "Sub-System Device Number" (SSDN) which will be used by the appropriate Front End to recognize the operation that it must perform, and containing sufficient addressing to permit it to know where to perform the operation. Finally, the DBM provides for the use of the Console a Process Data Block (or PDB) containing information regarding the nature of the "raw number" that will be returned from the Front End, what the "Common Transform" to convert the binary number to a raw unit (such as volts on an MADC) is, and finally the transformation, if any, to convert volts to an engineering quantity. If a transformation requires constants, these are also delivered.

With the return of this information from the central data base, the DPM now creates a message packet for the appropriate Front End, DEC-T, in our example. The message is delivered over the PCL to an ACNET task running in DEC-T. The request is queued in a servicing buffer, and if it is a request for multiple replies, it is left in place until torn down, which the requesting console DPM will indicate by refusing to accept any further reply messages. DEC-T in this case will determine from the SSDN encoded information that the service request is an MADC reading at a particular location, and set up the MADC controller to provide the reading. At the requested frequency DEC-T will execute the reading, and deliver a response message packet over ACNET (again coalescing all similar frequency responses for the particular console into one message). When the DPM receives the message packet from its local ACNET driver, it will place the raw response into a "mailbox" in its local data pool, and indicate for the application program (the Parameter Page in our example) the arrival of new information by incrementing an accompanying serial identifier. The appropriate calibrations may then be executed and the result displayed for the user.

In outline, these are the necessary steps for data acquisition in the Fermilab system. Several other "properties" such as "Basic Status" are handled in an analogous fashion, although instead of calibration data the data base contains masks and text to explain particular bits of digital status words. "Settings" are

accomplished in a somewhat inverted fashion; wherein the applications services may accept information in engineering units which may in turn be "anti-calibrated" into raw numbers for delivery to a particular Front End and module. An SSDN is used in an exactly analogous fashion to provide the Front End with a definition of the desired operation and its location.

The example just cited is a particularly straightforward one. The situation can become rapidly more complex (and even cumbersome) as the following discussion will illustrate. In the previous example, the actual acquisition of the "raw" number was effectively accomplished by the Front End. In the case of an "intelligent" distributed parallel processor, such as a FRIG utilized for a Tevatron satellite refrigerator plant, there is an MADC unit directly controlled by the FRIG processor, which is constantly reading all channels from that private MADC at 1 Hz. In this case, the above example is modified in the following fashion. If a console applications page wishes to read a private FRIG MADC channel, which might have a name like T:A1SPWE, the local DPM will access the central data base via DBM and will again be delivered an SSDN and the other necessary information. In this case, however, the SSDN contains different information for the Front End, which might once again be DEC-T. The SSDN will contain system identification rather than absolute addressing information, as well as 'sub-channel' information that may be translated into a request for the precise MADC channel reading desired. In this case DEC-T will establish, using the GAS Protocol, a list of requests to be periodically serviced by the appropriate FRIG processor, and will assign an appropriate list identifier. It is the job of the distributed FRIG processor to be prepared to return the read data to satisfy the list at the appropriate frequency, but the items of the list are not respecified at each repetitive reading. DEC-T then creates a response message packet over ACNET and sends it toward the console requesting the data, until the console refuses to continue to accept it, at which time DEC-T then tears down the list request to the FRIG. At this point one may note an example of a type of inconsistency which arose from the fact that, as mentioned in the onset, the Control System was not designed and specified in advance. The GAS protocol used to communicate between the CAMAC Front Ends and distributed processors is in fact "more general" than the over-lying central data base can support. As an example, GAS is multidimensional, supporting arrays of devices indexed by device "types," device "aspects", and device "entries". A type might be an MADC channel, a motor controller, or even a closed loop. An aspect might be the set point of a closed loop, or its current reading. An entry might specify which one of "N" closed loops. It is possible to perform readings or settings using the GAS protocol by running the concatenated indexes in quite arbitrary fashion, and indeed it has often been found quite desirable to do so, especially when the implicit distributed data blocks in the distributed processors become quite large. Thus, there is a mismatch between some of the generality possible in GAS and the Device services supported in the Central Data Base, making it impossible to read arbitrarily a multi-dimensional array using the Data Base Services. In these cases, the specification of the GAS "TAN" is made explicitly in an applications program, and delivered for service to the Front End, which in turn returns a raw array to the end user. In these cases, however, all data base services such as calibrations are forfeited, and must be explicitly provided by the coder of the applications program.

