DISTRIBUTION OF INTELLIGENCE AND INPUT/OUTPUT IN DATA ACQUISITION SYSTEMS

CHARLES W. ROSE
Case Western Reserve University
Cleveland, Ohio

JAMES D. SCHOEFFLER
Cleveland State University
Cleveland, Ohio

Summary

Low cost, high performance microprocessors are being used in several distributed intelligence architecture to replace conventional data acquisition systems. Each distributed topology has certain attributes which affect its suitably for data acquisition applications: cost and position modularity, behavior in the presence of a fault, logical complexity, and bottlenecking.

The software organization of these systems is impacted by the distribution of intelligence and input/output, particularly in the areas of task communication, error recovery, data base management, and operating systems structure.

A ring structure has been developed for data acquisition and control, which uses distributed microprocessor intelligence and modified serial CAMAC protocol. A general purpose instrument and communications control module based upon INTEL 3000 microprocessors has been built.

Introduction

The computational power of the microprocessor combined with its low cost is already having a large impact on instrumentation used for data acquisition and control as new functions and instrument enhancements are added using the microprocessor capability to eliminate digital logic and reduce data for better presentation or control, Such “smart” sensors and instruments also make system design more attractive because error recovery and data integrity can be accommodated much more easily. Data acquisition and control needs lead to an interconnection of sensors, actuators, and controllers. The microprocessor is having an even more significant impact here as its computational power is used to

implement truly distributed systems. That is, both the data acquisition task itself and the task of communication among devices may be assigned to microprocessors. Hence its impact is double: first, increased capability in instruments, sensors, and dedicated control systems; and second, as a communication device to link such intelligent components into a distributed data acquisition and control system.

The most significant questions about this technology are concerned with reliability, error recovery, allocation of tasks to components in the distributed system, and the costs of implementation and maintenance. Furthermore, ease of use of such systems in the variety of industrial applications they must serve is equally important. Since all of these questions are most impacted by the structure of the distributed system, it is appropriate to examine alternative structures for distributed systems and evaluate the impact of the structure on the important questions listed above.

Distributed Systems-Taxonomy And Analysis

The Need for a Taxonomy

Distributed systems are new phenomena, and since few systems have been put into service and studied, few insights into the relative merits of the various topological possibilities have been developed. Each topology has its advocates, but without a common framework for discussion, evaluators are reduced to discussing the specifics of a few given realizations rather than investigating the general characteristics of classes of topologies.

Recently, however, Anderson and Jensen [AND 75] proposed a taxonomy of computer interconnection which is a valuable first step toward the classification of distributed systems by topology and functionality. We shall use this taxonomy as a framework for our analysis of distributed architectures for data-acquisition.

Introduction To and Use of the Taxonomy

The Anderson/Jensen taxonomy is shown in Figure 1, The taxonomy describes three hardware archtypes: processing elements (Ms), paths between them, and switching elements or intervening intelligence. For our purposes, we will assume that sensors and system-level peripherals in any data acquisition system will be controlled by processing elements (CPU’s or microprocessors), and can thus be considered as processing elements.

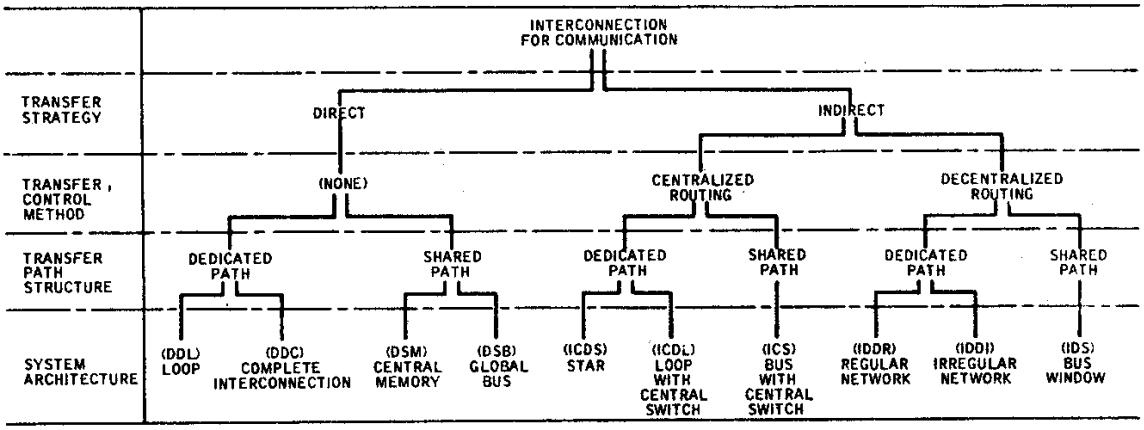

FIGURE 1. The taxonomy.

Transfer Strategy refers to the presence or absence of a switching element between PE's which can alter or switch the flow of a message. Direct transfer strategy implies that no such intelligence is present.

The next level is present only for Indirect systems and defines the switching method. A single Centralized switch or a Decentralized distributed routing scheme with several switches may be employed.

The third level defines the nature of the paths themselves -- Dedicated (non-shared) or Shared.

