DEVELOPMENT OF AN INTEGRATED CONTROL AND MEASUREMENT SYSTEM

ORNL/TM--9012

DE86 002631

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Wayne W. Manges

March 1984

DEVELOPMENT OF AN INTEGRATED CONTROL AND MEASUREMENT SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Wayne W. Manges

March 1984

ACKNOWLEDGEMENTS

I wish to gratefully acknowledge the guidance of my Graduate Committee, Dr. D. W. Bouldin, Dr. J. M. Googe, and Dr. J. D. Birdwell, in preparing the technical content of this thesis.

Special appreciation is due the Instrumentation and Controls Division of the Oak Ridge National Laboratory, operated by the Nuclear Division of Union Carbide Corporation for the United States Department of Energy, for supporting this effort.

I am indebted to J. M. Jansen, Jr. of the Instrumentation and Controls Division for the debates and discussions that helped clarify many of the concepts in this thesis.

My thanks also to LaWanda Klobe, Robin O'Hatnick, Debbie Grubb, and Bonnie Brummitt for their tremendous effort in assembling and typing the final document.

Above all, I wish to recognize the encouragement and support of my wife, Pamela, without whose patience and understanding this achievement would have been impossible.

ABSTRACT

This thesis presents a tutorial on the issues involved in the development of a minicomputer-based, distributed intelligence data acquisition and process control system to support complex experimental facilities. The particular system discussed in this thesis is under development for the Atomic Vapor Laser Isotope Separation (AVLIS) Program at the Oak Ridge Gaseous Diffusion Plant (ORGDP). In the AVLIS program, we were very careful to integrate the computer sections of the implementation into the instrumentation system rather than adding them as an appendage. We then addressed the reliability and availability of the system as a separate concern. Thus, our concept of an integrated control and measurement (ICAM) system forms the basis for this thesis.

This thesis details the logic and philosophy that went into the development of this system and explains why the commercially available turn-key systems generally are not suitable. Also, the issues involved in the specification of the components for such an integrated system are emphasized. Finally, this document is based on my experience and expertise as well as that of respected experts in the field and input from colleagues in the Instrumentation and Controls Division at Oak Ridge National Laboratory.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The Instrumentation and Controls (I&C) Division of the Oak Ridge National Laboratory (ORNL) has been involved in the development of minicomputer-based instrumentation systems for more than 15 years. Early uses emphasized instrument system integration, but today's microprocessor technology has led to the incorporation of computers in instruments. This thesis emphasizes the use of supervisory computers to integrate the functions of a collection of instruments into an instrumentation system rather than the use of embedded microprocessors. However, some of the issues discussed in relation to supervisory computers also apply to embedded computers.

The implementation of the instrumentation and control system described here is based on the author's experience with the integrated control and measurement (ICAM) system currently under development for the Atomic Vapor Laser Isotope Separation (AVLIS) Program at the Oak Ridge Gaseous Diffusion Plant (ORGDP). The ideas and concepts represent those of the author as a result of six years of experience in the ORNL I&C Division. The many discussions and debates during this time have no doubt influenced the contents of this thesis. Numerous conferences and workshops on process control, computer control systems, and systems integration were also influential. The content of this thesis is my interpretation based on the experience, professional contacts, and course work taken over the years. It does not necessarily represent the current feelings of

either a majority of the employees or the managers of the I&C Division or any other part of Union Carbide Corporation or the U.S. Department of Energy.

This thesis addresses the pertinent issues involved in the development of an instrumentation system for a complex experimental facility, a type of facility quite common among the plants in Oak Ridge. This type of facility must operate within prescribed limits over long periods to demonstrate its economic feasibility. A small-scale proof-of-principle demonstration has been completed, and the next step is to scale up the facility to determine if any scale-up problems exist before committing to install a full-scale production system.

Several key issues differentiate this type of laboratory data acquisition and control system from its industrial counterpart. Among these are the uncertainties involved in the process being instrumented, the importance of the data obtained, the degree of sensitivity to cost and schedule, and the human interface requirements. Many industrial users are concerned with how long it will take for the control system to pay for itself,[1] while our major concerns are data availability and flexibility. For research and development (R&D) facilities such as those described here, it is very important not to design roadblocks into the system. Any assumption that leads to a size, speed, or capability restriction must be evaluated very carefully. It is usually more cost effective in these facilities to "err on the side of flexibility." That is to carefully

evaluate the cost of supplying too much capability as well as the cost and consequence of providing too little. The compromise usually taken for the AVLIS design has been to provide for additional capability that may be needed later but not to implement it until it is actually needed.

Because of the unique characteristics of developmental facilities, there has been a continuing need at ORNL for instrumentation systems not available directly from industrial suppliers. This thesis concentrates on the development of such a laboratory data acquisition and control system, emphasizing those areas that differ significantly from industrial systems. This thesis stresses the interaction of all components to form an integrated system that is more than just an accummulation of instruments. A key concept is the selection and implementation of all components, from the sensors in the process area to the operating system software in the supervisory computer. Mistakes here can cause serious problems in the final system design.

The Real-Time Computer Systems Group of the I&C Division was chartered with the responsibility for providing such integrated systems. The issues involved and addressed here are generally passed on by word of mouth or through mentor guidance. One purpose of this thesis is to discuss the issues and considerations of major importance to new members of this group. This thesis is not a detailed treatise on any one subject but rather a summary of the critical issues involved in the integration of an entire instrument system.

While an understanding of the issues presented here will not answer all of the questions, it will help the new engineer ask the right questions.

The development of an integrated instrumentation system involves four general areas: the overall design concepts, the architecture, the process interface hardware and software, and the human interface. In each of these areas, I will describe the specific issues involved, present various alternatives and their consequences, and, in some cases, explain why a particular one of these was selected for implementation for the AVLIS program. In the implementation of these systems, the use of new technologies could lead to delays and failures. Therefore, I will emphasize the use of proven, reliable technologies.

2. OVERALL DESIGN CONCEPTS

The design of an integrated system for a given process involves first a specification of the requirements of the process being instrumented. These initial discussions must include reaching agreement on an overall instrumentation philosophy. To establish the relative priorities of the critical issues and to suggest the concepts that are to be applied in the implementation of the instrument system, AVLIS program personnel held early meetings involving the individuals responsible for instrument development, project engineering, data system development, and controls technology development. This group set out to describe how the instrumentation would be accomplished if no resource or schedule restrictions were imposed. Subsequent implementation plans introduced the compromises necessary to meet the restrictions imposed at the time of the implementation. Such agreement will permit individual members to assess the suitability of specific compromises in the various areas of instrumentation involved. Without an initial philosophy agreement, compromises made in isolation could seriously impact the success of the overall instrumentation effort.

Initial design discussions should center on the issues critical to the development of an instrumentation system, including reliability, availability, maintainability, expandability, flexibility, and system integration. The goal is not to maximize any one of these features at the expense of the others, but rather to provide a mix

facility such as this, it is counterproductive to provide an operator support system that is independent of the experiment/development support system. As the process becomes better understood, what was previously considered a developmental experiment becomes a part of the routine operation.

The experimenter does have requirements that differ from those of the operator, however. (This is one of the areas where support is lacking in industrial process control systems.) The operator is primarily concerned with what the process is doing right now. The experimenter is more interested in how the process reached this state in order to better understand the process itself. The experimenter therefore requires that the ICAM system obtain data faster than it can be viewed in real time. The experimenter expects to be able to reduce the acquired data and have it presented in a form that permits recommending a change in the operations procedure and studying the effect of that change. The real-time requirements are substantially less, but the amount of data required is greater. The experimenter usually does not have major responsibility for the overall operation but may have control over some experimental subset of devices in the process. The observer simply examines data with no responsibility for controlling any part of the process.

Modularity is one of the classic approaches to the issues of reliability, availability, maintainability, expandability, and flexibility. This decomposition of the system into functionally independent entities can offer significant advantages if used properly.[3]

that will achieve the overall goals of the development effort. Any design is evaluated on these issues as well as on overall cost.

In this thesis a process is assumed to be essentially steady state in the long term. That is, to fulfill its requirements, an ICAM system must function reliably over long time periods (several hundreds of hours) without serious interruption. An example of a system that need not provide steady state operation is the instrumentation of an explosion. Theoretically, all preparation for the experiment is done before the test is initiated. The instruments need operate for only a few seconds, with the analysis of the data sometimes taking several years. Some laser fusion experimental facilities are of this nature and instrumentation system for one such facility has been detailed in the literature.[2] The design presented here supports the long-term, steady state operation of an experimental facility. Although some of the concepts in this system could be implemented in burst type systems without compromising its speed requirements, not all of the concepts presented here would be usable in such a transient process.

A consistent user's interface is essential in an integrated system. Three classes of users are supported in the AVLIS process: operators, experimenters, and observers. In general, a steady state process requires an operator who is responsible for the routine operation of the experimental facility. Since the facility under consideration in this thesis (AVLIS) is currently under development, operation of the facility itself is a development issue. In a

facility such as this, it is counterproductive to provide an operator support system that is independent of the experiment/development support system. As the process becomes better understood, what was previously considered a developmental experiment becomes a part of the routine operation.

The experimenter does have requirements that differ from those of the operator, however. (This is one of the areas where support is lacking in industrial process control systems.) The operator is primarily concerned with what the process is doing right now. The experimenter is more interested in how the process reached this state in order to better understand the process itself. The experimenter therefore requires that the ICAM system obtain data faster than it can be viewed in real time. The experimenter expects to be able to reduce the acquired data and have it presented in a form that permits recommending a change in the operations procedure and studying the effect of that change. The real-time requirements are substantially less, but the amount of data required is greater. The experimenter usually does not have major responsibility for the overall operation but may have control over some experimental subset of devices in the process. The observer simply examines data with no responsibility for controlling any part of the process.

Modularity is one of the classic approaches to the issues of reliability, availability, maintainability, expandability, and flexibility. This decomposition of the system into functionally independent entities can offer significant advantages if used properly.[3]

However, one of its major disadvantages is that increasing modularity can increase intercommunication requirements and thereby negatively affect some of the critical issues. A balance must be achieved in which the modularity of the instrument system is compatible with the modularity of the process.

It would be ideal to phase in functionality and performance improvements as well as repair failures in the instrumentation system without making the entire system unavailable, and modularizing both the hardware and software can permit support of this type of upgrading. Modularity can significantly reduce the overall cost of maintenance and development. However, modularity in itself is not sufficient to ensure a viable system design; the modules must be decoupled so that necessary interaction is minimized if the overall design is to benefit from modularity. This decoupling applies to hardware as well as software because the cost of both can increase with the complexity of their interaction. This also encourages the use of standard interface techniques.

3. ARCHITECTURE

The architecture of a system must be specified carefully if it is to achieve the overall performance as well as address the overall goals of the program. The architecture for the AVLIS design had to address the schedule requirement for the different phases leading to an operating plant in the 1990s. This, along with the need to assure long experimental runs that can survive the inevitable failures of the supervisory system, led to the architecture described here. The concepts used to develop this architecture include modularity, independence, and integration. The decoupling of modules to form a logically consistent structure dictates that the architecture of the system be such that internodal communication is minimized, while availability requirements dictate that the speed of response to human and process interactions be in a range commensurate with their respective needs. This section describes the issues involved in designing an architecture to perform the functions required.

3.1 Hardware Issues

Hardware architectures are usually described in terms of the level of distribution of the process hardware. The two extremes discussed here are the completely centralized and the totally distributed control architectures. The range of options between these two extremes is also discussed.

The system architecture must lead to an integrated design. Since the sensors that interface to the process are widely distributed and the data obtained must be displayed to a single operator, it is necessary to concentrate these data and provide an efficient way to transport and display them to the operator. In the area of control, the operator and the experimenter need to be able to manipulate the process parameters from a central location. Unlike industrial systems, where the process is very well understood and the operators study manuals to learn its proper operation, our facilities are run by sophisticated operators who know the process better than do their industrial counterparts. We must therefore provide the operator and the experimenter with extensive capability in the control room. If the process is well understood, the cost of designing for unknowns may not be cost effective, but processes where significant unknowns exist need built-in flexibility.

Some alternative architectures for process control and data acquisition systems (PCDAS) include both highly distributed and highly centralized designs. One example is a hostless system available from Westinghouse (and others) in which all control and data acquisition is done at the most distributed level and the operator is supplied with data from the various nodes upon demand. This architecture relies on high-speed communication between the nodes and the operator. Even so, this high bandwidth path may not be fast enough. The operator may become frustrated and distracted waiting for the display to be updated. One of the key considerations in a

distributed system is how fast new data from the process reaches the operator. In steady-state processes, the time involved to update the operator's display presents little cause for concern for overall plant safety. It is, however, very frustrating and distracting to the operator to have to wait several seconds for the display to be updated. My experience indicates that from a human factors standpoint it is better to display data that are a few seconds old than to force the operator to look at a blank screen while waiting for current data.

Other widely distributed systems suffer from the same disadvantages as the hostless design. These do offer high reliability, since a single failure will not usually cause a significant problem in the overall process. If several of the distributed nodes must communicate, however, in order to perform their required function, a failure in any one of them may bring down the group if the design does not take the interdependence into account. Also, the communication line itself could fail, introducing a new failure mode. The highly distributed system, therefore, is not recommended for the large experimental facilities being considered here.

The opposite architecture, consisting of a large central host, is also not recommended for the application under discussion here. Three of the heaviest loads on computers in process control systems are process interface, user interface, and interprocessor communications. By centralizing everything in one large host, we completely eliminate the need for the interprocessor communication, but to

expect one processor to handle both the process input/output and the data display is quite optimistic. In this type of design the operator is generally given the lowest priority; consequently, as the process requirements expand, he gets slower and slower responses. The usual solution is to get a larger processor for the entire system. This design also has the disadvantage of having a single-point failure mode. If the main computer goes down, all intelligence in the system is lost. For these reasons, a single computer implementation is not recommended.

The most workable systems are found in the spectrum of architectures between the highly distributed system and the single host system. Using supervisory control concepts and supplying setpoints to the local process controllers provide reliability and ease the speed and communication requirements of the processors. Supplying intelligence and programmability at a low enough level we reduce the cost and time requirements for modifications. The subsystem concept lends itself very well to this architecture. By defining virtually autonomous subsystems, we can granularize the system to a logical level without requiring extensive communication to perform the subsystem's normal function. This architecture also provides the needed extendability and flexibility; as subsystems grow, we can allocate larger and larger processors to them. Also, changes within and among subsystems can be accommodated by rearranging the inputs to the low-level systems.

This supervisory distributed architecture requires dedicated controllers at the process level to avoid having the general purpose computers involved with the fast loops sometimes required. (The characteristics of these dedicated controllers are important in the implementation of the ICAM system and will be more fully described later.) Another implication of this architecture is the availability of data for the operator and the experimenters. The use of a processor whose main function is to supply the operator with current information greatly improves the probability that the data will be available when needed.

Figures 1-3 illustrate the three concepts of highly distributed, host-centered, and modular design.

3.1.1 Distributed Intelligence

Distributed intelligence implies the availability of computer power at various levels in the architecture, permitting the offloading of some operations necessary for control of the process. The use of distributed intelligence offers significant promise in data acquisition and control systems. It offers significant challenges as well. Distributed process control systems in particular use the concept of distributed intelligence extensively. The key to using it intelligently is to include enough capability to minimize the requirements for external communication but not so much as to disproportionately increase the software development effort required to implement the function in the low-end device. Larger systems generally offer better tools for software development. The use of

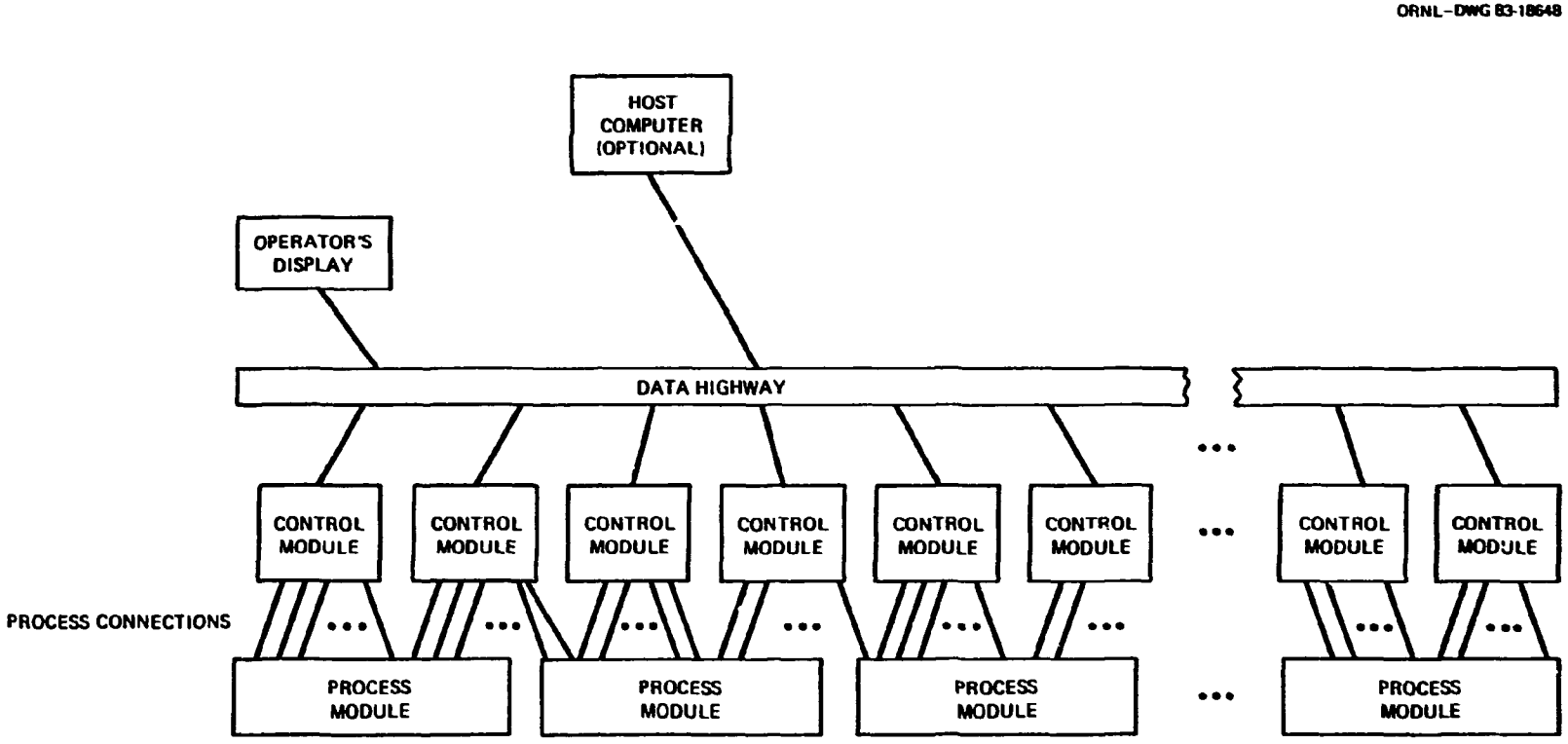


Figure 1. Highly distributed process control system.