It should be emphasized, however, that in spite of these limitations the GAS protocol has been widely used at Fermilab; specifically, it is used for communications to the following systems indicated on figure 1: Tevatron, Switchyard, and P-Bar Vacuum scanners; Tevatron, Switchyard and P-Bar Refrigeration; Tevatron, Main Ring, Switchyard, Booster, and P-Bar Beam Position Monitors (BPM's); Tevatron and Switchyard Quench Protection Monitors; Tevatron Power Supply Control; Tevatron Higher Order Function Generators; various GPIB (IEEE-488) devices interfaced using in-house designed systems; and sundry other special devices such as tune measurers and flying wire scanners. It should be noted well that these are complex devices serving a wide variety of purposes in several hundred locations. In spite of this extreme diversity, only ONE handler was necessary for all four of the CAMAC Front Ends, and a very few (less than a half dozen) GAS drivers to match differing computer configurations at the distributed micro-processor level were necessary. This situation should be contrasted with the over 75 specialized drivers that have been constructed for other distributed processors which have not been designed to communicate using GAS but instead have been forced into communication via involved streams of CAMAC CNAF sequences.

Before leaving the subject of the acquisition of data through the Front Ends, let us consider two more examples. These will be from the Main Ring and from the Linac. The conventional Main Ring Accelerator has not been significantly reinstrumented with the exception of the introduction of Tevatron style Beam Position Monitors; conventional service has not changed in the service buildings. The Main Ring utilizes several Lockheed MAC-16's and two DEC PDP-11/55's which formerly were operated as Front Ends between the accelerator equipment and the Main Ring host, which was a Xerox XDS-530. This host supported a 15Hz data pool, which was fed directly by the MAC-16's. As part of the conversion/integration project, the MAC-16's were not replaced for the Main Ring (although several were eliminated for the 8-GeV transport line, the switchyard, and one from the Booster). Therefore, the MAC-16's still collect the data from the conventional Main Ring modules exactly as they have for years past. A new interface was built, however, so that the data gathered by the MAC-16's for delivery to a 'higher' computer is now delivered to the Main Ring PDP-11 Front End called MR-DEC. Since the total size of the 15Hz vector so gathered is relatively small, it is possible for MR-DEC to keep a 15Hz data pool just as was done in the past. Thus, when an applications program requests a "Reading" from a device whose reading is supported through the Main Ring data pool in MR-DEC, the Console DPM delivers an ACNET message containing an SSDN to MR-DEC which uses the SSDN to point to a "mailbox" in its own memory containing the desired reading. A reply message is constructed (repetatively if so desired) and sent to the Console DPM as long as the Console DPM continues to accept the messages. In essence, the operation is exactly analogous to other front end procedures except that the data to be acquired are asynchronously and continuously delivered to a mailbox in MR-DEC.

There are a few special cases where the present control system is a little less responsive than the old system, where application programs executed in the XDS-530. It was possible then, as it still is now, to execute certain MAC-16 I/O commands directly at the request of the applications program. This might be used, for example, to gather some information which is not routinely provided in the MR data pool in MR-DEC. In the past it was relatively easy for the XDS-530 to request the MAC-16's to execute the I/O commands in an ordered and synchronized fashion. In the multi-processor host environment, with asynchronous messages delivered through ACNET drivers over the PCL links, it is harder to maintain definite synchronization, if desired, and certainly can be slower.