The lowest level refines the actual system architecture in terms of ten topologies which represent the basic classes of distributed architecture.

The taxonomy is concerned only with the topology of interconnection, and evaluates the topologies with respect to a set of broad attributes which are application-independent:

1. Cost Modularity. This attribute refers to the incremental cost of adding an additional node (processing element or I/O sensor/ device) to the topology.

2. Place Modularity. This attribute refers to the degree to which an incrementally added element is restricted in location and function.

3. Connection Flexibility. This attribute refers to the alternatives avaifiblefo`rconnecting an incremental element.

4. Failure Effect. This attribute refers to the degree of fault tolerance of the topology with respect to path, switching elements, and nodes.

5. Failure Reconfiguration. This attribute refers to the ease with which a distributed system can be reconfigured to mask a failure to allow continued operation in a degraded mode.

6. Bottlenecks. This attribute refers to the possible performance degradation arising from non-uniform flow of communication in a system or from saturation of a shared resource.

7. Logical Complexity. This attribute refers to the totality of decision which must be made during communications, including those made by source and destination processes and by switching entities.

The advantage of this taxonomy is that in many cases, these attributes can be specified for entire classes of systems (i.e., fairly high in the tree). For example, all DXX systems have chosen the poor cost modularity of dedicated paths over the risk of bottlenecking in shared paths, while IXX systems have made the opposite choice. Whether such choices are, in fact, justifiable in an instance, is a function of the requirements of the application.

Distributed Topologies for Data Acquisition

Data acquisition applications impose certain general constraints upon the topologies of the systems which implement them.

First, since it may be desirable to add a sensor or display at any point within the system, cost and place modularity are important attributes. It must be inexpensive to add the PE in terms of interfacing, and its location and functionality must not be limited.

Second, the minimization of failure effect is important. A single sensor/PE failure must not bring down the system, and a single path or switch failure should not either.

Third, these systems should be easily reconfigurable to mask a fault and allow gracefully degraded operation.

Fourth, since data rates and an estimate of the maximum number of sensors are usually known a priori, bottleneck can usually be avoided by picking an appropriate design for an appropriate level of performance. The amount of local processing possible in the sensor/PE will also tend to reduce the probability of a bottleneck in a communications resource.

Fifth, since the sensor/PE's will tend to be homogeneous, communication flexibility tends to be fairly unimportant.

Finally, since the motivation for distributed systems in data acquisition system is low cost and high performance (fast response time), excessive logical complexity is to be avoided.

Conventional data acquisition systems like that shown in Figure 2 display several of these characteristics. Analog and digital sensors may be added anywhere in the system at the expense of the wire and multiplexor port. When the ports fill, however, the incremental cost of the next sensor is high. Remote sensors may be added at an existing site in the same manner. New sites require communications gear and a communications line which is a high cost.

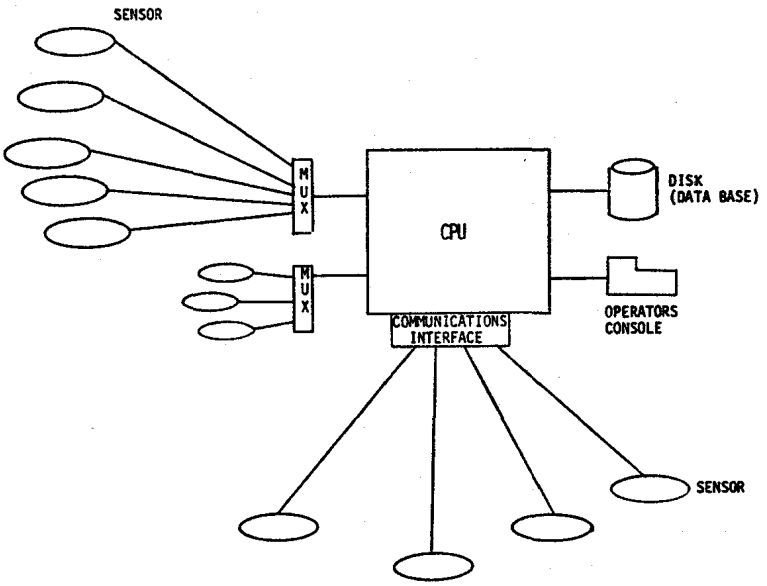

FIGURE 2 Conventional Data Acquisition System

Unless the central computer or multiplexor fail, the system is not vulnerable to catastrophic failure, and reconfiguration is degenerate. A redundant multiplexor could be added.

The system is very simple from a logical and communications standpoint.

Which distributed architectures provide the same characteristics while offering increased performance and potentially better reliability at reasonable cost by exploiting the flexibility of the microprocessor?

Data acquisition systems are distinguished in that every sensor/PE is not required to communicate with every other PE. In fact, most communicate directly with only one Or perhaps two (a file control PE and an operator communication PE, perhaps) as shown in Figure 3.