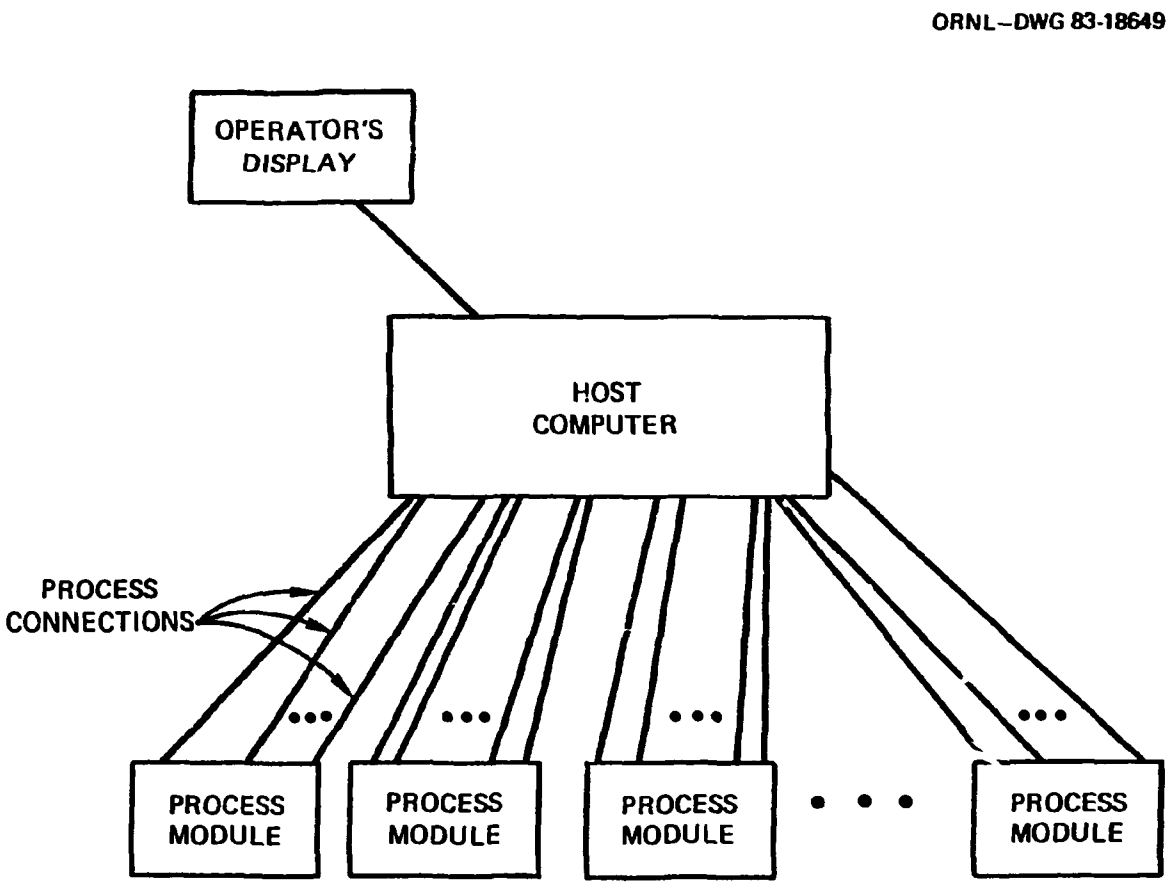


Figure 2. Central-host-based process control system.

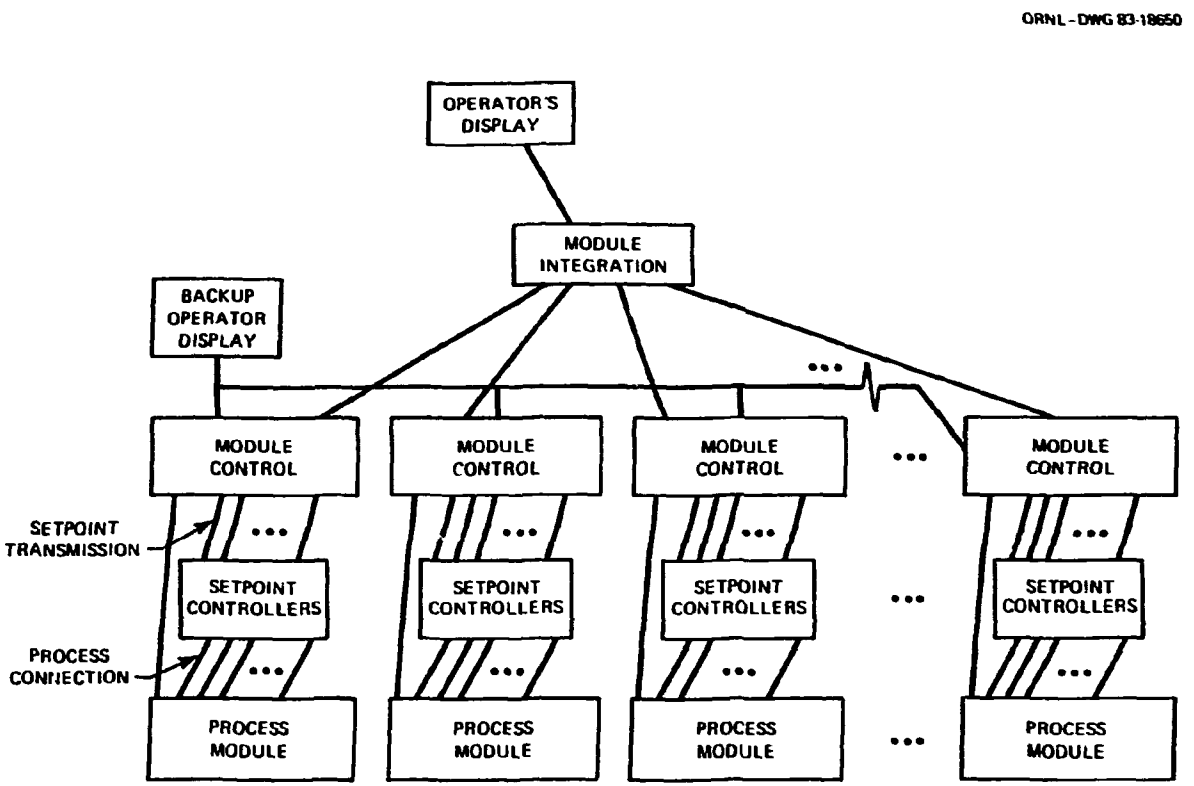


Figure 3. Hierarchical supervisory control system.

distributed computing concepts in process control is well documented.[4,5]

Although the use of programmable controllers (such as the Modicon 584) as intelligent nodes has made the automation of sequential operations much simpler and more reliable, the ease of programming offered by these controllers has also led to problems. In the past, a wiring change was needed to modify the program in a relay logic panel. This was usually done by one individual, and it could be controlled administratively; no one could modify the wiring without a change notice and proper signatures. The advent of programmable controllers, however, has made program changes so easy that a large number of people make changes with little notice from others. Such changes are very difficult to spot, whereas the wiring changes were usually flagged by signs and tags. The documentation of the software describing a change is usually not tied to a knob or wire, and is therefore not as visible to operations personnel as a piece of cardboard hanging on an instrument panel.

Our solution to this problem is to program the device from a host. This gives password control as well as a means of verifying changes. Periodically the program in the controller is checked against the "authorized" program stored in the host. Any differences are flagged and can be rectified.

Most top-of-the-line programmable controllers offer a host interface package which usually includes hardware and software. Some problems occur with these systems. Because dealing with

software is new to many of the vendors, some of them may not provide the level of software support needed to use the system in a process control environment. The whole issue of networking is also relatively new to the vendors. The asynchronous interface to the host is one indication of their lack of experience with host interfaces. The RS-232 interface usually provided is slow and imposes a heavy load on most host processors.

In addition to the documentation and speed issues noted above, another capability that makes the host interface valuable is the integration of real-time data from the unit into the overall instrument system. This also makes possible the support needed to answer operator queries about the state of internal permissives that indicate the state of certain limit switches and other devices that inhibit actions potentially dangerous to equipment or personnel. It is relatively simple to implement permissives in the controller, but quite difficult to make known to the operator which permissive is not made up when a request is denied. With the host interface and the ladder logic from the controller available, software can be written to check the permissives when an operator request is denied.

An important characteristic of a programmable controller is how the data acquisition scan is accomplished with respect to the solving of the internal ladder logic. Most units perform the data acquisition scan asynchronously. That is, they could be solving the tenth ladder network based on information that is current, but a scan is done reflecting new states before the eleventh network is solved. In

many processes this procedure could cause serious problems. The units that are specified in this program use a synchronous scanning technique. That is, they perform a data acquisition scan only at the end of the ladder logic solution. The entire logic is therefore solved with consistent data.

In hardware architecture reliability is addressed in a number of ways. A key concept used to address reliability and availability is redundancy but in the AVLIS program the concept is to address reliability, availability, and maintainability issues as a system problem. Making the most reliable part of the system redundant does not improve overall system availability. Although many vendors supply systems where redundancy is advertised to improve system availability, this is true only if the least reliable parts of the system are made redundant. The AVLIS design takes the approach that complex systems are virtually doomed to fail eventually and we therefore take steps to mitigate the consequences of system failure. Specifically, the ICAM architecture provides sufficient intelligence at a level low enough so that a failure in the supervisory portion or in a communications channel will not cause the system to shut down immediately. The goal is to have the local instrumentation provide sufficient capability for several minutes of operation without the higher levels. This capability requires more complex controllers in some cases, and the controllers will be described in detail in a later section.

Another concept used in the AVLIS design is to provide an architecture that does not restrict the choice of process hardware. This allows the instrument engineers to choose those tools that can best do the job and does not force the selection of tools that may be inadequate but must be used simply because of a system restriction. Random selection of a wide variety of tools can be counterproductive, however, so the AVLIS design specifies recommended instruments and instrument interface techniques to encourage standardization. Odd devices and interfaces can be accommodated, however, without prohibitive cost.

A final concept in the AVLIS design addresses economic as well as reliability issues. The implementation plan proposed for the program permits the phasing in of the expansion necessary for the larger facilities. Each new facility provides a test site for the next level of capability. Figure 4 shows how the replication of lower levels developed on smaller facilities can be fitted together to produce a viable system for a large facility. The photographs presented in Figures 5-11 illustrate the difference in scale between the two facilities--EB-I (electron beam) and the Materials Handling Development Module (MHDM)--that exemplify the first two stages of development.

3.2 Software Architecture

A key feature in the architecture of this system is the architecture of the software. To maintain flexibility and minimize

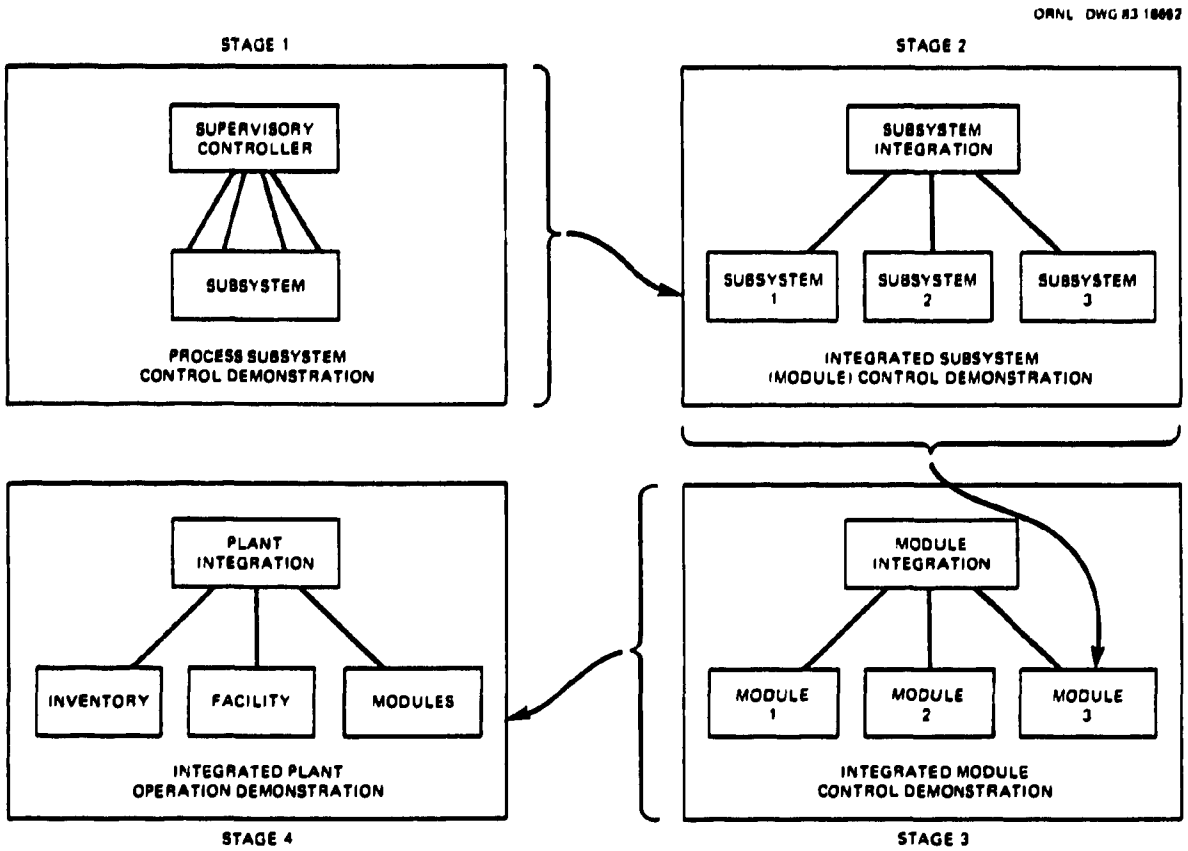


Figure 4. Phased implementation of the ICAM system from prototype to a plant and scale facility. Each stage represents a facility designed to demonstrate a capability needed to operate a plant.

Figure 5. EB-I local instrument racks.

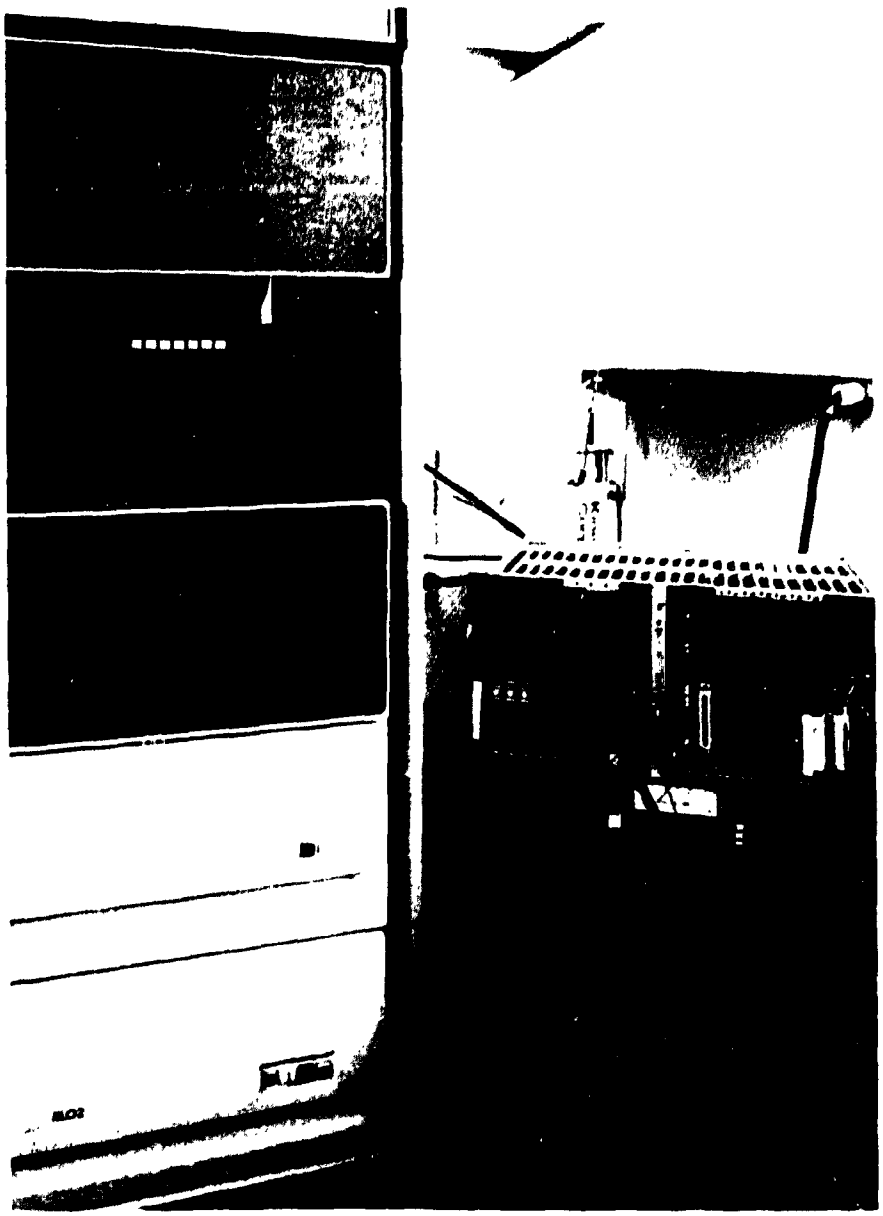


Figure 6. EB-I supervisory computer.