Before 1983, the Linac was operated directly as a collection of I/O devices of the Linac XDS-530 system. All the interfacing was purchased from the computer manufacturer and operated directly in response to computer I/O instructions. When consideration was given to the problem of retaining the 15Hz reading/setting

response times that the Linac systems personnel felt crucial to proper operation of the Linac, it was determined that it was not possible to place an "interface" between the existing Linac I/O equipment and another computer manufacturer's I/O architecture (e.g. DEC UNIBUS) without an unacceptable degradation of bandwidth. Therefore, it was necessary to reinstrument the Linac completely. This was done with a slightly different version of distributed parallel processing, utilizing Motorola 68000 series processors. One processor was provided for each major Linac system, for a total of approximately fifteen. Each processor maintains its own complete data pool at 15Hz. The interconnection technology chosen was IBM SDLC on a fiber-optic link. One of the stations serves as a "primary" station and all others are "secondaries", and the primary station supports both an SDLC interface and an Ethernet interface; the Linac Front End PDP-11 supports another Ethernet port on a UNIBUS card. When a Console DPM requests service from the Linac system, the Linac Front End creates a list of requested readings which is delivered to the Linac Primary. The Linac Primary in turn requests the appropriate secondary for the reading via a message over the SDLC system. The secondary finds the reading in its local data pool and provides the answer (repetatively if so requested) to the primary via the SDLC system. The primary creates a response vector to be delivered over Ethernet to the Linac Front End, which in turn creates the ACNET reply message (repetitively if desired) to the Console DPM. Again, the analogy to other Front Ends and styles of data acquisition is quite good, although in detail given aspects of reading or setting services may be either faster or slower. It has been possible to maintain a 15Hz service between a human-controlled knob at a Console Applications level, a physical setting in the Linac, and within the same 15Hz cycle get a readback of the result of the setting. The closure of this "loop" is quite tightly time-constrained in the present system, however. The Linac has an additional feature of note; it is possible for any secondary to request and receive information from any other secondary, and to display such information locally to a user at a small local interaction station.

Central Services

There are some services which are provided centrally rather than being executed when an operator requests a particular application at a console. Some of the most generally used are the Logger facility, Central File Sharing, Save-Compare-Restore, Alarm monitoring and reporting (Aeoleus), the Central Data Base Editor (DABBEL), and the Applications program librarian.

The Logger is an example of a Central Applications Program, one which executes on the Operational VAX. There is a local Data Pool Manager (DPM) on the VAX which operates exactly analogously to those resident in each Console PDP-11. The logger is a program which runs continuously. It gathers data at pre-selected time intervals from across the accelerator complex and saves the readings, with time stamps, in a "shareable" file. The Logger system may be controlled via what is known as "File Sharing" from a Console Applications program; the lists of data being collected may be examined and/or modified, and stored data may be accessed, evaluated, and displayed.

"File Sharing" is a central service which allows programs from any node on the PCL network access to centrally stored files, both for reading and writing. The access is highly analogous to standard file access via Fortran on a single processor, but actual storage is on an Operational VAX disk. An arbitrary number of nodes may access a particular file simultaneously for reading, but only one node at a time may open a particular file for writing purposes. File sharing is the most commonly used means of sharing information across all nodes. In this regard File Sharing serves a purpose frequently assigned to a data base in other control systems architectures.

"Save-Compare-Restore" is a system whereby settings of a predetermined listing of devices, kept in a File Sharing list, may be collected (using the "Save Property" in the Data Base) and the settings saved in a file, with a time stamp and descriptive heading. At a later time this "Save" file may be used either to restore the accelerator subset of devices to the identical condition as at the time of the "Save", or to compare the then existing state of the accelerator with the condition at the time of the "Save".

Alarm reports are also serviced by the Operational VAX, although the exception conditions are actually noted and reported as far "downward" into the accelerator control system as possible, either in distributed processors or in the Front Ends. As the exception condition report is passed up through the system, an "Error Message Code" (EMC) is constructed. The EMC may be thought of as an "inverse address" and is a unique "trail" of the exception report. The EMC's are also recorded in the Data Base and provide an alternate key (in addition to the Device Name) for entry to the Data Base. The VAX central alarm server (Aeoleus) receives the EMC and the exception condition value. It accesses the data base using the EMC to pick up the device descriptor, appropriate masks, and error message texts. This information is then forwarded to all Consoles, where copies of an SA are running which post the alarms in windows on a screen specific for this purpose. The windowing may be established at each Console for the purpose of the operator, and may be different at each Console.