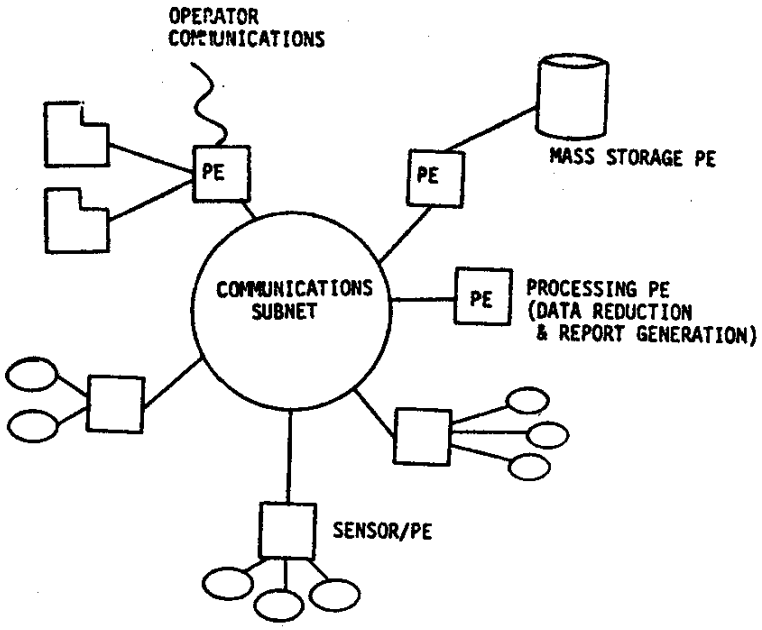

FIGURE 3 Generalized Distributed Data Acquisition System

Message routing, therefore, tends to be fixed and in most cases an Indirect transfer scheme is not warranted because of cost and logical com.plexity. We shall concentrate on Direct transfer systems and look later at two specific indirect schemes of possible interest.

Direct-Dedicated Systems (DDX)

The two DDX systems, the loop (DDL) and complete interconnection (DDQ have very different characteristics which arise from the numbers of paths in the system.

The DDC topology is equivalent to that of conventional star connected data acquisition systems in which a sensor communicates only with those elements with which it shares data (the central CPU).

The DDC topology has a poor cost modularity since the addition of the nth PE requires n-1 communication interfaces and links which must be accommodated at each of the existing sites. In a data acquisition environment, in general, there is not a requirement for all sensor/PE's to communicate and therefore the cost is correspondingly less, but still high. The cost of cable installation alone in large factory data acquisition systems is significant, and the multiplexor costs are a sizable portion of the hardware investment.

Place modularity is good, and the failure effect and failure reconfiguration is quite good, unless of course, the processing or mass storage resource fails. Providing redundancy for these resources is a costly and difficult job under a DDC topology.

The topology is free from bottlenecks except at the processing and mass storage nodes, but the presence of microprocessors at the sensor will reduce the possibility of exceeding the bandwidth of these resources.

Although the logical complexity of this topology is simple, the installation cost of large systems is a major disadvantage. In fact, it is precisely this cost which has stimulated interest in the newer line-sharing distributed architectures for data acquisition and control.

The DDL [FARB 72] [AEC 73] topology or loop, on the other hand, has good cost and place modularity. A sensor/PE may be placed anywhere on the loop with a simple interface and communicate with every other node. Since messages are passed from node to node successively, the failure of a single node or path between two nodes can bring down the entire ring. Thus, for unidirectional, active repeater loops, the failure effect and failure reconfiguration attributes are poor. Passive repeater systems, in which each node simply looks at each message as it goes by to determine whether it is addressed to that node, are less vulnerable to PE failure, and redundant path-loop collapsing mechanisms may be added to improve the graceful degradation of a loop. These additions increase the cost, of course.

The logical complexity is quite low. This topology,is subject to bottlenecks if the message traffic between adjacent or nearly nodes on the loop is high for sustained periods.

The primary advantage of this topology is the cost Csince most loops are bit or byte serial) and the high modularity. For passive repeater loops, a PE failure can be masked by load sharing among the remaining elements if process or task addresses are soft; that is, if they are not bound to one processor but are kept in a table and checked by the communications software in each PE.

Direct Shared Systems (DSX).

The direct shared memory, DSM, topology is the classic shared memory multiprocessor architecture and is of no interest here.

The Direct shared bus, DSB, or global bus, is very attractive for miCToprocessor-based data acquisition systems. Access to the bus is shared by the PE’s by some allocation scheme and messages are sent directly to the
destination node.

Cost and place modularity for PE’s is good. Elements connect to the bus via a standard interface, and it may be possible to connect the PE anywhere on the bus.

Cost and place modularity of the communications bus itself is poor in that the total bandwidth is fixed and increasing it would require extensive changes to PE interfaces and perhaps to the communication medium itself. However, for data acquisition systems, acceptable bandwidth (to avoid bottlenecks) can be provided, if at all, at design time.

Failure effect for PE's is good, provided that the failure does not hang up the bus. If soft process addresses are used, the dropped load may be picked up by other processors.

As with DDL topologies the failure characteristics of the bus itself are poor. Replication may be costly depending upon the width of the bus.

The primary advantages of this topology for data acquisition are the cost and place modularity of PE's, and PE failure characteristics.

It is clear, therefore, that of the DX topologies, DDL and DSB are the principle candidates for data acquisition systems because of cost and place modularity for PE's, low overall cost arising from little logical complexity, and good fault reconfiguration characteristics for PE's (this is only true for passive repeater loops, however) in the presence of soft process addresses.