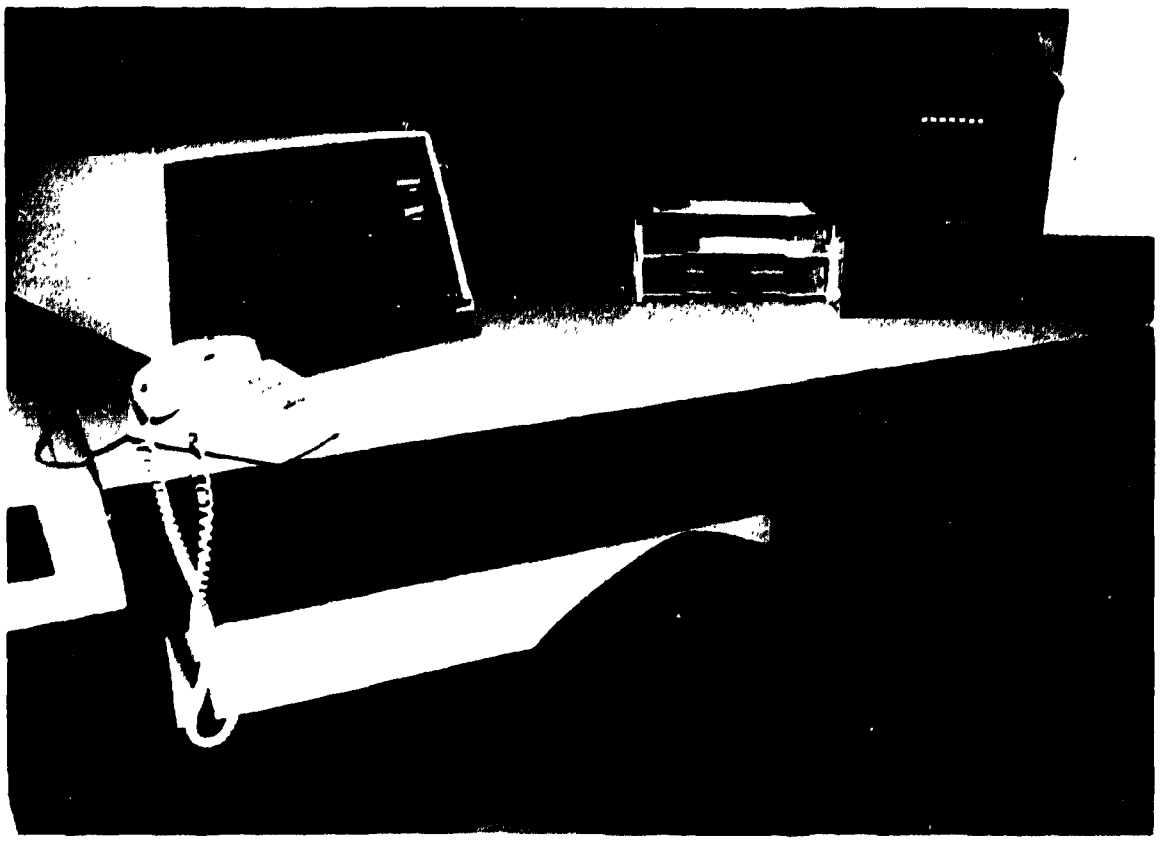


Figure 7. EB-I operator's console.

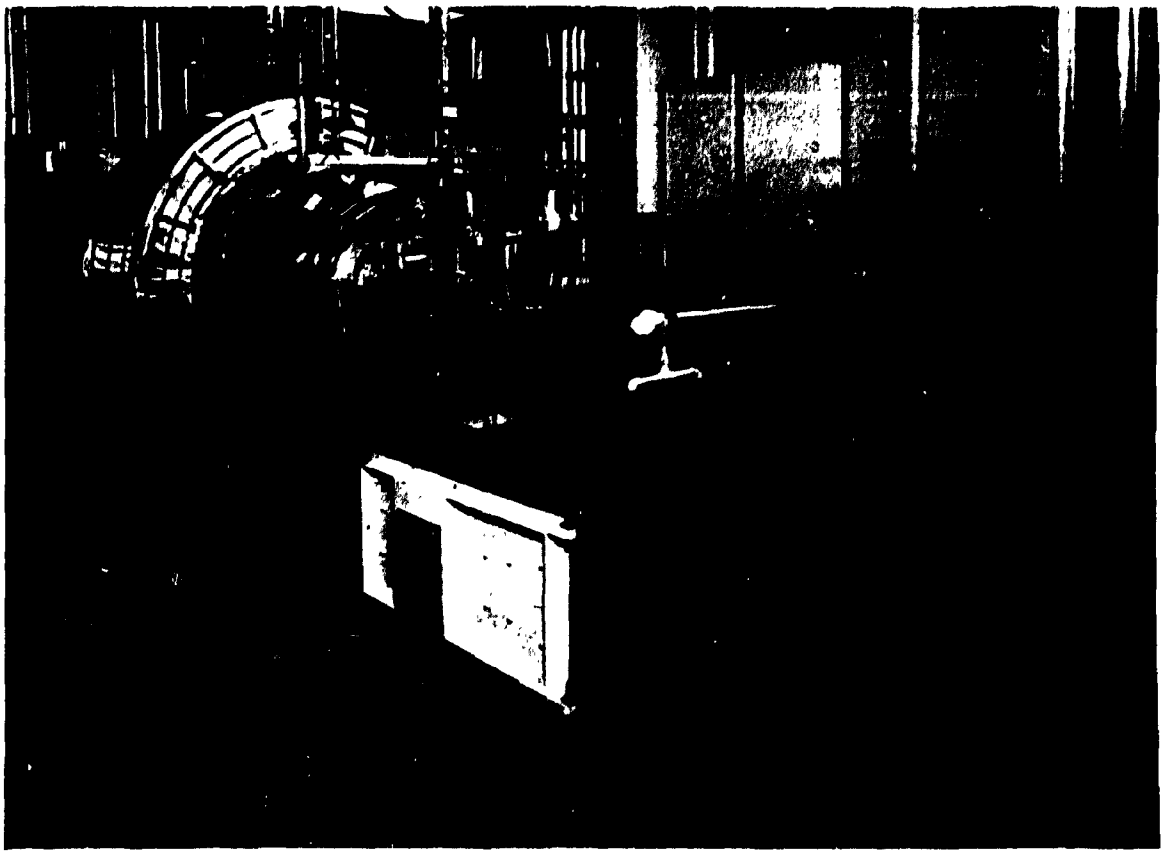


Figure 8. EB-I vessel.

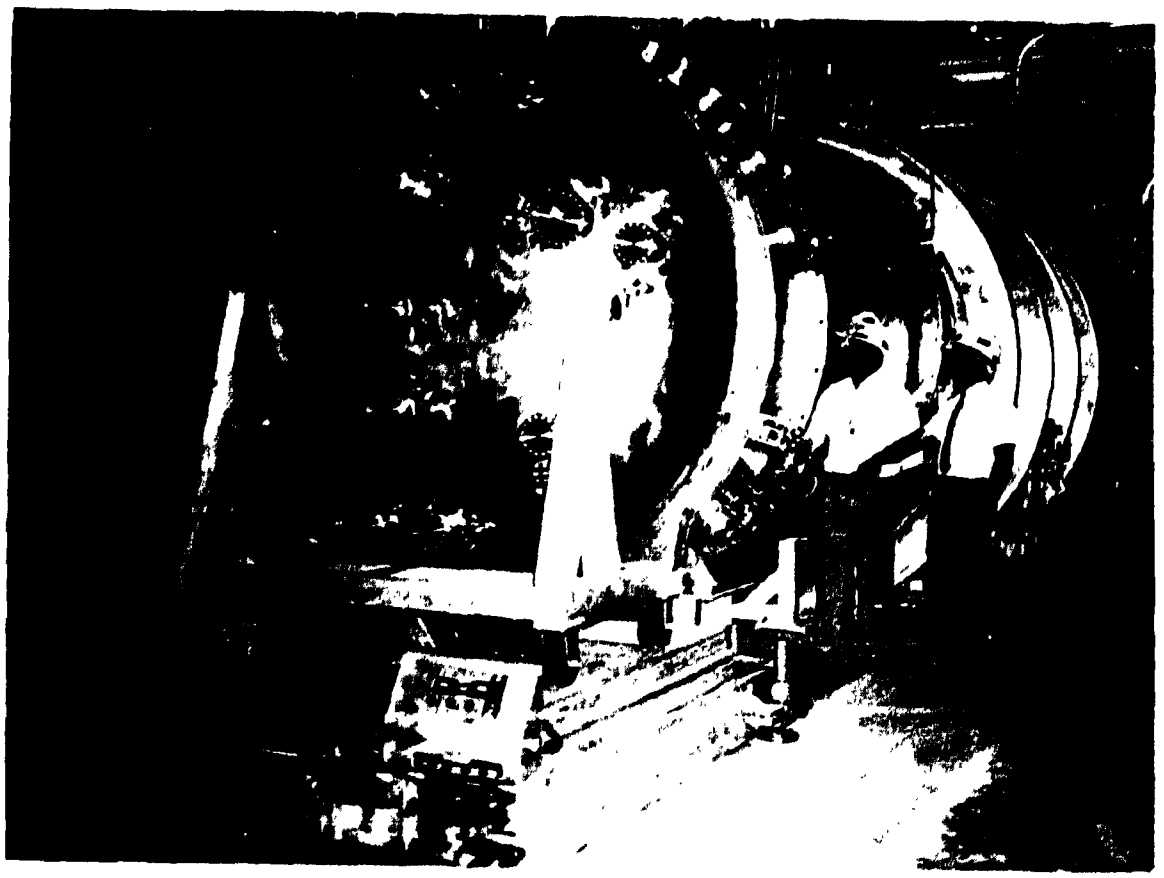


Figure 9. MHDM vessel.

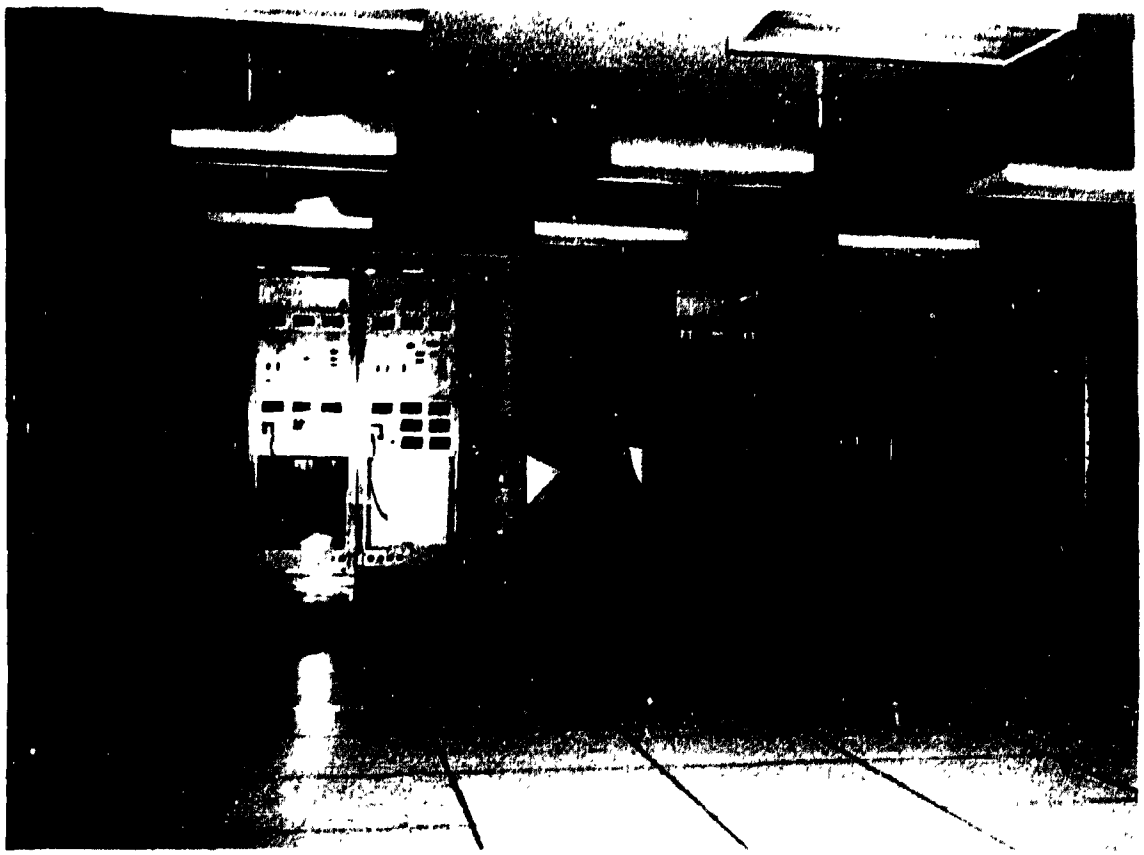


Figure 10. MHDM local instrument racks.

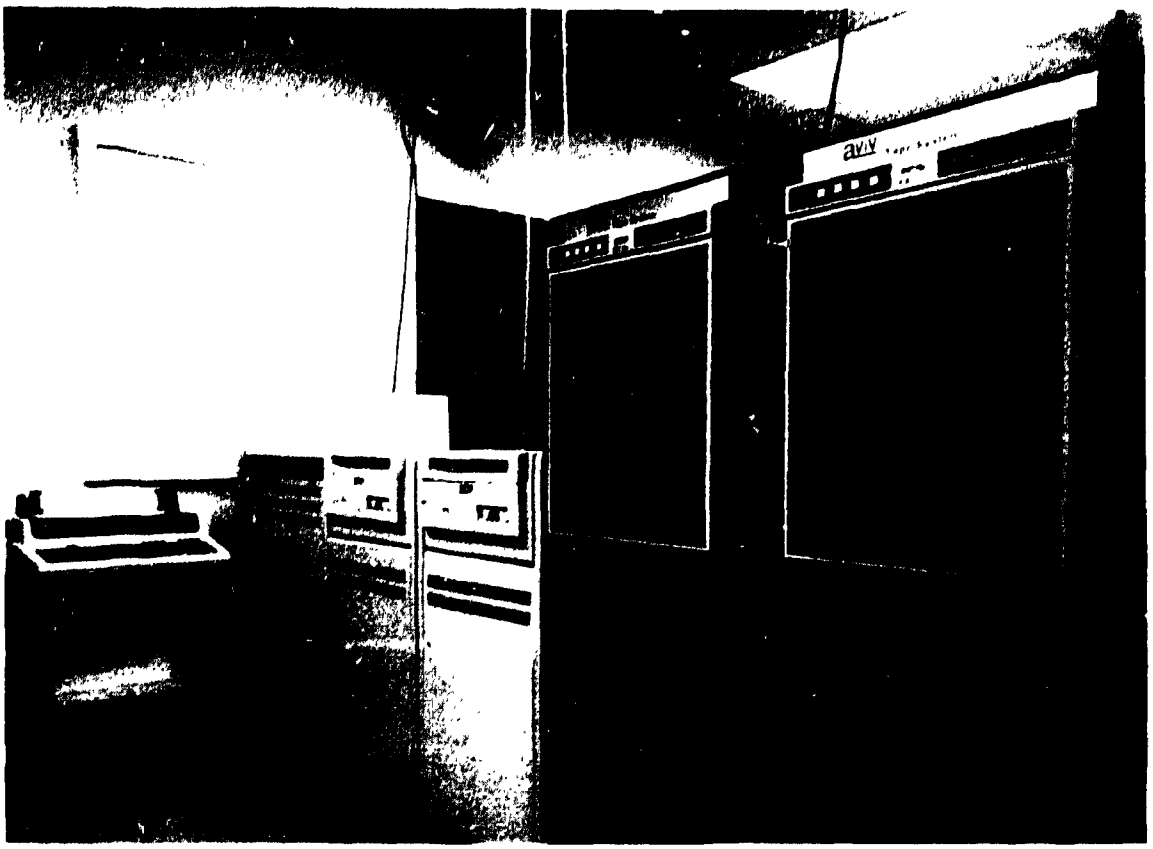


Figure 11. MHDM data concentration controller -- lowest level supervisory control computer.

communication at all levels, we maintain appropriate data structures at each level in the architecture. These data structures must be dynamically modifiable without perturbing either the process or the acquisition of data from the process. The use of a data base at the various levels permits autonomous operation, with communication necessary only when a change is required or detected. By maintaining a live data base containing real-time data at the user interface level, we can display the real-time data at an update rate that does not encourage distraction or frustration.

An alternative to this type of software design is known as "scan and dump," a technique which forces any program that needs data to interrogate the process interface directly in order to obtain it, thereby tying all display and computation to the speed of the process interface. This offers the advantage that the data obtained can be highly time correlated. Since the AVLIS process is essentially steady state, the design selected was not the scan on demand but rather a live data base containing representative data values, a technique which assumes that the time constants of the process are long compared to the time required to obtain the data. This is not completely true for all aspects in the AVLIS process, but exceptional cases can be accommodated using transient digitizers. This design allows the users to obtain representative data quickly while not tying data acquisition of all channels to the time constants of the fastest portions of the process.