The central data base which is accessed by the Data Base Manager (DBM) or Aeoleus, among others, must be capable of being edited for the addition or modification of devices. A program DABBEL exists to permit the entry, modification, and a formatted reporting of devices. DABBEL is biased toward making the "mass entry" of devices via prepared structured files as easy as possible. A screen editor service for single device entry or modification with extensive assistance for a user less familiar with the structure of the data base is under development.

Applications programs for execution on the Consoles are created on the development VAX. An extensive library of useful subroutines for access to various services described in this paper for the gathering of data is provided. In addition, an equally extensive library for the utilization of Console equipment (keyboard, trackball and knob, interrupt button, touch panel, monochrome and color displays, etc.) is available. Also provided are block outlines of applications programs for the naive user. When a user has written and compiled his code, and an RSX command file for linking the code has been created, the code and command files are submitted to a facility called the "Applications Program Librarian". This librarian facility captures the code and command files, executes the assembly, linking, and loading, and inserts the code in the Program Index. The actual structure of all applications programs is that they are tasks called from a master index program. The code capture feature permits management to have access to a complete source listing of all executing code. Also, library routines may be modified and all affected programs may be found and automatically relinked and reloaded. There are more than 500 applications routines at present. Not all of the 500 applications programs are completely different; several are essentially clones of common services. The most obvious example is the Parameter Page which has been cloned for multiple purposes several dozen times. Applications programs may be directly accessed from inside other applications programs, rather like subroutines. This permits a sequence of operations to be performed in an arbitrary fashion.

Clocks

The accelerator is controlled in real time by a set of "events" (encoded markers) carried on a master clock. There are several clock-like signals which are distributed accelerator-wide. The most heavily used is called the "Tevatron clock" (TCLOCK). TCLOCK is a 10 MHz signal with 10 bit encoded frames. The frames may start on any unused "tick", yielding a 100ns resolution. Some events are more sensibly timed with respect to beam revolution. Two additional clock services (TVBS and MRBS) are synchronized with the (varying) revolution frequencies of the Tevatron and Main Ring respectively. These might be used, for example, to time injection or extraction kickers. An additional accelerator-wide service is MDAT (Machine Data) which broadcasts frames containing the instantaneous excitation energy of the Tevatron and Main Rings, as well as the time derivatives of the excitation energy. These signals, usually distributed at 720 Hz, are used by function generators providing signals for correction elements and low level RF control, among others. Finally, machine abort signals are delivered around the accelerator so that all systems might have access to information concerning the occurance of abort requests.

Recently Added Services; Services Under Development

The Fermilab Accelerator Control system has continued to evolve as service requirements and technical capabilities have suggested. Four newer projects in varying stages of development are: (1) A Data Pool service for colliding-beam experimenters to recover accelerator data of interest, (2) Accelerator Operation Sequencing, (3) A byte-stream oriented link service using IEEE 802.5 token ring technology, and (4) An improved controller for flying wire scanners. This list is not complete but is meant to be suggestive of the areas now under research and devleopment.

During the next "colliding" running period for p-bar on proton experimentation there will be four experiments taking data, including the large colliding detector known as CDF. All of these experiments have requested regular, repetitive access to those accelerator data necessary, for instance, for luminosity determination. Preferably, the individual experiments would like to specify their individual requirements in dynamically alterable lists. Upon reflection, it was realized that these requests had much in common with services one might define and implement for a new type of console service. This additional console service is termed "mini-console" service at Fermilab. The crucial feature of both the "experimenter's data" service and future "mini-console" service is believed, at present, to be the ability to provide data at frequencies typically no greater than 1 Hz. This has permitted a preliminary specification of service based on a central data pool maintained in an additional central VAX as shown in Figure 1. The idea behind this proposal is that at the existing level of twenty consoles one believes that the message handling and structuring of service request in Front Ends (where essentially all requests for data are ultimately serviced) is becoming saturated. The hope is that by pooling "slow" requests in only one more pool in the new VAX, the demands on Front End services will not scale directly with the number of data return request vectors initiated by experimenters, or directly with the number of mini-consoles.