Their disadvantage lies in the vulnerability of the topologies to communication path failures. Redundancy of paths can mitigate this disadvantage, particularly in bit or byte serial systems.

Indirect Topologies IXXX.

Of the indirect topologies, only the centrally switched loop and bus are of possible interest.

All of the distributed routing systems are too complex logically, and reconfiguration to allow load sharing requires extensive modification of the router tables or programs. In general, the cost/performance of these topologies for data acquisition is not acceptable.

Since there is not a high degree of switching in a data acquisition system, the investment in a central star switch is not warranted, and its vulnerability is high. In fact, if the switch were a PE, the topology would resemble that of a conventional system with its attendant disadvantages as shown in Figure 2.

The centrally switched loop and bus have characteristics similar to their direct counterparts with the exception that the central switch can be used to logically reconfigure the system in the event of PE failure by rerouting messages destined for a failed PE to ones capable of handling its function. This is equivalent to the soft process address technique described

earlier, but the function is implemented once, thus reducing the complexity of each PE interface. The penalty, of course, is poor failure effect in the event of a switch failure, and redundant switches will probably cost more than the soft process address scheme. It is reasonable, therefore, to reject these topologies for vulnerability reasons.

Indicative of the suitability of DDL and DSB topologies for distributed data acquisition and control systems is the fact that of those systems commercially available or proposed as standards, all utilize one of the two topologies [ARON 71].

For example, the CAMAC standard [COST 72] developed for nuclear instrumentation is a DSB system, and a DDL serial CAMAC standard [AEC 73] has been proposed.

The IEEE standard 488-1975 Coriginally Hewlett-Packard) [KNOB 75] is a DSB as are FOXBORO's INTERSPEC [FOX 73] and Honeywell's line sharing system [BAIL 76].

The Honeywell system, shown in Figure 4, is a hybrid since all PE's and devices residing on branch busses are reached via the switch which repeats the message on all branches. Main bus communication is carried out in a standard DSB manner.

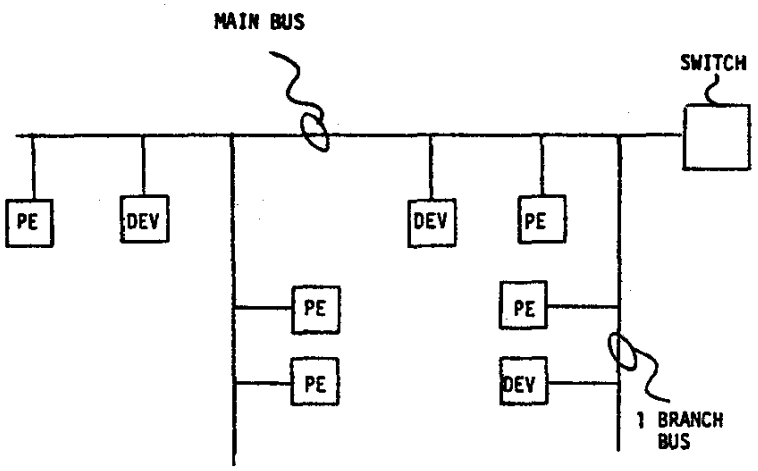

FIGURE 4 Honeywell Linesharing System.

An additional motivation for these standard communication schemes for distributed topologies is that devices and instrument manufacturers are encouraged to provide standard interface to them, thus allowing cheaper systems. Integration of microprocessors into instruments simplifies the task of providing multiple standard interfaces from a protocol standpoint.

Effects of Distribution On Data Acquisition System Design

Figure 3 pictured a generalized distributed data acquisition environment with the topology of distribution unspecified. Given the similar characteristics of the acceptable topologies, DDL and DSB, it is possible to analyze the effect of distribution on data acquisition systems while treating the topology as a “black box.”

These effects are primarily software and performance related and include: the interprocess and operating system communications interface, handling of a distributed data base, error recovery and graceful degradation, and response time and throughput.

Interprogram Communication and Operating System Interface

One form of intertask communication has been proposed which appears to be very adaptable to distributed multicomputer systems [HANS 69]. All communication involves transmission of data and synchronization through the exchange of messages between tasks (including tasks and the operating system and tasks and I/O devices). It is attractive because this is also the natural approach for task and I/O communication in a distributed system.

In this scheme, data is communicated by inserting it into a message which is then sent to the task needing the data. For example, a task wanting data from a common data base sends a message to one task responsible for maintaining that data base with a request for specific data. This data is returned in a reply message to the requesting task. Since all accesses to the data base are through one task, the problem of simultaneous access is essentially eliminated and moreover access is completely under the control of the one task.

Of course such a system must be very efficient for otherwise the overhead involved in message exchange might create delays which are intolerable
in a realtime system.

A set of primitives associated with this scheme allow synchronization of multiple tasks, prevention of lock out of high priority tasks by lower priority tasks, and release of resources in the event of premature task abortion.