The data base design used in AVLIS provides the independence and speed necessary to adequately address the needs of the process. Reliability is enhanced, ease of development and maintenance is improved, and on-line modification of the data base gives the flexibility to help assure success in experimental runs of extended length. Some vendors are beginning to tout data-base-driven systems. Among these are Kinetic Systems Corporation's LION, Electronics Modules Corporation's EMCON-D3, and Quadrex Corporation's FLIC. Some of the issues discussed in this paper are reflected in these firms' designs. As their designs mature, they may become more useful in developmental facilities.

3.2.1 Data Base Design

If we assume that the process being instrumented is expected to operate for extended periods of time without significant interruption we must provide the capabilities to modify operation parameters. We can do this without a data base, but it would not have the ability to obtain accurate information about the process without interrogating the individual process interface components. The data base concept also supports the modularity required to produce a viable system design. By maintaining in memory a table of the important parameters of the instrumentation system, we can introduce new functionality to the system by adding new hardware and software modules. The common area provides the communication among modules required for cascade control as well as decoupling the functions of the modules. The

diagram in Figure 12 illustrates the modularity concepts as implemented in the AVLIS ICAM system.

Although maintaining a global common area for communication among modules is considered poor practice under the goals of structured programming,[6] the alternatives are even less attractive. Without a data base through which to communicate, each module would have to be responsible for obtaining and displaying the data for which it is responsible. The modularity permitted by decoupling data acquisition from data manipulation and display makes the use of the on-line data base an overall asset to the maintainability of the system. The common area serves primarily as a "data store" rather than a communications mechanism.

Notice the building block structure of this design as shown in Figure 12. If a new process input/output device is added, it is transparent to the modules responsible for data display and manipulation. Similarly, the data display and data manipulation modules do not need to be concerned with the source of the data. This decoupling of the data sources and data sinks permits greater flexibility and expandability than does a complete top-to-bottom architecture where each module must be concerned with source/sink interaction. Application design becomes designing up to the data base interface from the transducer and designing down to the data base interface for data display and manipulation modules.

The selection of parameters for the data base is based on experience and knowledge of the process being instrumented. Our

ORNL-DWG 83-18651

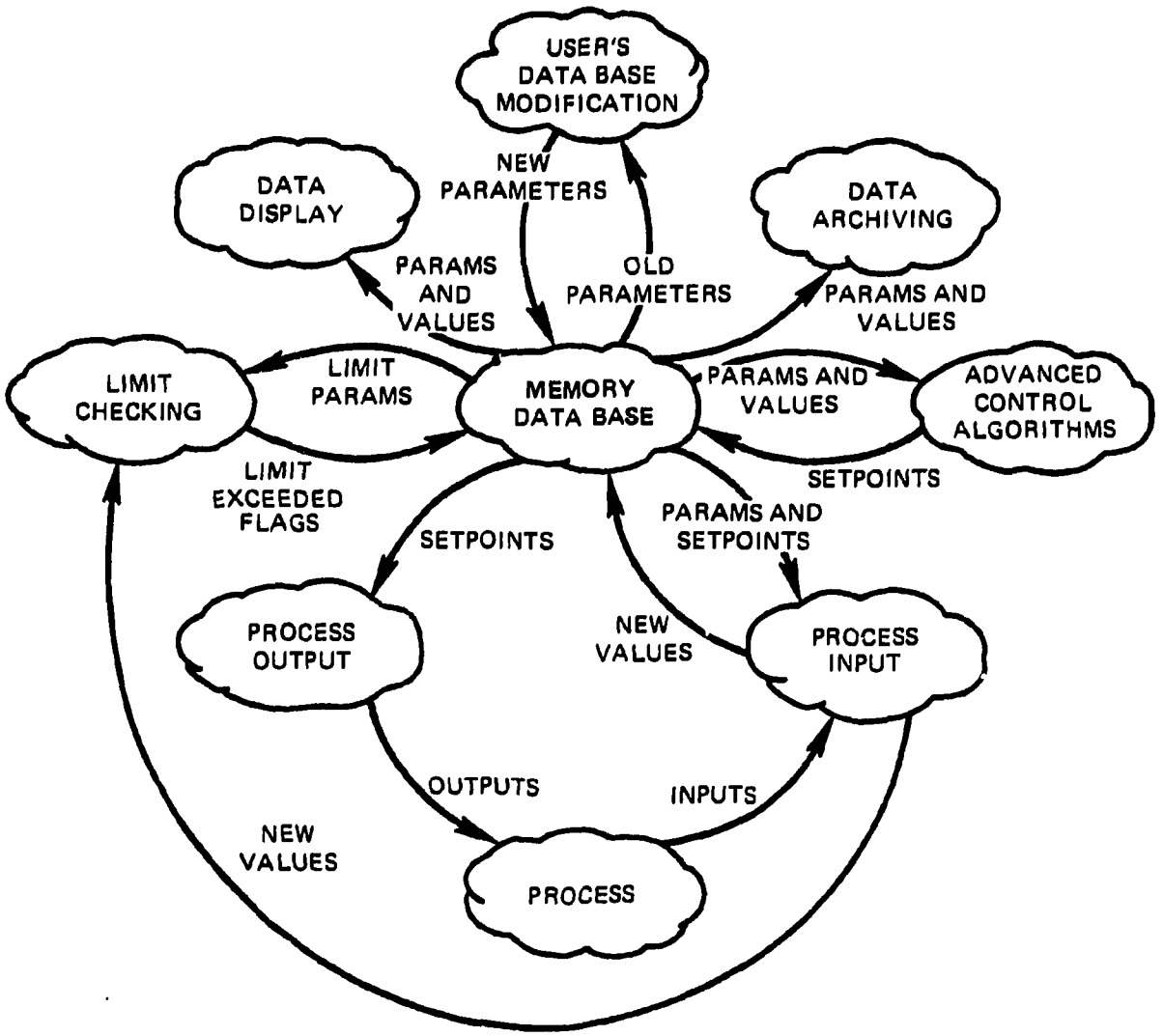


Figure 12. Data base architecture diagram illustrating modularity.

experience has shown that the best way to design such a system is to understand the process as thoroughly as possible. It is more effective to have the instrumentation team learn the process than to have the process scientists learn about instrumentation systems.

The data base is the keystone of the integrated system. Careful design is important and time consuming. Commercial data base management systems (DBMS) generally are not suitable for use in a real-time environment, primarily because their speed and reliability are suspect in the control of complex experiments. Also, a DBMS is much more complex than is necessary for the type of access required in process systems. However, since our data base is also maintained on disk for backup and non-real-time requirements, we are considering the use of a commercially available package for accessing it.

It is crucial to maintain the integrity of the on-line data base to ensure data availability. A resource management technique and archiving of on-line modifications are used to help ensure this. However, care must be taken when accessing the system data base. There are times when the data in the data structures are not consistent, and any access during these times will yield inconsistent data. We use a semaphore protocol that guards against these problems to ensure against this. One alternative is to declare the entire data base a resource and lock it whenever any user requests access. This would cause problems, though, since nearly every program in the system needs access to the memory data base and many need the access in real time. A better approach is to break up the resource at a

level low enough to make contention unlikely. At this level of granularity it is important that the mechanism be fast and efficient.

Note that contention problems exist only when a program is writing into the data base. If all accesses were read-only, resource management wouldn't be needed. This implies that we can allow shared read accesses but all reads must be locked out when a write is occurring. The writer, of course, cannot proceed until all readers have finished. This brings up the concepts of priority access and queuing of requests. All of these techniques can be used to control access to the data base, but we chose a simpler technique for the base level of access control in the AVLIS ICAM design.

We divided the data base into three resources--the name table, the scan table, and the name indexed tables. This division is consistent with the design because access to the name table and scan table are not indexed but rather are searched in the design. The indexed tables, however, permit each individual point's parameters to be declared a resource, which permits users to access individual points without interfering with other users. The name and scan tables are handled differently than the indexed tables. Since users need read access quite often to the name and scan tables, shared read access to them is provided. Any writer must declare his intentions and wait for all readers to finish, and no new readers are permitted once a writer has declared. This type of resource management has much less overhead on each access than techniques using queuing and

prioritization. It was determined that queuing and prioritization were not justified for this implementation.

For individual point accesses into the indexed tables, however, we wanted faster access with less overhead. Since writing into the indexed tables is common, all accesses were forced to be exclusive. This makes reading a bit more troublesome, but the data acquisition tasks (which run in real time) reap the benefits of the reduced overhead. This technique also reduces memory requirements because it is not necessary to keep track of the number of readers and determine whether all readers have finished.

Deadlock, one of the classic problems in resource management, can occur when a single program requires access to more than one resource at a time. For example, a deadlock occurs if two users need the same two resources to complete their work and they each allocate one resource and wait for the other one. The problem is that neither one will ever finish because each is waiting for the other's resource to be released. For a more detailed description of deadlock and some proposed solutions, see the ORNL/TM report by R. W. Hayes.[7] The simplest way to avoid deadlock is to require each resource user to allocate all resources needed in the beginning, to allocate and de-allocate them in a prescribed order, and, if any resource is not available, to release all held resources and wait before trying again. (This is known as rollback.) Since most real-time tasks do not need access to more than one resource at a time, this limitation is a problem only for the non-real-time portions.

Because it is important to keep access simple for the real-time tasks, we implemented a simple administrative procedure rather than one using prioritization and queuing. Note that this simple technique does not assure that the highest priority access will be granted first; it is merely a first-come, first-served technique.

In commercially available process control systems that do not have a data base, the major problem of resource management is in the hardware from which the data are obtained. Systems that do maintain an on-line data base are not usually concerned with the problem of resource management if the worst thing that can happen is a small perturbation in the control of the process. Since such systems are in general not concerned with the implications of displaying or storing bad data, the inconsistency in the data base is so transitory as to be of little or no concern. However, our concern for the long-term value of data in the AVLIS system requires that we do as much as possible to guarantee its consistency.

3.3 Implementation of the Architecture

In the AVLIS Program the ultimate goal is the instrumentation of a plant-scale facility. Since the overall plan called for several intermediate facilities before the full-scale plant, our task was to prototype various portions of the proposed plant system on the smaller scale facilities. This presented the opportunity to test some early concepts and designs. The approach has been to prototype the lower levels of the architecture first and then introduce a new

layer in each succeeding facility. This technique assures us a working base on which to build at each stage, and the approach approximates what is known as top-down design with bottom-up implementation. The concept of "evolution rather than revolution" to help ensure success in large computer projects is documented in the literature.[8] We emphasized the evolutionary approach throughout this design effort.

4. PROCESS INTERFACE

This section describes the concepts and actions necessary to design a process interface that will produce an integrated control and measurement system.

4.1 Hardware Considerations

The interface to the process must be rugged and reliable. The speed may also be important in some systems. Many complex experiments involve conditions that introduce electromagnetic interference (EMI) into the signals from which the data are obtained. It is critical that the installed instrumentation system not only survive in this environment, but perform within specification as well. [A number of vendors quote the environmental conditions under which their product experiences catastrophic failure, but they do not state the conditions that will cause it to fail to perform within specification. A classic example of this type of "specmanship" is the maximum common mode voltage rejection (CMVR) rating of an amplifier/multiplexer system: Unless the vendor includes a maximum operating common mode voltage, the engineer must assume that the device will not operate at any voltage where the sum of the common mode and the normal mode voltage is greater than the maximum normal mode range of the device.]

Other important specifications include common mode noise rejection (CMNR) and normal mode noise rejection (NMNR), which are measures of how well a device will function in the presence of unwanted signals on the input leads. Low-level signals can be completely swamped by noise; it is therefore important to use devices whose specifications in this area match the environment in which they are used.

Three conditions--common mode voltage (CMV), transients, and temperature excursions--often cause problems (errors or failure) in process instrumentation hardware. This behavior can be characterized by either catastrophic failure or simply operation outside the quoted specifications of the equipment. In this section I will describe some of the critical specifications the instrument engineer should investigate before choosing instrumentation hardware. Again, these remarks are not directed to the person who designs the instrumentation components, but rather to the person responsible for the overall integration of the instrumentation system.

An important implication here is the extensive communication required among participating members in any group responsible for the development of an integrated system. Too often the person responsible for the computer systems is not involved in the overall instrumentation development. This is a very dangerous situation if the overall goal is to produce an integrated system. Such a situation emphasizes the need for the initial philosophy discussions stressed earlier.

Ungrounded transducers or ground loops introduced in the wiring can cause seriously high common mode voltages. Although grounding the transducers can be used to help suppress high common mode voltages, this solution is sometimes not possible or not adequate. To accommodate these voltages, a stage of isolation is used in the front end of the instrumentation equipment. This may take the form of optical, transformer, or capacitive coupling. (I will discuss these alternatives in more detail in the next section.) Since isolation techniques are expensive to implement, an instrumentation system usually consists of one (or a few) high-quality isolation amplifiers with a multiplexer to use the same amplifier for a relatively large number of channels. (The next section will cover the use and specification of multiplexers.)

Transient suppression can be accomplished with solid state clamps of some kind. Transorbs (R) or another brand of metal oxide varistor (MOV) can be specified. The engineer must be certain, however, that they are fast enough to protect the circuitry and that they do not clamp at a common mode voltage where operation of the system is required. It is also important that these clamps be installed properly. They should be installed from each signal lead to instrument ground because other connections do not adequately protect the input amplifier. Transients in the system can cause either catastrophic failure or simple inaccuracies. Inaccuracies caused by transients are usually easy to spot because they show up as wildly varying signals, often beyond the physical capability of the

transducer in the circuit. Note that transients can cause both common mode and normal mode noise. The normal mode noise rejection (NMNR) figures in the specification are helpful in estimating the effects of transients here. Normal mode noise rejection implies that the noise is at a much higher frequency than any time constants in the system. If this is not the case, normal mode filters cannot be used. The errors introduced by common mode noise are caused by its conversion to normal mode noise in the amplifier. This is well documented in several good articles on instrumentation grounding and noise problems. (A booklet published by Acurex Corporation and authored by James H. Chandler entitled "How to Make Accurate Low-Level Measurements" is a good source.[9]) Note that fused inputs are not of much help from a maintenance standpoint. Whether a fuse or an input transistor, the device must be removed for replacement. The only advantage may be the low cost of the replacement part, but this is easily swamped by the high cost of the labor to replace it.

Problems caused by temperature excursions can be more difficult to detect and correct than those caused by transients. Such excursions can cause catastrophic failures, but more often they simply degrade the quality of the data obtained. Temperature cycling does often cause catastrophic failures. For this reason equipment should not be powered down unnecessarily. In general, it is much better for the equipment to be left on at all times if it has no moving parts to wear out. One approach to the handling of inaccuracies caused by temperature drift is to control the environment in which the

equipment is located, while another approach is to specify that the equipment meet military specifications for temperature. However, this can be expensive at best and would severely restrict the functionality available, because many companies' products do not meet these requirements. The engineer must be careful in estimating the overall system accuracy and include temperature drifts in the calculation, which may be a problem because this specification can be difficult to find or use.

As mentioned previously, the cost of a high-quality input amplifier usually precludes the use of an amplifier on each channel. Such amplifier per channel designs are expensive, but they are used in cases where speed and isolation are both considerations. If super high speed or high isolation is not required, multiplexer input systems are used to reduce the cost of the input system. This section includes a discussion of some of the more common multiplexer systems available.

Multiplexer systems must incorporate some method of isolation to prevent common mode voltage from damaging the amplifier and the computer circuits. Possible techniques include relay and solid state switched multiplexers and isolation using optical and transformer coupling. Where isolation is not an issue and signal levels are not in the millivolt range, solid state (transistor) switches can be used ahead of the input amplifier. The speed of these input systems is their most attractive feature. Some can read inputs at a rate of 200,000 samples/s fairly easily. By preceding the solid state

multiplexer with some other method of isolation, they can be also used where isolated systems are required.

Some problems do exist in the use of solid state switched systems, and although this type of input system is most common, it must be well understood before it can be applied successfully. Solid state switches cannot be used in systems where high isolation is required. Also, current technology limits switching transistors to about 35 V maximum input, and when power is removed from these input circuits their input impedance drops to near zero. Since all of the signal must pass directly through the transistor, it is important to have low "on" resistance. Some solid state multiplexers cannot be used for low-level signals because the noise in these devices is enough to cause measurement errors. These significant problems limit the applicability of the solid state multiplexer systems. The applications where they are used successfully consist exclusively of high-level, grounded transducers where high speed is required.

There are three major types of relay-switched multiplexers, the simple relay input switching, the drive guard type, and the more expensive flying capacitor switching. In the relay input technique, the signal from the transducer passes through a relay to the amplifier input. By opening and closing different relays, different transducers can be connected to the amplifier. Normally these systems force the input amplifier to be at the same ground potential as the transducer being scanned. This is necessary to achieve reasonable common mode noise rejection. Some systems of this type use

three wire leads and switch the guard with the signal leads. Another technique, known as driven guard, involves an amplifier that keeps the quard lead at the same potential as the signal, thus improving noise rejection. Figure 13 shows the three types of relay-switched multiplexers. Figure 13a shows a schematic diagram for the relay switched, and Figure 13b shows the flying capacitor multiplexer. Shown in Figure 13c is the driven guard configuration which, with a small sacrifice in common mode noise rejection, requires only two wires being switched.

The flying capacitor technique never lets the transducer come into electrical contact with the amplifier. Isolation is obtained by switching a capacitor from the transducer to the amplifier. In this scheme the relays are used to switch a capacitor alternately to the transducer and to the amplifier, a technique which offers the advantage of true isolation since the transducer and the amplifier do not have to be at the same ground potential. It is truly a differential input device, and only two wires from each transducer need be attached to the amplifier.