The first implementation of this central pool service (directed specifically at the colliding-beam experimenters) is nearing completion and will be utilized during the next running period late this fall. The additional VAX should be available on a similar time scale; prior to its delivery the service can be developed and supported (at a moderate level) in the existing Operational VAX. The experimenters will use DECNET and their experiment data acquisition VAXes to send the requested data specifications in a vector to the central pool manager, and the acquired data will be returned to the experimenter VAXes over DECNET. The intention is to provide the experimenters with a library of subroutines, with simple calling and argument structures. The experimenters will not be required to develop any deeper understanding of the data acquisition methodology.

When the specifications of a general Application Program for the present system were developed over five years ago it was intended to be possible to "call" one application program from another, and conceivably to "return", just like a subroutine call. An 80 byte transfer vector was provided to facilitate this possibility. The earliest use of this service was to make it possible to start a "SA" from a "PA". The complex sequence of events necessary to set up p-bar on proton collisions once adequate numbers of cooled p-bars are stored in the accumulator ring has encouraged expanded definitions of transfer vectors for communication from one application program to another. The approach taken has been to utilize the originally implemented 80 byte vector to point to arbitrarily larger files available through the File-Sharing service, thus allowing transfers of large amounts of imformation between programs. The expanded capabilities now available have proven adequate to support all services envisioned as necessary to date.

The necessity to support intercommunicating distributed processors which will take data for emittance and intensity measurements throughout the cascaded accelerator complex has coincided with the availability of commercial support for the IEEE 802.5 token ring network. A new UNIBUS based interface using the Texas Instruments TMS-380 chip set for interfacing to IEEE 802.5 has been constructed. It is envisioned that it will be, relatively speaking, straight-forward to modify the existing "GAS handler" in the Front End to distribute and receive byte-streams over this link, permitting communication using developed data base services to the distriubted processors located on the 802.5 ring, while permitting direct interchange, also using the GAS protocol, between processors without using any of the "host" system as a communication manager.

As was shown in Figure 2, where CAMAC and Multibus distributed processors for the Tevatron at a typical service building location were indicated, there was originally no use of VME in the Fermilab Control system. The advent of the availability of 16 and 32 bit microprocessors, and the increasing utilization of commercially available VME modules, has suggested that it is important to be able to construct and interface subsystems using VME backplanes. One of the earliest examples of this technology at Fermilab is a new Flying Wire Scanner Controller. This controller interfaces to the central system via a processor resident in VME that uses the GAS protocol interfaced to the usual CAMAC-080 FIFO buffer. The GAS processor uses shared memory to communicate with a Motrola 68020 processor which operates the flying wire and stores the data from the passage of the wire through the beam. Other similar VME based devices are being developed.

Concluding Summary

The Fermilab Control System continues to evolve as accelerator requirements suggest additions and technology improvements permit. It is anticpated that over the next year the remaining services originally specified over five years ago will be completely implemented. Operational experience with the entire complex in the integrated control system environment will continue to suggest valuable improvements using existing equipment and communications technology. The installation of newer services, such as the central experimenter's pool services and the first processors mutually linked via

IEEE 802.5 links, will provide valuable experience which will be relevant for future controls system planning at Fermilab as well as being a practical laboratory for some of the ideas suggested as appropriate for the control of very large accelerator complexes such as a future SSC.

References

1. D. Bogert, L.J. Chapman, R.J. Ducar and S.L. Segler, IEEE Trans. Nucl. Sci. NS24,3 (June 1981) 2204.
2. D. Bogert and S. Segler, in: Europhysics Conf. Computing in Accelerator Design and Operations, eds. W. Burse and R. Zelazny (Springer, Berlin, 1983)338.
3. H. Edwards, Ann. Rev. Nucl. Part. Sci. 35 (1985)605.
4. D. Bogert, Nuclear Instruments and Methods in Physics Research A247(North Holland, Amsterdam, 1986) 8-24.