When processes are co-located in the same PE, the message discipline requires only the source and destination process names. This approach causes several serious problems where the processes are not co-located. The program must know the eventual location of each process in the system, and, thus the topology becomes bolted into the application software. The cost of application software is usually the dominate cost in real time data acquisition and control systems, and thus such inflexibility is passed on to the user in terms

of a non-optimal topology/processor mix or large non-recurring software costs. Furthermore, as we shall see later, this program-writing time binding of process to processor precludes dynamic reconfiguration and load sharing under failure of a processor, and thus adversely affects graceful degradation.

The approach of choice is to assume that no two processes are co-located and to force all communications to use the distributed protocols, i.e. communication by unique,process name and message. The actual process distribution is defined by process tables in each operating system or in the hardware communications interface. Messages to local processes are simply turned around in the protocol handier and passed back while messages to remote processes are transmitted over the DDL or DSB. Thus, the configuration is quite flexible; binding occurs at systems generation time, and dynamic reconfiguration is possible if the I/O devices are distributed rather than dedicated to each PE.

Data Base Considerations

The problems of maintaining physically and logically distributed data bases are well known, if not solved. In these applications in which processes must access and update remote data in a real time manner, files or records may be locked to avoid concurrent updates or interference. When an entire file or record is transferred to a remote processor for update and eventual rewrite, or when update data is accrued remotely over an extended period, the lock out scheme is unworkable. To deny the real time processes access to the existing data for long periods of time is not acceptable, but to allow processes to compute and act on data which is no longer current may also not be acceptable.

Transferring raw data to the site of the data base and computing updates there defeats the purpose of remote intelligence and is likely to place a heavy load on the communications facility.

There are applications in which distributed data base maintenance is easily handled and in which distributed intelligence is quite meaningful. Figure 3 could represent a data acquisition environment in which the distributed processors acquire data, convert to engineering units, alarm check and reduce data in their own local data bases. Only summary information or alarm conditions are transmitted to the processor and/or data base processor which controls the logger and operator communications. In this case, each local data base is independent of the global history file.

Another application occurs in a production control environment in which local data bases are interrogated periodically to update the global data base, and changes in the global base are then transmitted to the remote systems affected.

There is a need for a general solution to these problems which allows a user to tailor the distributed data base to his configuration and application. The microprocessor may help considerably in this area because of its ability to perform complex computational tasks at low cost and consequently may be dedicated to such tasks as management of data base problems.

Error Recovery. In conventional systems, input/output transfers are completed using a handshaking synchronization scheme, and data errors on the bus are very infrequent. In distributed I/O systems, however, errors are more likely, and thus error detection and correction schemes employed in data communications systems must be considered. Error correcting codes often occupy too great a fraction of the message bandwidth to be cost effective, and for most systems and environments, retransmission of erroneous messages is the scheme of choice.

Standard retransmission schemes exist which prevent lost or duplicate messages in the event of communication errors [CAM 75). This is extremely important in data acquisition applications in which counting of events is critical or in the control of a stepping motor.

The recovery from processor element crashes is simplified somewhat by the presence of microprocessors at all nodes. Processor crashes can be detected by other processors as a result of time out conventions on messages and the passing of “are you healthy messages?” Thus, hot restart can be effected by resetting all message synchronization bits to their initial states after the crashed PE is reinitialized. Note that this solution is valid only in passive repeater DDL’s, since a crashed PE interface to the loop will disrupt communications around the loop in an active repeater system.

Graceful Degradation. This attribute is equivalent to failure reconfiguration and the distributed data acquisition system is constrained to the value of that attribute for communication path failures. For processing element failures, the same topological constraints obtain, but the software organization and distribution of resources determine the extent to which the potential is realized.

For example, in Figure 3, the failure of a sensor/PE would disable input from those sensors controlled by the PE. However, if the processing site were to fail, its load could be picked up by the Operator communications PE if the following conditions held: [1] all input/output devices needed by the processing site were distributed; (2) the operator communication PE held a copy of the necessary tasks and had sufficient processing capability; and finally (3) if soft process addressing were employed so that messages to the processing site tasks would be routed to those tasks in the alternate processor.

To the extent that I/O devices are not bound to a specific PE and that soft addresses are employed, the potential for failure masking and reconfiguration may be realized.

Throughput and Response Time

The throughput of individual processors in a distributed intelligence system will tend to improve over uniprocessor multiprogrammed systems because extensive operating system overhead and context switching time can be avoided by actual concurrency in the system. System throughput will degrade, however, if there is a significant amount of interprocess communication among processes which are not co-located because of the communication delays.

Response time to external events will be at least as good as uniprocessor environments and perhaps better, since a simple operating system is likely to have a much shorter response time than sophisticated real time multi-tasking systems (50 users compared to 200-300 users). Of course, if the response processing is located remote from the processor to which the stimulus is attached, response time will degrade significantly, but this situation is to be avoided if at all possible and microprocessor; based sensor/PE's will help.

Peripherals cannot use conventional methods to interrupt processors in a distributed I/O environment. Instead, demand or LOOK AT ME messages must be transmitted to the appropriate PE after which a data request message is sent to the peripheral. A message, rather than single instruction I/O discipline and its attendant protocol overhead will slow down all I/O. The alternative, dedicating the peripheral to a PE may preclude graceful degradation in the event the PE itself fails.