Relays, however, are electromechanical and have certain inherent weaknesses; among these are speed and life expectancy. Since the flying capacitor multiplexer must be preceded by a filter to prevent high input currents on time-varying signals, it is even slower than the straight relay input systems. Speeds as high as 1000 channels/s can be achieved with well-designed flying capacitor systems, but the input filter restricts the reading of the same channel to less than

ORNL-DWG 83-18663

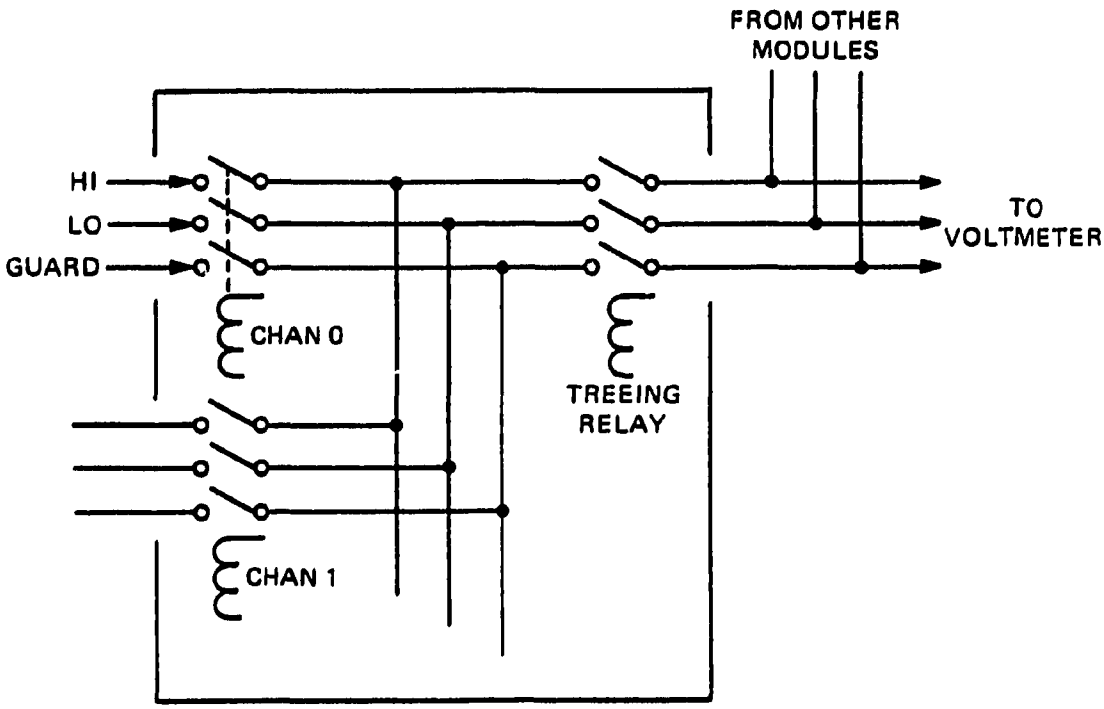


Figure 13a. Reed relay switching module.

ORNL-DWG 83-19396

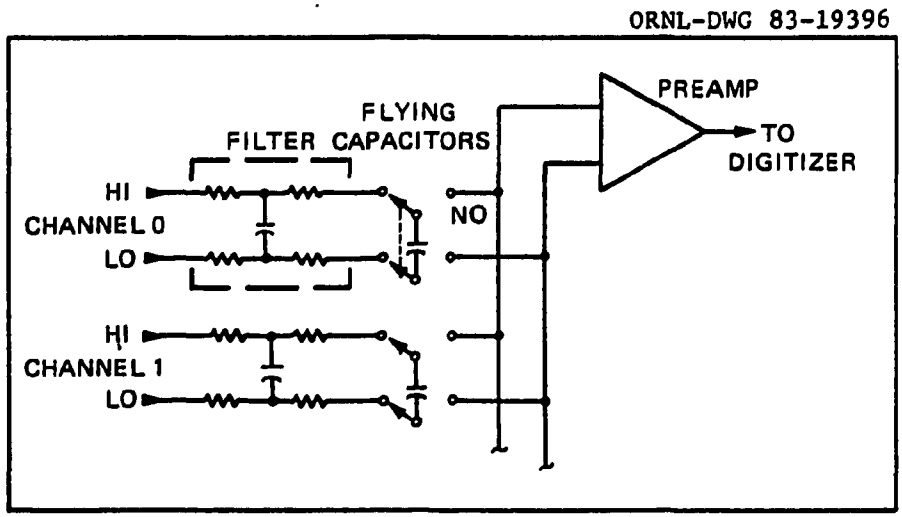


Figure 13b. Flying capacitor multiplexer.

Figure 13. Three types of relay-switched multiplexers.

Figure 13. (continued)

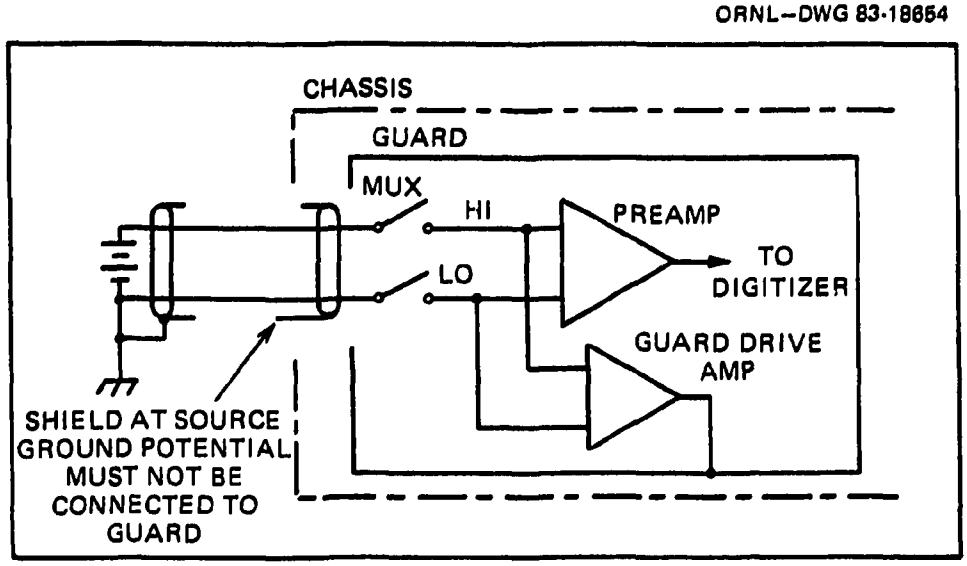


Figure 13c. Driven guard input system.

20 samples/s. Straight relay input systems can reach somewhat higher speeds, and they have no restriction on reading the same channel. Life expectancies of relays appear to be improving, currently in the millions of operations (about three years), but their failure modes can cause obscure symptoms. A failed relay opening too slowly can cause a completely different channel to be read incorrectly.

The use of isolation techniques can permit the use of less expensive analog equipment. Such techniques involve the conversion of the signal to ac and coupling through either a transformer or an optical fiber to isolate the ground potentials between the amplifier and the transducer. The first stage of amplification must still withstand the full voltage, however, and devices that accomplish this without introducing errors that impact the measurement for low level signals are not easily found. These types of isolation techniques are beginning to be used in integrated circuit isolation amplifiers. They provide high isolation where an amplifier per channel is acceptable and the higher performance is needed. Some applications have channels with as much as 2000 V of operating common mode voltage. These devices represent a viable solution, but their temperature stability may be a problem.

Note that although most emphasis is placed on input systems, the same problems of noise on the signal leads can cause failures in the output circuitry. A transient induced on an analog output lead can cause a failure of the output circuit. The important considerations

are output impedance and common mode voltage rejection. Isolation is therefore important for output signals as well as input signals.

The current implementation of the integrated control and measurement (ICAM) system uses flying capacitor multiplexer channels for most of the analog input signals, with some amplifier per channel on high-speed channels and a high-voltage isolation amplifier on some high CMV channels. Temperature drift has been our most persistent problem to date in all input types.

4.2 Overall System Errors

Measurement inaccuracies can be caused by temperature variations, calibration errors, software errors, common mode and normal mode noise, and digitization errors. Digitization errors are caused by the fact that all digital equipment is of finite resolution, and they are relatively easy to characterize. The length of the data word in memory is 16 bits, and the length of the word in the analog-to-digital converter is 12 bits. Therefore, the error introduced by using the 12-bit analog-to-digital converter (ADC) is about 0.05% (1/2048) and swamps the data word error (1/32768). Note that a 12-bit ADC uses only 11 bits for analog data, the 12th bit being reserved for the sign of the voltage. Round-off error can also be considered to be a type of discretization error. It depends on the type of calculation required to produce the desired value. This type of error is very sensitive to the manner in which a calculation

is done. Subtracting nearly equal values is a classic way to lose accuracy in a measurement.

Software errors can cause inaccuracies in the system readings, but they are usually of sufficient magnitude that they can be detected by viewing the data in real time. Errors of this type can be caused by resource contention problems in the hardware or by software, or failure to properly account for exception conditions. Again, these errors are difficult to predict and cause serious problems but are easy to spot when they occur. The more serious errors in the system are those that are not detected easily, particularly those so small that they are not easily noticed in real-time displays. Electrical noise and temperature characteristics cause errors of this type.

Errors caused by electrical noise can be transient or steady state in nature. Transient noise can cause easily spotted but annoying errors (spikes) in the data, whereas the small errors introduced by a steady noise signal, either common or normal mode, are more difficult to detect and account for. These are minimized by careful specification of the hardware and attention to detail in the installation procedure. (The same can be said for errors caused by temperature variations.) A technique used frequently in designing systems is known as derating: specification of the equipment to withstand conditions above any expected in the specific application. This approach gives a much wider safety margin and reduces the probability that an unexpected error will show up. In this project derating is

accomplished by specifying the hardware and software for worst-case conditions and then taking steps to ensure that our equipment is never subjected to such conditions. Others have referred to this as "fronting up" (as opposed to backing up) the instrumentation.

Calibration errors can be detected by reading and checking a standard on each block of channels. However, this is a gross check and is not adequate for a quality check because it does not check all channels and all gain/channel combinations. Another approach, known as automatic calibration verification, consists of a programmable voltage source and a precision digital volt meter (DVM) that are interfaced directly to the computer so that it can set any voltage and read it on either the DVM or the analog input system.

A source of error that can be difficult to detect is caused by sharing a transducer among more than one detection instrument. Many instruments are designed to be the only equipment directly attached to the transducer. This can be manifested in an open circuit detection circuit, a non-isolated input circuit, or an input impedance that drops to near zero when its power is removed. Open circuit detection is sometimes done by running current directly through a transducer, which would cause serious errors to any other devices connected to that transducer. Using a 4- to 20- milliamp (mA) transmitter does not always remedy the situation. Many input devices still exhibit zero input impedance when the power is removed, but removal of power from a portion of a system for maintainance is quite common and should not cause the entire system to become inoperative.

Another problem with the 4- to 20- mA current loop is again related to common mode voltage. Many devices assume one side of their input to be at ground, but obviously not every device can be at the end of the loop. The engineer should be alert to these problems and specify equipment without open circuit detection and with isolated inputs that can withstand the maximum voltage produced by the transmitter. Open circuit detection can be performed by the transmitter since it should be the only instrument directly attached to the transducer.

This shared transducer problem is very prevalent in computer based instrumentation systems because they often exist in parallel with a backup instrumentation system. The cost of the backup system is high, and there is usually a tendency to cut corners and buy low-quality hardware for it. This will seriously degrade signals being read by the computer-based system. Backup instrumentation also presents the classic problem of redundant data. Any time the same data are available from two or more different sources there is the chance that they will not agree. In the case of backup instrumentation this is almost certainly the case. One solution is to have the computer read the backup instrumentation meter rather than the transducer. This forces the data to agree, but it also propagates errors from what are often low quality devices. Backup concepts will be discussed in detail later in this thesis.

All of these error sources impact the overall error specification for the instrumentation system. A detailed treatise on error sources is presented in a series of articles in *Control Engineering* (June and July 1980).[10,11]

5. SOFTWARE ISSUES

It has been said that the key to success in an integrated process control system is the software.[12] This, along with the understanding that software development costs probably will be the single most expensive part of an integrated process control system, forces a careful evaluation of software development issues. These include both the development of software and the selection of packages among those available from various vendors. The AVLIS design recommends steps to minimize life-cycle costs for the software used in the program. This section describes the role of software in the AVLIS integrated control and measurement system.

5.1 Multitasking Concepts

Most real-time operating systems permit the concurrent execution of more than one program (task). This is known as multitasking or concurrency. Although it is almost essential in the design of an instrumentation system, some caveats must be observed to preserve the integrity of the overall system. Also, not all real-time systems are multitasking; RT-11 is an example of a real-time operating system that permits only two concurrent operations, designated foreground/background. RSX-11M is a true multitasking operating system; it permits the concurrent execution of more than 200 tasks.

An important characteristic of multitasking systems is how the concurrent operations are handled. The computer itself is executing

only one task at any instant in time; the ability of the operating system to multiplex the use of the processor effectively gives the system its illusion of concurrency. If the only technique used to trigger this multiplexing is time, the system is called time sharing. A more effective technique in real-time systems is one based on a combination of priorities and events. A real-time, multitasking operating system will look for another candidate for the processor when the currently executing task begins an input/output operation or puts itself into any state where it no longer can effectively use the processor. A time-sharing system gives every task the same slice of time without regard to the current state of the process. The most commonly used real-time operating system is probably Digital Equipment Corporation's RSX-11M.

Use of a multitasking system has many advantages, but the user must be careful in developing application software. This is one reason why real-time programming is more difficult than business or scientific applications programming. Simply because one line of code immediately follows another, there is no guarantee they will be executed in succession. Any number of interrupts can occur between two lines of code, and several seconds can elapse between the execution of successive lines of code. However, this is a problem only if there is some form of coupling between the executing software and something in the real world (outside the computer). If something can happen in the real world between the two lines of code, the programmer must realize that the two lines of code will perceive the real

world in two different states. This is sometimes referred to as a "window." Whenever it is essential that two lines of code detect the real world in the same state, the programmer must take steps to ensure that nothing happens to cause the real world condition to change between the execution of the two lines. This usually means running at high priority with the interrupts disabled. This procedure is dangerous, however, and is done only when timing is absolutely critical. Usually a better design would produce software that is not so time critical and that could use operating system utilities to eliminate (or at least minimize) the window.

The AVLIS software requirements dictated the use of proven, well documented real-time operating systems. Digital Equipment Corporation's RSX-11M and VAX/VMS proved to be the best choices for our designs. Other operating systems were considered but did not meet the real-time requirements while still furnishing a good software development environment. We also avoided the languages whose performance in real-time is suspect. We are using FORTRAN and some PASCAL, both of which are supplied and supported by DEC. Wherever we can we use FLEX, a structured preprocessor for FORTRAN, and we use PASCAL for the code to be stored in read-only memory (ROM) in the dedicated process controllers and other stand-alone devices.

In software development for instrument systems such as described here, an important consideration is what type of expertise is needed. Because programming computers is more a logical approach to problem solving than matriculation in a set of required courses, many of the

computer programmers working today come from fields like math or engineering as well as computer science. The computer is actually a tool for use by people working on a problem. Since this tool is still quite cumbersome to use, experts in the operation of tool itself are needed to make it perform its function. As the field matures, there will be a convergence of the expertise of the users of the tool and the expertise required to use the tool. In other words, both the computers and their users are getting smarter. Until this merger occurs, however, either the tool experts have to learn the applications or the application experts have to learn how to use the tool.

New developments in computer hardware and software are important in that they make possible some new architecture. A good example of this is the microprocessor and the non-Von Neumann computer architecture practical. Using parallel processors, computer scientists and engineers are investigating ways to increase computer speeds by hundreds with hardware that is available today.[13]

The amount of computer science background needed to produce an instrument system is usually minimal. The job is more likely to require a background in instruments and systems integration. The current programming techniques that encourage structured design and analysis[14] are important to reduce overall costs in a software project. It is interesting to note that in the early days of software development, the code that optimized computer resources was considered the "elegant" or "efficient" code. Today, the life-cycle

costs of software are the driving goals. This encourages the optimization of human resources over hardware costs.

There are important differences between writing a computer program to solve a specific problem and developing a software system that creates an operating environment. My concern here is the development of an integrated instrument system, and the software must be an integral part of the overall system. Software is developed in much the same manner as any other development occurs. A mechanical engineer may write a program to solve the heat equation in a rectangular bar for his work on some project. After he (or she) obtains the answer, the program may not be run again for a long time (if ever). On the other hand, a software system must be on-line continuously and must respond to external stimuli. The software will be used after the person who wrote the original code has moved on to other activities, and other people will be responsible for maintaining and enhancing it. Some of these issues apply to business systems as well as instrument systems, but I am concerned here with instrument systems where the stimuli originate in a process rather than with people. Such systems generally have few human users but many stimuli from the "real world."

It is interesting to examine the relative difficulties of the various types of programming listed earlier. The Federal Government is very concerned with the cost of software development. Because of this, they have studied the number of lines of code produced per day, which is at least a coarse measure of difficulty. The generally

accepted rate of application programming is ten lines of code per day. For system software, this decreases to about five lines per day, and for real-time system software, about three lines per day.[15] This, of course, includes all life cycle costs, not just writing the code. There is some controversy over the absolute numbers, but it is the relative numbers that are of interest here.

Since the user is the person who best understands his or her problem, it would be ideal for the user to do the programming. Some day this may be the case; current trends are in that direction.

How does one assess the quality of software? It is impossible to get software with much of a user experience base in systems for developmental facilities. Therefore, expressions of mean time between failure and mean time to repair are meaningful only for very limited software systems. Even when this information is known for operating systems and vendor-supplied software packages, it generally is not readily available. We assess the quality of software in the same way a conscientious engineer might assess the quality of a new electronic module: he would look at the design and the attention to detail in the construction. In software, the design and attention to detail are probably even more important than in hardware. Just as one would not buy electronic equipment that looks as if it had been designed and built by an amateur, software not designed for professional use should not be incorporated into software systems designed for professional use. The source code should be investigated and assessed by a software professional for structure and attention to

detail. Since signals from the real world can cause changes in the way a program is executed (and it is impossible to simulate all possible real-world configurations), it is impossible to completely test any real-time software system. Although necessary, testing for specified functionality is not sufficient for the acceptance of a software package. A recent publication[16] was devoted entirely to a discussion of software development and reliability.

5.2 Control Concepts

The implementation of process control must address backup control, manual control, coupled subsystems, control modes, and integrated process control. This section deals with the concepts used to address these.

The issues involved in the control of an experimental process are somewhat different from those in an industrial process. As mentioned before, most industrial process control systems are automated versions of a manual process. The process has been controlled for years by operators, and now we want to automate the process. This is generally not applicable in the experimental process control environment. Most of our processes, like the AVLIS process, have never been operated successfully at anywhere near plant scale; and our major goal is to design a system that can be operated by a minimum number of operators.

Laboratory-scale facilities are controlled by a relatively large number of operators and experimenters, but that type of control does

not extrapolate well to a plant-scale facility. Our goal in this program is to develop a control system that will allow the process to be controlled economically by a staff whose number and training is minimal. Another factor is the economic competition among advanced enrichment processes. Since the U.S. Department of Energy is interested in selecting the most economically attractive enrichment technology for future funding, it is important that the process control system be able to control the process economically, with enough flexibility to adapt to changes in the requirements of the control system.

5.2.1 Control Modes And States

The ICAM system for the AVLIS process defines four control states and three control modes. The states are:

1. startup
2. shutdown
3. exception handling
4. steady state control

It is important that any control system handle these four states. Many only handle steady state and rely on something (or someone) else to handle the other three. The AVLIS ICAM system is designed to function in all four.

The system has the following three modes that can be enabled in any of the four states listed above:

1. Programmed supervisory: The control system is programmed to perform a series of actions based on overall requirements. This

is the optimal control mode in our design where overall performance is monitored and adjusted to minimize some overall cost function for the process. The setpoints for the controllers are set by the control system based on this cost function, and user input to the process control system is via the cost function. The parameters in the cost function may be adjusted in real time by the operator to affect overall process control.

2. Operator supervisory: The operator is responsible for selecting appropriate setpoints and actions. This is the normal "supervisory" control system. It is, by definition, sub-optimal.
3. Manual: The operator can directly manipulate final control elements (effectors) from his control panel. This mode is intended for maintenance and testing.

These control modes and states seem fairly simple but when combined with the requirement that all transitions occur without perturbing the process ("bumpless") and without requiring operator intervention ("balanceless"), it becomes very difficult to find a commercial controller that will completely meet our requirements. Figure 14 illustrates the input/output requirements for our controllers. The transition most difficult to support is the bumpless transfer from remote/auto, where the computer is supplying the setpoint for the controller, to remote/manual, where the computer is supplying the signal to be output to the final control element.

A control-related concept used in the AVLIS ICAM system is that of a subsystem. By defining relatively independent entities as

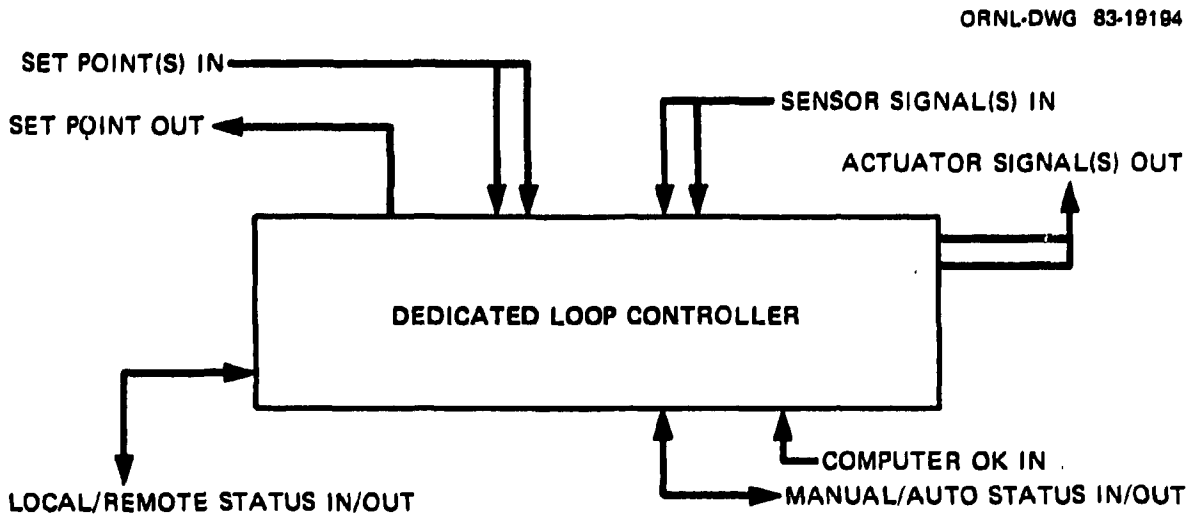


Figure 14. Dedicated loop controller showing communication signals required for integrated operation.

subsystems, we can place each subsystem in one of the three control modes independently. This allows a part of the facility to be in manual mode without impacting other parts, and the ICAM system continues to monitor the process regardless of the mode of control.

5.2.2 Control Algorithms

The primary use of computer-based (digital) process control systems is to replace conventional (analog) process control systems because of the advantages offered by the digital versions. Among advantages are cost of implementation, ease of programming, and reduction of error. The first control algorithm usually implemented is the "bang-bang" or on-off control, followed quickly by the proportional, integral, derivative (PID) algorithm. These and their derivations are still among the primary control algorithms used in process control systems today. The discussion here will include these as well as adaptive control and optimal control.

5.2.2.1 PID Control

PID control is the original attempt to automate a manual control strategy. If we watch the way a trained operator controls a simple heater system, we can see him implementing the PID algorithm; increasing gain with the error and the time over which the error has been accumulating, but reducing the gain as the setpoint is approached to prevent overshoot. A major difficulty in implementing this algorithm in a digital computer is the derivative component. In analog controllers, differentiators are more like a lead-lag compensator (rather than the pure lead known as a differentiator). Pure

lead (or derivative as it is called in PID nomenclature) is dangerous in a process control system because any noise imposed on the signal is differentiated, making it worse. Some implementations differentiate the error rather than the process variable signal to help alleviate this problem, but then a setpoint change is differentiated and applied to the controller. Either of these conditions causes serious instability in the control system. A number of approaches can help minimize this, but the best is known as compensated rate feedback, which is described in detail in Appendix A. The basic idea is to replace the pure integral (lag compensator) and pure derivative (lead compensator) with a lead-lag compensator. In this way the lead compensator prevents the control system from overshooting while not causing serious stability problems. Another approach to the problem of PID has been to drop the D and use only PI. This, of course, forces the system to overshoot and should be avoided.

A disadvantage of the compensated rate feedback technique is the loss of direct control over the three tuning parameters--gain, reset, and rate. There are corresponding parameters in the compensated rate feedback technique, but they are coupled and must not be modified independently. This is not a problem, however, because control of the process is generally better without the knobs to adjust. Many of the single loop controllers sold today use a digital computer rather than an analog controller. It is therefore important to understand how the algorithm designed.

This brings up the concept of direct digital control (DDC). Earlier in the history of process control systems it was considered dangerous and not cost effective to use DDC, that is, having a digital computer generate the signals that manipulate the final control element. This attitude has changed, however, because most controllers are DDC inside and only look like they are analog. It is important to understand why the DDC was not considered good practice when using these computer-based controllers. It is important that the controller not be dependent on a communication line to produce its control action; it should receive setpoint and status information from the host but should be functional without that link. In our ICAM system, these are referred to as dedicated loop controllers rather than single-loop controllers since they can have multiple control loops.

Availability is the primary issue involved in the use of DDC in the AVLIS ICAM system. It is important in AVLIS that low level setpoint control be functioning even if the host is unavailable. This can be implemented with dedicated analog or digital controllers. It is a part of the concept of distributed liability. A key requirement, though, is that control continue for several minutes without host intervention, which sometimes means the controller has to use a complex control algorithm and have access to diverse sources of data. This requirement reflects the belief that controllers should not rely on outside intervention to maintain safe control parameters. All control devices are designed to hold their previous setpoints as the

default action. The high-speed reflex actions required for equipment and personnel safety are handled in the programmable controller.

5.2.2.2 Advanced Control Concepts

Many control systems today can accommodate cascade control, that is, the output from one controller serves as the input to another. This is a useful capability, especially if the first controller can be a model of the process being controlled rather than an actual controller. If so, this then makes optimal control and adaptive control possible.

Adaptive control was one of the first of the advanced control strategies to be implemented when computers began to be used in process control. By basing the control action of a loop on the status of another control signal, or a calculation of some figure of merit, engineers could handle exception conditions within the automatic control system. At first the only possible reaction to the exception condition was process shutdown, and this was sufficient for a time. Then it became obvious that by reducing certain operating parameters sufficiently one could assure safe operation and speed recovery. This is one of the underlying concepts in optimal control.

In optimal control the operation of the process is considered in an overall, global sense. At a given operating point there are optimum setpoints that will produce maximum product for minimum cost. (The terms "product" and "cost" are used very generally here, meaning any output and any input.) The control algorithm might consider the inventory of raw material for example, and reduce plant throughput,

which is cheaper than exhausting the entire inventory of raw material and shutting down completely.

5.2.2.3 Artificial Intelligence

Artificial intelligence and optimal control are two entirely different concepts. Artificial intelligence is not a substitute for optimal control. Optimal control is based on a mathematical model of the overall operation of the process and comparison of the current operating conditions with the model to determine what parameters must be adjusted to restore optimum performance. Optimal control is a mathematical process, whereas artificial intelligence attempts to emulate a skilled operator and is therefore more empirical than mathematical.

Recently there has been a great deal of research and development in the area known as artificial or machine intelligence. The idea of teaching a computer to think has tremendous appeal and potential. The early program ELIZA,[17] which simulates a psychiatrist analyzing the person at the keyboard, is considered one of the first attempts at artificial intelligence. The author of the program ELIZA, however, has denied that his effort marks a significant advance in the area.

The systems called "expert systems" are gaining acceptance today and performing useful functions in the areas of medicine and maintenance assistance. Digital Equipment Corporation (DEC) is one of the leaders in using expert systems to assist in diagnosing ailing computer systems. The use of expert systems in process control for

utilities and nuclear reactor control is being investigated at Oak Ridge National Laboratory and other installations. The idea is to record, over a period of time, a skilled operator's responses to varied plant conditions. The computer also logs plant conditions before and after the action, so that later the computer can respond to interrogation from a novice operator regarding appropriate responses. The expert system benefits from the experience of a number of individuals and can recall which past actions produced "good" results under similar conditions.

5.2.2.4 Backup Control Considerations

The AVLIS ICAM design differentiates between backup and manual control. Backup control is independent of the host computer system. It can be enabled automatically upon host failure or by operator action. Manual control is achieved by disabling the setpoint control action and thus directing the output to the final control element. This can be done from either the local panel or the host computer's user interface.

An important consideration is how manual is manual. For example, a complex heater arrangement may have 40 to 50 heater controllers that need to be adjusted separately in manual operation. How can we expect an operator to do this effectively? In fact, he has been trained to use the automatic controls, and, even if he had to operate the process manually in the past, he has not done it recently. This is especially critical in backup control systems

because no supervisory system is available to oversee manual operation.

Engineers spend a great deal of effort designing the control system to take care of all routine control actions with minimum operator intervention. Then what happens if the computer system fails? The operator must return to manual control from a rack of panel meters spread over several cabinets containing a lot of distracting information. Some feel that the best backup control action is to simply shut down. The AVLIS plan is to maintain local control in backup mode for a time, while trying to get the host system back on line. If this fails, the operator will have to take action, either to shut down the process or try to operate it manually. The goal has been to reduce the number of operators required, so we can't expect to have enough operators to operate the process as they did when it was a bench top experiment. We must then provide operator assistance even in backup control. An intelligent backup panel appears to be a reasonable compromise. This provides the operator with some assistance in operation but the expense and complexity of a fully redundant hot backup host system is avoided.

5.3 AVLIS ICAM Functions, Features, And Techniques

The major functions provided by the AVLIS integrated control and measurement (ICAM) system are described in this section. Some of these functions are included because of the development nature in both the process and the instrument system, with the understanding

that the cost of adding functionality after delivery is routinely one hundred times more expensive.[18] For systems where the process being instrumented is well understood, some of these capabilities are not necessary. The design described here functions as a skeletal framework upon which to build the final product. Included with each function is the use it might have in the final product; but, since this system designed as a tool and not a solution to a problem, it will have capabilities that are as yet unknown. This is in contradiction to systems built to strict criteria. If one of these systems does something that is not included in the specification, it is considered a problem. Because our system is expected to be used as a tool, we don't know what it will eventually have to do. It would therefore be ineffective to develop a system restricted to performing only those functions programmed in at its inception.

The following functions are provided in the ICAM system:

1. data acquisition at rates prescribed by the operator
2. limit checking
3. exception reporting and display
4. exception action initiation
5. graphic data display
6. tabular data display
7. user interface that is fast and easy to use
8. setpoint and advanced control

5.3.1 Data Acquisition

The data acquisition section is list driven but does not use the standard scan table used in some other systems.[19] The purpose of a scan table is to define how and when the process signals are to be read by the computer, and some scan table techniques assume that all process input comes from the same device. This assumption improves efficiency but sacrifices modularity. The block scan table used in our system allows separate responsibility for different scan blocks and scan rates, thereby allowing the interruption of data acquisition for any particular block without interfering with other blocks.

Different scan rates can be defined at the time the data point is entered into the system. As used in some systems, a fixed scan rate simplifies the design and, if fast enough, does not introduce undue aliasing errors caused by sampling the data too slowly. Our system supports differing scan rates because there was no way to know the maximum scan rate and we could not risk making all of the signals dependent on the needs of the fastest one. Again, commercially available systems are designed for processes that have relatively long time constants and, even if the data can change faster, the controllers don't need to respond any faster. The AVLIS system needs the data for other than control action, so we support independent scan rates.

The block scan technique used in the AVLIS design takes advantage of the block scan hardware available in many analog front end subsystems. Block scanning allows the user to specify the beginning

channel number and number of channels rather than specifying each channel. Many subsystems support direct memory access (DMA) on block data transfers. This technique offers significant throughput advantages over random channel transfers.

The block scan technique permits different device types. Some examples of the device types supported on the AVLIS ICAM system are CAMAC (computer assisted measurement and control) input/output modules, programmable controllers, and computed points. The computed point "device" gives us the capability of providing values that are not available directly from transducers but that are actually combinations or functions of transducer signals. This also permits cascade control by passing setpoints through the system. Some examples of computed points are running averages, weighted averages, maximum value over time, or the name of the point that has the highest temperature in a particular region in the process.

The engineering unit (EU) conversion of the input values is performed by the data acquisition software. This simplifies the interface to the data base and permits all control and other computations to be in engineering units. Some systems do all computation and control in integer arithmetic, which speeds up the computations but makes the implementation of complex control algorithms especially difficult. By keeping the conversion process synchronous with the scan we simplify intertask communication (a major goal of the AVLIS design) and sacrifice speed. The philosophy of this is that first we will make it work, then we will make it fast. This can cause

problems if no "hooks" are provided to add speed later if it becomes necessary. By performing the reverse calculations on the data stored in the data base, we could have all data stored and entered in EU format while performing all arithmetic in integers. We don't think this will be necessary, but it can be done if it does become necessary.

We primarily use weighted least-squares curve fitting for the engineering unit conversion of transducer signals.[20] This fits the data structures better than table look-up, and it provides reasonable accuracy. A simple least-squares fit on thermocouples proved unsatisfactory because this technique minimizes the integral of the error over the entire range of interest, and minimizing the average error produced unsatisfactory results at the process temperatures. The weighted technique allows us to specify where we want good accuracy, and we take whatever we get in the rest of the curve. By trial and error, we were able to get within 1°C of the National Bureau of Standards (NBS) tables for type S thermocouples in the range 120 to 1200°C, while maintaining 2°C accuracy throughout the remainder of the temperature range, 0 to 2500°C.

Our experience with other systems warned that without care, we could display data containing inaccuracies and inconsistencies, especially during a high interference condition. Such a condition might be caused by an electrical arc that occurs at the instant the analog-to-digital converter performs its conversion. Taking action to reduce the occurrence of the interference is not sufficient by

itself. The system must handle glitches without displaying erroneous data. This could have been done in a number of ways, including digital filters in software or hardware, but a simpler technique using the limit-checking facilities was developed. This makes implementation simpler and more easily modified than do the other alternatives. Simply stated, the technique checks the new value against "bad data" limits entered by the user at the time the transducer is defined. If the limit is exceeded, updating of the data is suppressed until the excursion is repeated a specified number of times (usually three). This ensures that a transducer failure will be detected and appropriate notification given.

The same technique is used to suppress "hash" noise in the data. By specifying a "significant change" limit, the user can suppress the updating of data that has not significantly changed. Suppressing unnecessary updating in a distributed system is useful because there is no need to communicate unchanged values to the next higher level. We protect against the slowly changing signal by always comparing the new value to the original value. An important goal of this function is to trade off local computation power (required for the limit check) against communication bandwidth in notifying the upper levels of each change, no matter how small.

5.3.2 Limit Checking

The limit-checking algorithms provided in the AVLIS ICAM system use EU converted data exclusively. This makes development and maintenance easier and trades off computational speed and power. We felt

that this was an equitable trade, which be reversed later if raw speed becomes more important. As noted earlier, limit checking is useful for more than just exception condition alarms. Limit checking is also used to implement hash and glitch filters and to initiate actions based on limits, which implement a simple "bang-bang" control algorithm. Since the annunciation and logging of limit excursions are enabled independently, the user can create limits that are recorded but not annunciated.

5.3.2.1 Alarm Functions

By enabling the annunciation and logging of a limit, a user can create a traditional alarm. Our system supports many different types of alarms, which are listed below with a summary of the use of each type.

1. High/low absolute -- classic alarms
2. High/low deviation -- checks controllers against setpoints
3. High/low rates -- checks rate of change
4. significant change -- for hash filtering (see EU conversion in Sect. 5.3.1)
5. bad data -- for glitch filtering (see EU conversion in Sect. 5.3.1)

Any one of the first three limit types above can be specified as a "bad data" limit or "significant change" limit for use in the filtering techniques described in the discussion on EU conversion. Also, by specifying an action task to be initiated upon limit excursion, the user can initiate simple control action sequences.