Summary

We see then that the availability of microprocessors and the good modularity and reconfigurability of DSB and passive repeater DDL topologies can provide high performance data acquisition systems which better satisfy the requirements of these applications than conventional systems at a lower cost.

A Microprocessor-based DDL Topology for Data Acqusition and Control

To achieve the advantages of distributed data acquisition and control in industrial applications, it appears that a system will be composed of modules dedicated to a limited function which in turn are interconnected and communicate with one another so as to act like an integrated system. Since the nodes in such a system will vary from devices through microcomputers to minicomputers, the structure for the interconnection is important if simplicity and reliability is to be achieved [Schof 76].

As an example of the microprocessor and its use in such a system, one distributed system has been designed and tested [Linn 75] [Schof 75]. This consists of a ring connected network (DDL) of modules each of which may be a computer, microprocessor, smart module, or simple I/O device. Because more than one computer exists on the ring and because more than one device initiates communication, all messages are designed to be general in that both source and destination address space is provided. Hence there is no need for one device to act as the central message communication machine. In order to maximize the simplicity of solving the device contention problem, a delay-insertion protocol was selected for the ring. This further insures that no device can monopolize the communication bandwidth of the ring.

The message structure is shown in Figure 5 where messages corresponding to commands, replies, interrupts, and blocks of data transferred are shown. Message lengths are variable in the system to accomodate special devices existing in the system, different modes of communication, and ease of accepting future additions to the system.

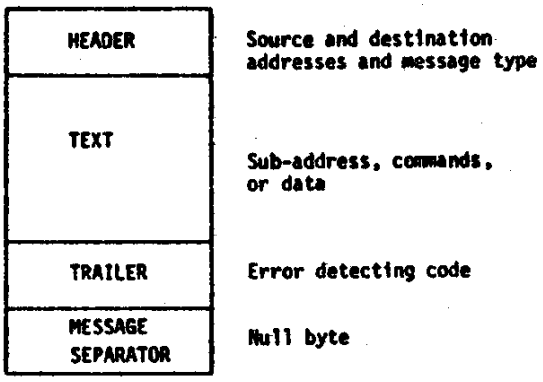

Figure 5

Message Structures

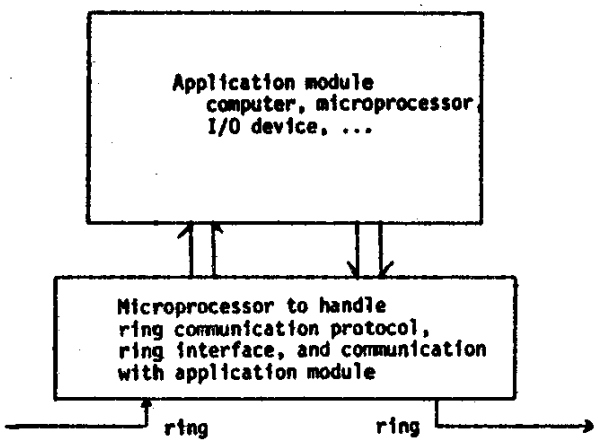

Figure 6
Use of a microprocessor
In a node

The ring network uses the delay-insertion protocol mentioned in the previous section. Thus a node may transmit a message only when it does not have its message buffer already switched into the ring. It then sets up the message in the buffer, switches the buffer in and then acts normally thereafter until it detects a buffer full of null characters, at which time it removes the buffer. This has both advantages and disadvantages, the former being the natural limiting of the rate of transmission of messages by any one node and the latter being the delay time incurred in retriev ng the buffer in case of heavy utilization of the ring. That is, in a critical time when message traffic is heavy, there may be long delays in finding a sequence of null characters which permits the retrieving of the buffer so that a sequence of critical messages from one node may be delayed intolerably.

This can be alleviated by modifying the delay-insertion protocol to a “shrinking buffer” alternative. In this case, individual bytes in the buffer are retrieved whenever a redundant character is detected entering a node. Thus a buffer with 15 characters could be retrieved by detecting a number of short redundant sequences (null character sequences) rather than one null sequence of length 15. The improvement in line utilization, throughput, and response time is dramatic with this change in communication protocol.

Of most concern here is the use of the microprocessor as the ring interface and communication controller. This software has several functions: monitor and pass messages through a node (which contains a one byte buffer); detection of messages destined to the node; insertion of messages onto the node; maintenance of a single message buffer within the node used in the delay insertion protocol; plus handling of the global addressing of virtual addresses in the system.

The structure of a node on the network is then as shown in Figure 6. The microprocessor-based ring interface is responsible for both the communication protocol and the interfacing to application programs in a computer (micro or mini-computer) or to modules connected to the ring (set of analog input/output modules for example). The extent to which special purpose microprocessor I/O hardware would be necessary for the interface is dependent upon the speed of communication around the ring.

The ring transmits bits within a byte synchronously, and messages (a sequence of bytes asynchronously. To facilitate a configuration independent software system, a virtual address system is provided. An address space is divided into pages (128 bytes/page) and this space addressed by page number and displacement. Pages may reside in any microprocessor or computer on the ring and may be moved upon demand. Software in the communication microprocessor at each node permits application programs to access data on any page, request pages, lock pages in place, transmit pages, etc. Application software can thus be created using data base stored on these pages along with standard software modules also stored on pages and later distributed around the system with no change to the software. A by-product of this organization is that all long messages are standard in that they consist of transmission of a page of the virtual space. This simplifies the communication software in the interface microprocessor.