Since in this system we do not pack digital points into bits, we can treat them the same as analog points except that their values are stored as ASCII strings (OPEN/CLOSED, for example) that can be compared against limits or setpoints. Very willingly we again traded storage requirements against ease of development and maintenance. We were designing this system for use on a DEC VAX computer, although its first implementation was on a DEC PDP-11/23. The large address space of the VAX allowed us to make this tradeoff without great concern. This has turned out to be a good tradeoff thus far.

By using a deviation limit we can implement a programmable limit, which can be useful in reducing nuisance alarms associated with setpoint changes. Another useful concept in alarm handling is that of conditional limits. This concept could be applied to eliminate nuisance alarms that occur as a result of a previous alarm already reported. The current implementation of the ICAM system does not support conditional alarms.

5.3.2.2 Alarm Display

Alarm status information is included on all displays viewed by the operator. There is also a dedicated alarm display, which was added at the recommendation of an operator in one of the AVLIS facilities. He wanted to be able to look to a dedicated monitor to get immediate alarm status information. The limit display contains the point name, the limit type, the limit value, the current value (updated in real time), and a representation of the current status of the alarm.

An alarm can be in any one of the following four states:

1. New alarm -- unreset, uncleared
2. Old alarm -- reset, uncleared
3. Cleared new alarm -- unreset, cleared
4. Cleared old alarm -- same as cleared new alarm

"Reset" is an operator action indicating that he has taken whatever action he can to clear the alarm. This action is called "acknowledge" in some systems.

Each alarm state is represented by a distinct attribute on the alarm display. This is designated as the Instrument Society of America (ISA) sequence A1 with silence option. (The silencing of the annunciator causes no action other than silencing the horn.) In a development system it is important that the operator know not only the current status but how the alarm got to the current status. Therefore, no alarm indication is removed from the screen until it is cleared and the reset button is depressed.

Some systems remove the point from the screen when it clears or require that each new alarm be "acknowledged," but both of these alternatives are unattractive in a development facility. We don't want important data to disappear, and it is our philosophy that nothing major should happen to the operator's displays without his direction. This procedure is not followed in some systems where an alarm might cause the display to change to a display that better shows the condition of the plant. We recommend a display but do not force a change. Forcing someone to acknowledge every alarm can lead

to his ignoring important status information and blindly pressing the button. Our concept is to acknowledge (or reset) all alarms that are currently in alarm. This can be a problem only if an alarm is exceeded and is cleared just before the operator pushes the reset button, which would hide the transient alarm condition.

Another concept in our system is that of supporting alarm priorities. We display the alarms in chronological order on the hard copy diary of alarms, but on the video monitor they are displayed in priority order, specifically in chronological order within blocks of like priority. The most critical alarms are displayed at the top of the screen while the less important ones are lower. (If the screen fills, the lower priority alarms are scrolled off the bottom, but they can still be viewed by forcing a scroll from the operator's console.) Very few commercially available systems offer anything except chronological order in the alarm display (if a dedicated display is available at all).

The features and functions described in this section are implemented in a system being used in a smaller scale facility to prototype the requirements for the material handling developing module (MHDM) facility at ORGDP. The prototype is installed on the Electron Beam I (EB-I) experimental facility at the Y-12 plant. The next implementation, which is currently under development, will use a VAX 11/750 in place of the LSI 11. (Refer to Figures 5 through 11 in Section 3).

6. HUMAN INTERFACE

The man-machine interface (MMI) for a process control system is at least a thesis in itself. Although many vendors claim to have solved the problems that plagued this area in the early days of the business, it is still important for the customer to understand the issues so that he can judge the success of the vendor in addressing those issues. Following are some of the issues that should be addressed in developing or specifying a user interface.

1. Effective data display: There is a tendency in using a computer to make it act like the object or system it is replacing. This is good in some cases, but to use a sophisticated color display system to look like a rack of instruments is not the best way to present critical data. Other techniques are available and are being developed all the time. The accident at Three Mile Island demonstrated that conventional display of information is not adequate during a crisis and we should not copy inadequate techniques with computer-based systems.
2. Host computer load: The tremendous quantity of data usually transmitted and received in order to generate a picture can represent a heavy load on the host system. Almost everything that goes on in a process control system involves input/output (I/O). For instance, if the resources of the host computer are tied up with I/O to the graphics subsystem, no data can get in from the process. Although a number of alternatives are

available, most are not very satisfactory. The vendor should be able to defend his I/O system and justify its load on the host computer. Almost no vendor will guarantee throughput since it actually measures the overhead as well as the actual data transmission. (Beware of vendors who quote raw speed and claim that the actual speed is the same.) The best options for the host interface include the parallel direct memory access (DMA) interface, the programmed I/O parallel interface, and shared memory.

3. Local image storage: One approach to managing the large quantity of data that moves between the host and the display is to provide local storage in the display system. This can speed up the display of the static part of the image, which usually makes up the largest percentage of the image. Some vendors, however, store the images on floppy disk or some other slow and unreliable medium. Local storage should be solid state memory or Winchester disk. These media have the speed and reliability to improve the overall performance of the system if handled properly. Note, however, that not all Winchester drives are suitable. The ones best suited for a process control system have sealed platters and voice-coil, closed-loop servo-head positioners (as opposed to the stepper motors in the personal computer type of Winchester disk).

4. Suitable speed: This should be the maximum time to change from any image in the system to any other, and it translates to about one to two seconds before frustration sets in. If the image

requested is not in local storage in the display system, it will have to be downloaded which takes more time. This is one of the figures of merit of such a system.

5. Judicious use of color: The images should not hurt the eyes. (Pure blue on a black screen is especially uncomfortable, and other pure colors can be inappropriate also.) The best colors to use for video display are pastels or earth tones. Avoid the pure colors and beware of systems that offer only pure colors as hard-wired choices.

6. Adequate resolution: A resolution of any less than 512 × 360 is usually unsuitable because the figures get distorted and the graphs show severe aliasing. Some of the new systems use an automatic anti-aliasing algorithm to remove the distracting appearance of "the jaggies." Check the appearance of single-width lines that are just off the vertical and horizontal for this effect.

7. Image flicker: Long persistence phosphors (LPP) are not the answer to the image flicker problem. There is currently no adequate LPP green available, so any line that contains a substantial amount of green will flicker more than others. Flicker is dependent on a number of things, and to avoid it without worrying about content and color requires an update rate of 63 Hz non-interlaced.[21] This rate is found only in high quality display systems.

8. Misconvergence: The monitors can be either precision-in-line (PIL) or Delta guns. This indicates how the guns are arranged in or the CRT. The important consideration is that a PIL type never requires or permits convergence adjustments. Any misalignment in the guns, causing color fringing in the corners, can be corrected only at the factory. No adjustments are available, and the convergence that exists when the monitor is unpacked is the convergence for the life of the device. The Delta guns require routine realignment by skill d technicians. Some manufacturers have attempted to simplify the procedure but it is still cumbersome at best. The Delta gun type monitor is usually capable of better convergence if properly adjusted, but it can drift out of adjustment in a matter of days. The PIL does not drift appreciably, but its convergence is usually not as good as can be obtained by a skilled technician on a Delta gun monitor. The place to watch for convergence problems is in the corners. Manufacturers usually specify convergence in a number of places, but the figure of merit is usually the extreme corners. A misconvergence of less that 0.8 mm over the entire screen is usually suitable, with 0.5 mm near the center. This figure is tough to get on a 19-inch (diagonal measure) monitor. How close the user will be to the monitor is the major consideration. If the monitor will be at the top of the room for project/program managers to watch, 1.2 mm for a 19-inch monitor would be adequate, but if the operator is going to be arm's

length from the tube, it needs better performance. For process control and data acquisition systems, I recommend the PIL tubes with 0.8 mm or less misconvergence anywhere on the screen.

7.0 SUMMARY

In this paper I have attempted to point out some of the major issues involved in the development of an integrated control and measurement system. I have tried to show where the type of systems addressed here differ from those in industry as well as those designed for use on burst type operations. However, in this thesis, no attempt was made to address any one of the issues totally. The intended purpose here is to provide enough information about these critical issues to enable the reader to ask intelligent questions of the supplier. Many of the issues mentioned in this paper could be expanded into a thesis by themselves. Additional information on these subjects may be found in the references cited or in subject indexes available in major libraries.[22-30]

LIST OF REFERENCES

LIST OF REFERENCES

1. Merritt, Rich. "Large Control Systems: Big Price Tags But Even Bigger Returns." *Instruments and Control Systems,* 54(12): 35-37, 40-42 (December 1981).

2. Daniels, Robert. "Nuclear Fusion Reactor Uses Highly Integrated Monitoring and Control." *Comput. Technol. Rev.* 1(3): 391-397 (Summer 1983).

3. Shooman, Martin L. *Software Engineering: Reliability, Development and Mangement.* New York: McGraw-Hill Book Company, 1982. Page 110.

4. Dartt, S. R. "Distributed Control Systems - Its Pros and Cons." Plastics Tech. Dept., G.E. Company, Pittsfield, Massachusetts. *Chemical Engineering* 89(18): 111-114 (September 6, 1982).

5. Kompass, E. J. "Integrated Hierarchical Process Control System is Distributed to Dedicated Loop Controllers," *Control Engineering* 30(3): 93-94 (March 1983).

6. DeMarco, Thomas. *Structured Analysis and System Specification.* New York: Yourdon, Inc., (1978).

7. Hayes, Robert W. "Distributed Resource Management: Deadlocks," Oak Ridge National Laboratory Report ORNL/TM-8086, June 1982.

8. Humphrey, Watts S. "In Favor of Evolution," *IEEE Spectrum* 19(12): 17 (December 1982).

9. Chandler, James M. "How to Make Accurate Low-Level Measurements," published by Acurex Autodata.

10. Chase, Donald. "Consider Every Error Source For Data Acquisition Design," *Control Engineering* 27(6): 65-69 (June 1980).

11. Chase, Donald. "System Architecture Affects Data Acquisition Accuracy and Flexibility," *Control Engineering* 27(7): 83-86 (July 1980).

12. Owens, Dale R. "A Case for Custom Software," *Instruments and Control Systems* 54(11): 65-67 (November 1981).

13. Herbert, Evan. "Minis and Mainframes," *IEEE Spectrum* 20(1): 28-33 (January 1983).

14. Page-Jones, Metler. *The Practical Guide to Structural Systems Design*. New York: Yourdon Press, 1980.

15. Boehm, Barry W. "Software and Its Impact: A Quantitative Assessment." *Datamation* 19: 48-59 (May 1973).

16. IEEE *Computer* 16(11), November 1983.

17. Weizenbaum, Joseph. "ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine," *ACM Commun.* 9(1): 36-45 (1966).

18. Shooman, Martin L. *Software Engineering: Reliability, Development and Management*. New York: McGraw-Hill Book Company, 1982. Page 15.

19. Eason, R. O. "Development and Comparison of Algorithms for Generating a Scan Sequence for a Random Access Scanner." Oak Ridge National Laboratory Report ORNL/TM-7504 (September 1980).

20. Allgood, Glenn O., and Wayne W. Manges, Oak Ridge National Laboratory draft report, April 1983.

21. Heines, J. M. "Current and Future Problems Face Digital Displays." *Digital Design*. December 1981. Page 32.

22. Tsai, T. H., Lane, J. W. "Segregate Basic and Advanced Control Functions in Distributed Control," *Control Engineering*, 30(4): 123-126 (April 1983).

23. Larsen, G. R. "A Distributed Programmable Controller System for Batch Control," *Intech* 30(3) 53-55 (March 1983).

24. Yunker, R. W. "User Experience With Distributed Control in the Glass Industry," *Proceedings of the First Annual Control Engineering Conference*, May 18-20, 1982, pages 71-72.

25. Ash, R. H. "The Power, Profit and Potential of Distributed Computer Control Systems," *Proceedings of the First Annual Control Engineering Conference*, May 18-20, 1982, pages 79-83.

26. "Distributed System Controls Batch Process Plant," *Instruments and Control Systems*, 55(5), pages 73-74 (May 1982).

27. Cho, C. H. "Distribution of Intelligence to Microprocessor-Based Analyzers and Field Devices for Process Control," *ISA Transactions* 21(1): 61-67 (1982).

28. Evans, L. B. "Impact of the Electronics Revolution on Industrial Process Control," *Science* 195 No. 4283, 1146-51 (March 18, 1982).

29. Bala, J. L. "Distributed Processing: The Control Strategy of the 80s," ISA - Proceedings of the 1982 Symposium, April 27-29, 1982, pages 31-37.

30. Johnson, T. L. "Multitask Control of Distributed Processes," Proceedings of the Third IFAC Symposium, June 29-July 9, 1982, pages 269-73.

A-1. Johnson, E. F., *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 139.

A-2. D'azzo, J. J., and Houpis, C. H., *Linear Control System Analysis and Design: Conventional and Modern.* New York: McGraw-Hill, 1975, p. 358.

A-3. Johnson, E. F., *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 228.

A-4. Manges, W. W., "Control of Specimen Temperature Using Gas Mixtures," draft report (1982).

C-1. Johnson, E. F. *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 228.

APPENDIXES

APPENDIX A

USE OF COMPENSATED RATE FEEDBACK IN PROPORTIONAL INTEGRAL DERIVATIVE (PID) CONTROL

INTRODUCTION

Three-mode proportional integral derivative (PID) control offers significant advantages over straight proportional integral control.[1] Too often, however, the derivative portion is omitted because of the increased noise sensitivity it introduces. Although differentiation of a noisy signal can introduce instability, the omission of the derivative action from a control strategy can cause other problems. Because no component of the control signal in a simple proportional integral (PI) controller can change sign until the error signal does, the system must overshoot. Rate feedback, or derivative control, adds a signal that is proportional to the rate of change of the error signal. This signal will change sign as the setpoint is approached. Adding compensation to the rate feedback can minimize the noise sensitivity. In this paper, I will describe a technique that can be used to implement a PID algorithm using compensated rate feedback to avoid the noise problem associated with the use of pure derivative.

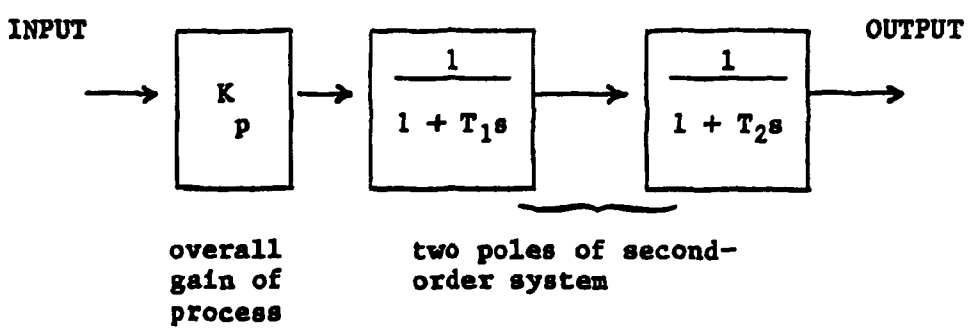
DISCUSSION

Part of the difficulty experienced by engineers in the design of control systems is caused by differences in the nomenclature used in various areas of engineering. In traditional electrical engineering, the integral controller is known as a lag compensator and the derivative portion of the controller is known as a lead compensator, the lead-lag compensator is missing from this correspondence scheme. The lead-lag compensator, when used in place of a pure lead compensator

(derivative control), has some very useful properties.[2] Among these are faster response with improved steady-state performance when compared to pure lead compensation.

In analog controllers, the derivative circuit is routinely implemented as a lead-lag compensator, since a pure derivative is more difficult to implement. This inherent lag compensation has prompted designers to neglect the lag portion when implementing the controller in software. This problem is further aggravated by the inaccurate documentation supplied with many analog controllers, which leads people to believe that the controller uses pure derivative action.

For the purpose of this discussion, I will concentrate on a second-order, overdamped process. These assumptions do not impose undue restrictions, however, since many processes exhibit these characteristics. In particular, I will consider the control of the temperature of a thermal system. Assuming that the system can be characterized as second order, we can describe it with a block diagram:



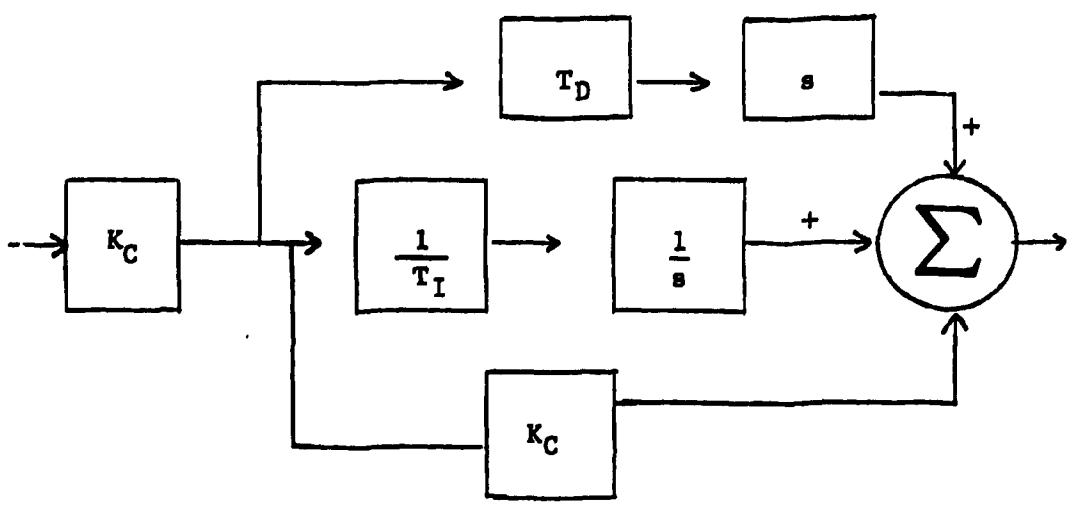
where the variables are defined as:

K_p = overall gain of the process

T_1, T_2 = two time constants of the second-order system

s = LaPlace transform operator

The control strategy is to add PID control to this system to maintain the temperature at a given setpoint. The traditional method of implementing this controller, especially in a digital computer algorithm, is to use a pure proportional section in combination with pure integral section and a pure derivative section as shown in the block diagram below:



where

$\frac{1}{s}$ = integral operator

s = derivative operator

T_I = reset time

T_D = rate time

K_C = proportional gain of controller

Thus the transfer function of the controller is:

$$G_C(s) = K_C + \frac{K_C T_D}{T_I} + \frac{K_C}{T_I}\frac{1}{s} + K_C T_D s \qquad (1)$$

Since this transfer function involves differentiation, the noise problems associated with the pure derivative would be exhibited. The Bode plot in Figure 1 shows the increase in gain at high frequency.