The extensive byte manipulation requirements of the communication application are much better met in later generation microprocessors. Normal length command and control messages are less than 20 bytes in length and correspond then to less than 200 bits per message (some are much less). Transmission of a 128 byte page in the long messages corresponds to approximately 1300 bits. With a fast microprocessor acting as the ring interface, approximately 1000 command and control messages per second or approximately 100 long page transfer messages per second can be transmitted. In a typical

application, the two mix of course and vary from second to second. Nonetheless, the combination of the messages leads to a bit rate for transmission of less than 200 kilobits/second. It is quite feasible to raise this rate to a more or less standard rate of 250 kilobits/second with little effort and to a megabit/second or more with a hardware assist in the interface (the microprocessor computing rate cannot respond this quickly unless even faster microprocessors were used. Nonetheless, the data rates are quite within the current technological limits. In evaluation of one application involving a great deal of data acquisition, control, and operator communication, line utilizations of around 40-50% were found to adequately satisfy the application needs and these were well within the capability of the distributed microprocessor network above. Furthermore, it has been found possible to implement microprocessor interfaces without special hardware which can respond quickly enough to satisfy currently available distributed I/O speed specifications.

The ring structure described above is a modification of serial CAMAC [AEC 73] and serves as the serial highway as shown in Figure 7. Current research underway at Case Western Reserve University is focused on discovering to what extent a single microprogrammable module based upon INTEL 3000 microprocessors can realize all of the modified CAMAC control and interface functions [BAK 76]:

1) Mastery of the Serial Highway'(Serial Driver).
2) Serial Highway Interface (Direct Connection of an instrument and/or controller to the serial highway).
3) Mastery of the Data Highway (Crate Controller).
4) Instrumentation Interface (Crate Module).
5) Interface of an arbitrary Processing Element to either the Serial or Data Highway.

The design of the dataway interface is complete. The significant elements of the design are: use of a bit sliced microprocessor, a pipeline architecture, a vector interrupt scheme, and a micro program suspendable clock. Each of these elements of the design was necessitated by the timing standard for dataway operations. A complete operation including address decoding, function decoding, and data removal from or placement onto the bus must take no longer than 1 microsecond.

The dataway read and write data paths are each 24 bits wide, and one of the standard 8 bit microprocessors would require three I/O instruction sequences to read or write on the dataway. This design contains a 24 bit wide implementation of the Intel 3000 microprocessor system; this allows the dataway to be read or written with a single micro instruction.

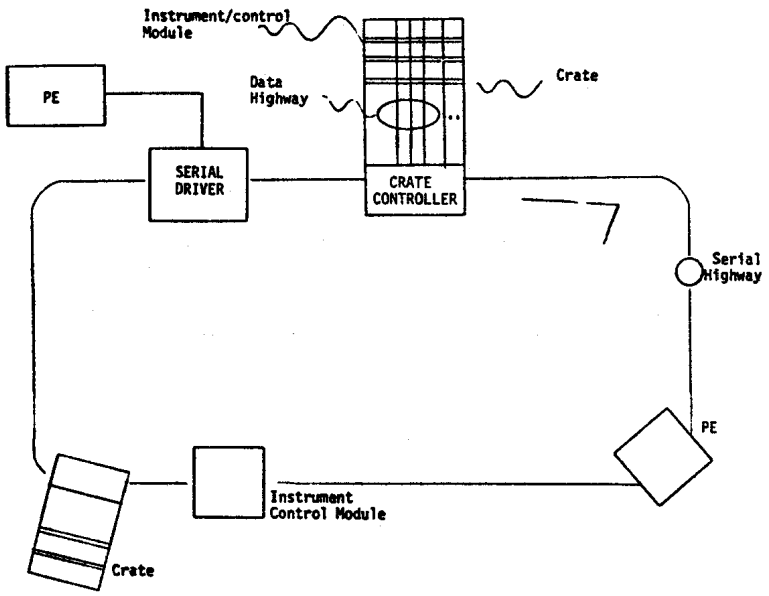

FIGURE 7 Modified CAMC Linesharing Scheme

To reduce the micro instruction cycle to its minimum. A pipelined architecture has been chosen. The Intel 3000 system allows an implementation in which the determination of the next micro instruction can occur simultaneously with the execution of data transfers or ALU functions. This is accomplished by adding a register of edge triggered D flip flops, the pipeline, to the standard Intel 3000 system.

The design of an instrumentation interface is also complete. This interface provides for both input and output connections to digital TTL level signals in measurement or control instrumentation. All Intel 3000 CP functions can operate on this instrumentation interface. This, therefore, allows a micro program to read, write, clear, selectively set, and selectively clear signals in the instrumentation. The instrumentation can access parallel outputs, signal the microprocessor, and enter parallel inputs.