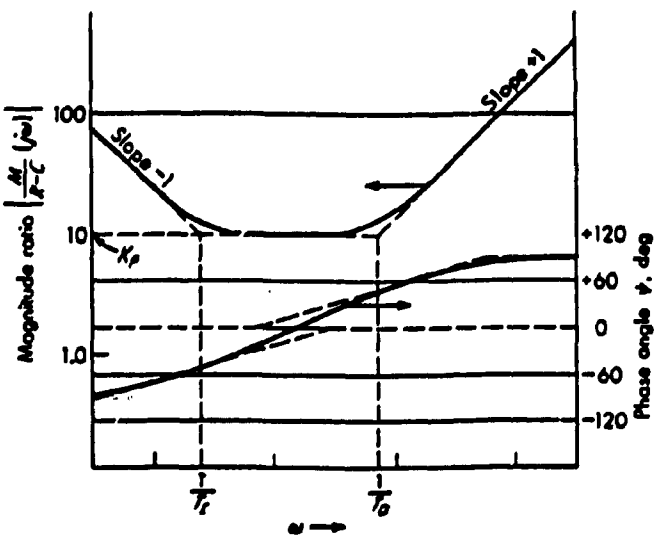


Figure A-1. Bode diagram for ideal three-model control. (Reprinted from Reference A-1.)

This, however, is the technique used in many computer-based control algorithms.

By modifying the pure lead into a lead-lag compensation network, we can avoid the problems associated with the pure derivative. The transfer function becomes:

$$G_S(s) = K_C \left(\frac{1 + T_I s}{T_I s}\right)\left(\frac{1 + T_D s}{1 + aT_D s}\right) \qquad (2)$$

where a is a compensation constant varying from about .05 to about .10 depending on desired characteristics.

Comparing the Bode diagram in Figure 2 with the previous one shows improved response at high frequencies.

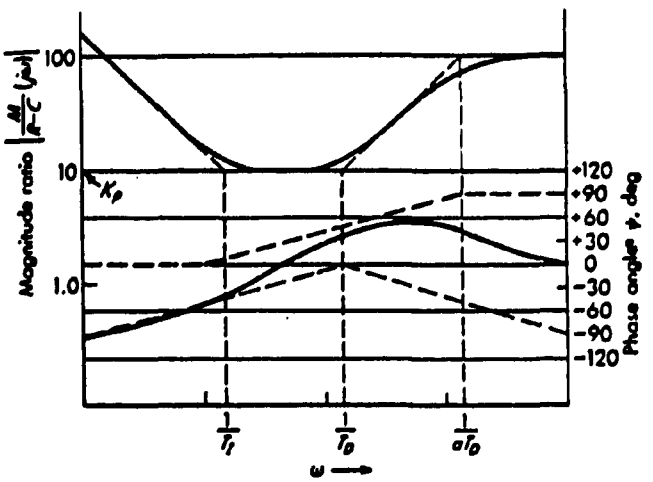


Figure A-2. Bode diagram for three-mode control with compensated rate action. (Reprinted from Reference A-1.)

The algorithm for the transfer function in Eq. (1) is quite simple compared to Eq. (2), which explains its current popularity. The transfer function in Eq. (2) requires a bit more manipulation before its implementation becomes evident. Since this is not a strictly proper fraction, it must be divided into the form

$$G_C(s) = \frac{K_C}{a}\left[1 + K_1 \frac{s + p_1}{s(s + p_2)}\right]$$

which, upon partial fraction expansion, yields

$$G_C(s) = \frac{K_C}{a}\left[1 + K_1\left(\frac{p_1}{p_2}\right)\left(\frac{1}{s}\right) + \frac{K_1(p_2 - p_1)}{p_2}\left(\frac{1}{s + p_2}\right)\right]$$

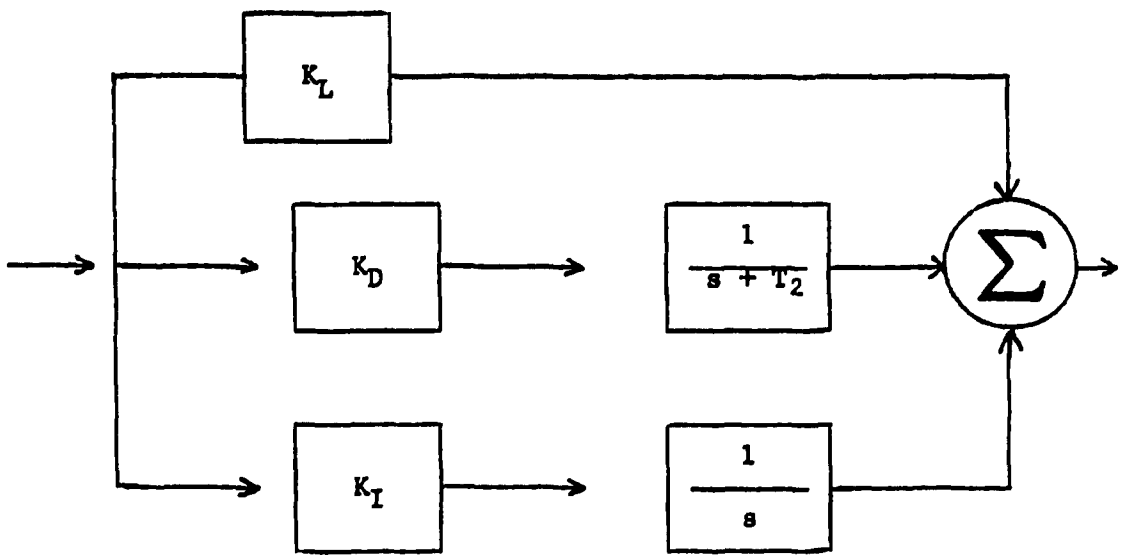
where

$$K_1 = \frac{1}{T_I} + \frac{1}{T_D} - \frac{1}{aT_D}$$

$$p_1 = \frac{1}{T_I T_D K_1}$$

$$p_2 = \frac{1}{aT_D}$$

The block diagram for this transfer function is easily found to be as shown:



where

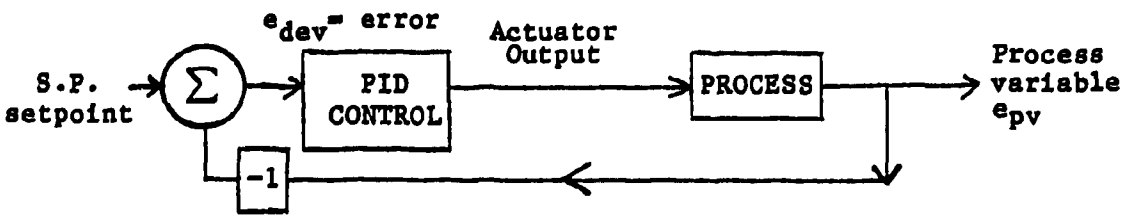
$$K_L = \frac{K_C}{a}$$

$$K_D = K_C \left[\frac{K_1}{a} - \frac{1}{T_I K_1} \right]$$

$$K_I = \frac{K_C}{T_I}$$

See Appendix B for a more detailed derivation.

The $\frac{1}{s + T_2}$ block can be expanded into a form that uses only pure integrals. Since integration is a stable operation and easily implemented in a digital computer, this is the form we will use. The following block diagram shows the overall control system $G_C(s)$ with all parameters as defined previously and indicates the position of the controller in the overall system.



The function of the controller is to act on the error signal and generate some action to correct the process. Therefore, the following equation is used to generate the actuator output:

$$\text{output} = K_L e_{dev} + K_D e_{dev} - T_2 I_p \int e_{dev} dt + K_I e_{dev} \int e_{dev} dt \quad (3)$$

where

I_p = previous output of the integrator

$K_L = \frac{K_c}{a}$

$K_D = K_c \left(\frac{K_1}{a} - \frac{1}{T_I K_1} \right)$

$K_I = K_c / T_I$

a = compensation constant $.05 < a < .10$

$$K_1 = \frac{1}{T_I} + \frac{1}{T_D} - \frac{1}{aT_D}$$

K_c = gain of controller

I/T_I = 'reset' of controller

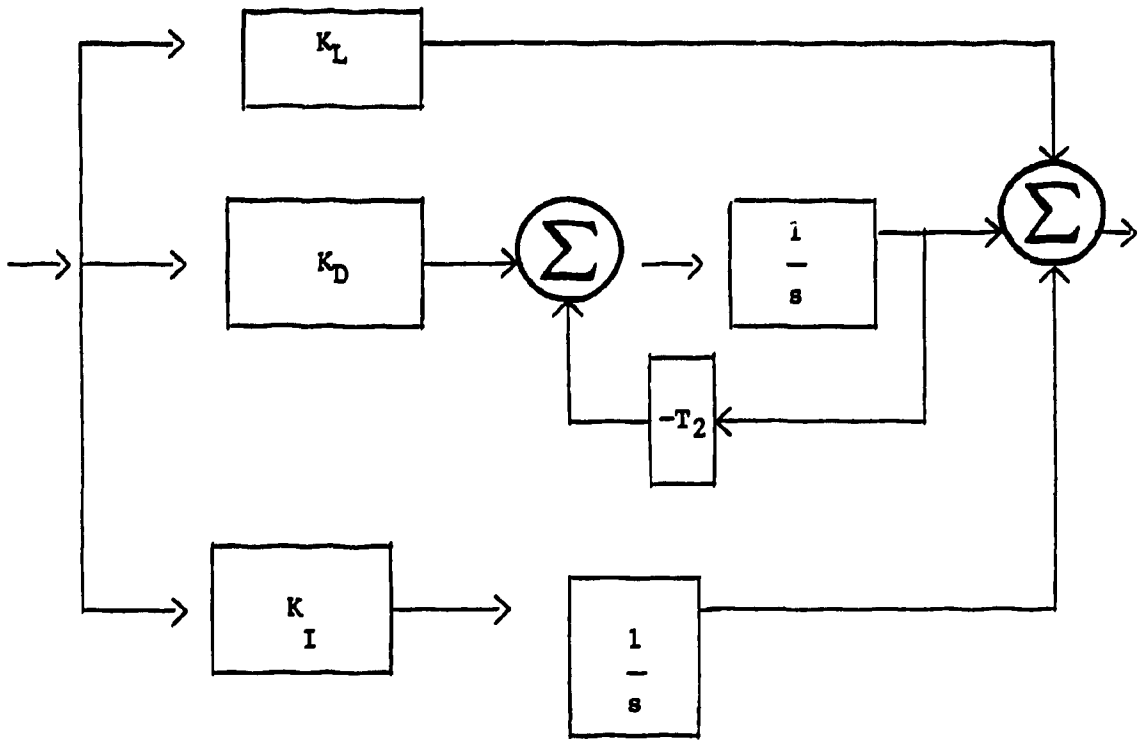
I/T_D = 'rate' of controller

Note that this implementation eliminates all differentiation and thus is stable for any input signal perturbation. It does, however, require memory and an initial value for that memory on the first iteration. Compare this with the equation of most conventional controllers:

$$\text{output} = K_c\left(e_{dev} + \frac{1}{T_I}\int e_{dev}dt + T_D\frac{de_{pv}}{dt}\right) \qquad (4)$$

This particular controller differentiates the process variable signal rather than the error in order to suppress perturbations introduced by a change in the setpoint. The implementation in Eq. (3) using compensated rate feedback does not have this problem. The advantages introduced by this compensation are worth the additional complexity.

This implementation in the time domain is fairly straightforward. Each block designated 1/s is replaced by a pure integrator. By varying the step size in the Continuous System Modeling Program (CSMP) model and using the various integration techniques available there, one can investigate the effects of sampling and of using a simple summation algorithm to approximate the integral.



Selecting the proper values for K_C, a, T_D, and T_I can be done using a procedure known as the Ziegler-Nichols Method.[3] This technique is described in Appendix C. Finally, the engineer may fine tune the controller using the CSMP to model the entire system. Note that the parameters to be changed are K_C, a, T_D, and T_I. These may be adjusted for desired performance, but not K_L, K_D, and K_I. A detailed report of this procedure as applied to a specific control problem can be found in Reference 4.

CONCLUSION

In this paper I have described a procedure for the implementation of an improved PID algorithm which can be used without the danger of noise susceptibility exhibited by controllers using pure derivative action.

Even techniques such as averaging the derivative only approximate the result obtained using compensated rate feedback. By the judicious initial choice of parameters and final tuning using a computer model of the system, the designer can develop a control system with the desired characteristics in a reasonable amount of time without having to resort to manipulating the physical system.

APPENDIX A REFERENCES

A-1. Johnson, E. F., *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 139.

A-2. D'azzo, J. J., and Houpis, C. H., *Linear Control System Analysis and Design: Conventional and Modern.* New York: McGraw-Hill, 1975, p. 358.

A-3. Johnson, E. F., *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 228.

A-4. Manges, W. W., "Control of Specimen Temperature Using Gas Mixtures," draft report (1982).

APPENDIX B

DERIVATION OF PID ALGORITHM FROM G (s)

$$G_C(s) = K_C \frac{(1 + T_{Is})\ (1 + T_{Ds})}{T_I(s)\ (1 + aT_Ds)} = \frac{K_C}{a} \frac{\left(\frac{1}{T_I} + s\right)\left(\frac{1}{T_D} + s\right)}{s\left(\frac{1}{aT_D} + s\right)}$$

Let

$$a_1 = \frac{1}{T_I}$$

$$a_2 = \frac{1}{T_D}$$

$$a_3 = \frac{1}{aT_D}$$

$$G_C(s) = \frac{K_C}{a} \frac{(s + a_1)(s + a_2)}{s(s + a_3)} \quad .$$

Using long division:

$$G_C(s) = \frac{K_C}{a}\left[1 + \frac{1 + (a_1 + a_2 - a_3)\ (s + a_1 - a_2)}{s^2 + a_3 s)}\right] .$$

$$G_C(s) = \frac{K_C}{a}\left[1 + \frac{s + \frac{a_1 a_2}{(a_1 + a_2 - a_3)}\ (a_1 + a_2 - a_3)}{s(s + a_3)}\right] .$$

Let

$$p_1 = \frac{a_1 a_2}{(a_1 + a_2 + a_3)}$$

$$p_2 = a_3$$

$$K_1 = a_1 + a_2 - a_3$$

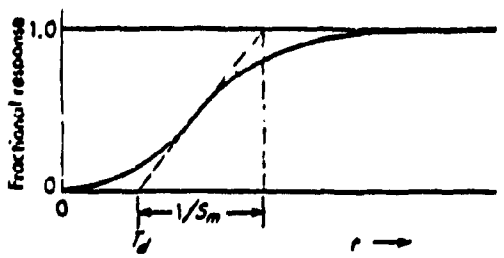
$$G_C(s) = \frac{K_C}{s}\left[1 + \frac{K_1(s + p_1)}{s(s_1 + p_2)}\right].$$

Using partial fraction expansion:

$$G_C(s) = \frac{K_C}{s}\left[1 + \frac{\frac{p_1}{K_1 \quad p_2}}{s} + \left(\frac{-K_1(p_1 - p_2)}{p_2}\right)\left(\frac{1}{s + p_2}\right)\right].$$

APPENDIX C

ZIEGLER-NICHOLS METHOD FOR SELECTING OPTIMUM CONTROL SETTINGS.



T_d = apparent dead time

S_m = maximum slope

Figure C-1. Diagram of Ziegler-Nichols method for selecting optimum control settings. (Reprinted from Reference C-1.)

For PID controllers, the following are generally applicable.

$$K_p = \frac{1.2}{S_m T_d}$$

$$T_I = 2\ T_d$$

$$T_D = \frac{Td}{2}$$

T_I = integral time

T_D = derivative time

K_p = proportional gain

APPENDIX C REFERENCE

C-1. Johnson, E. F. *Automatic Process Control.* New York: McGraw-Hill, 1967, p. 228.

[redacted]

Wayne W. Manges [redacted] He attended elementary schools in and around that city and was graduated from Connellsville Area High School in June 1965. The following fall he entered California State College, California, Pennsylvania. After graduation in the fall of 1969 with a Bachelor of Science degree in Education (chemistry), he accepted a position as teacher and science department head at Avella Area Junior Senior High School in Avella, Pennsylvania. In 1971 he accepted a National Science Foundation fellowship to attend Rennselaer Polytechnic Institute in Troy, New York. After four summers he obtained a Master of Science degree in Natural Science (chemistry and physics).

In June of 1976 the author elected to end his seven-year career in education and entered the University of Pittsburgh School of Electrical Engineering in Pittsburgh, Pennsylvania, graduating with a Bachelor of Science degree in Electrical Engineering in the fall of 1977. At that time he accepted a position on the staff of the Instrumentation and Controls Division at Oak Ridge National Laboratory in Oak Ridge, Tennessee, where he has remained through the present.

He entered The University of Tennessee, Knoxville, in the fall of 1978. He received the Master of Science degree in Electrical Engineering in March 1984.

The author is a member of The Institute of Electrical and Electronics Engineers and the IEEE Computer Society. Mr. Manges will continue in his position at Oak Ridge National Laboratory after graduation.

He is married to the former Pamela Herbert of Connellsville, Pennsylvania.