An instrument control module combining the dataway interface and the instrumentation interface has been designed. At present this design includes two identical instrumentation interfaces and during the module, execution of a dataway command the dataway address line, Al, selects between them. This design seems the most logical consequence of the CAMAC standard's specification of command operations. However, a change to this design is being considered which would place a bidirectional data port in one of the two instrumentation interfaces. The intention of such a change is to create an interfacing facility which is both a status/control interface and a data interface. This change will allow the instrumentation control module to interface to the Unibus or a floppy disk as readily as it now interfaces to A to D or D to A converters. Figure 8 shows the block diagram of this instrument control module.

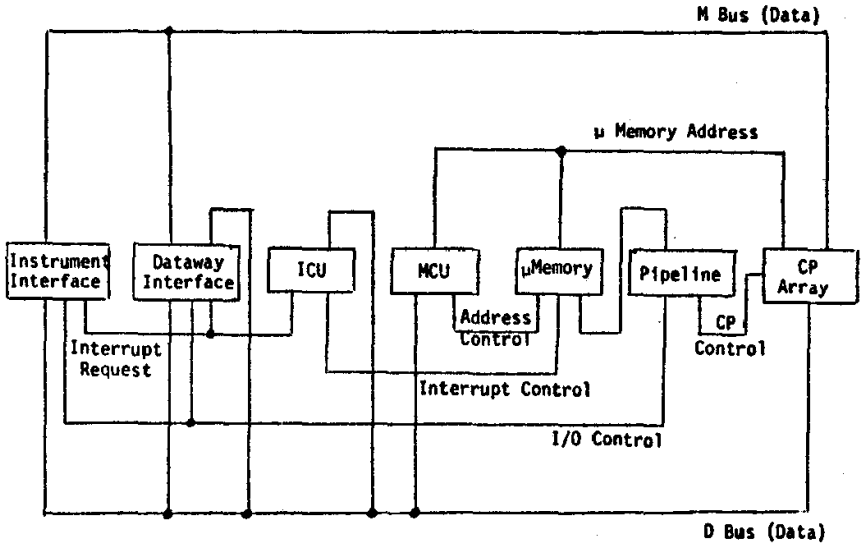

Figure 8
Block Diagram of an Instrument Control Module

This design is being implemented experimentally by substituting the main memory of a TI 960 computer for the module's micro processor memory. With such an implementation, the interfacing abilities of this instrumentation control module can be easily verified. The completed design and planned implementation focuses on two of the six functional elements of a CAMAC system: the dataway interface and instrumentation interface. Control of these two interfaces is readily accomplished with an Intel 3000 microprocessor system, and the instrumentation control module design which combines these two interfaces is a flexible and useful tool.

It is hoped that by microprogram modification and by slightly modifying the input/output structure, the remaining three functions can also be realized using these standard building blocks. Since the data highway timing is the most critical in the system, the INTEL 3000 microprocessors should handle the remaining tasks easily.

Conclusions

The microprocessor is such a versatile element that the attractiveness of distributed data acquisition and control systems for industrial applications seems a certainty. Furthermore, such systems will strip some of the faster, more time critical, more reliability-critical applications out of process control computer systems. The net result will be a simplification of process control computer systems together with an extension of economical applications.

References

[AEC 73] AEC/NIM and ESONE, "CAMAC Serial System Organization, National Tech. Information Service, TID--26488, December 1973; "Addendum and Errata"; May 1975.

[AND 75] G.A. Anderson and E. D. Jensen, "Computer interconnection structures," Computer Surveys, vol. 7, No. 4, pp. 197-213; December 1975.

[ARON 71] R.L. Aronson, "Live sharing systems for plant monitoring and central," Control Engineering; January 1971.

[Bail 76] S.J. Bailey, "Multiplexor: distributed control design component," Control Engineering, Vol. 23, No. 4, pp. 32-36; April 1976.

[Bak 76] C. Baker, Forthcoming Master's Thesis, Case Western Reserve University; 1976.

[CAM 75] F.I. Camerer, "A full duplex communications algorithm," A. R. Jennings Computer Center, Report No. 1164; February 1975.

[COST 72] L. Costrell, "CAMAC - A Modular Instrumentation System for Data Handling; Revised Description and Specification," National Technical Information Service, TID-25875; July 1972.

[Farb 72] D. L. Farher and K. C. Larson, "The system architecture of the distributed computer system," Proc. Symposium on Computer Communications Networks and Teletraffic, Polytechnic Press, Brooklyn, N.Y.; 1972.

[Fox 73] Foxboro Co., "Interspec technical information." Bulletins 2DC-100, 2DN-110, 240-100, 240-101, and 240-102, Foxboro, Mass.; 1973.

[Hans, 69] P.B. Hansen, "RC 4000 Software multiprogramming systems," A/S Regnecentralen, Copenhage, Denmark; April 1969.

[Knob 75] D.E. Knohloch et al, "Insight into interfacing," IEEE Spectrum; May 197S.

[Lin 75] E.Y. Linn, J.D. Schoeffler and C.W. Rose, "Distributed Microcomputer Data Acquisition." Instrumentation Tech., Vol. 22, No. 1, pp. 55-63; January 1975.

[RUSS 75] R.D. Russell and P. Sparrman, “A technique for integrating remote minicomputer into a central computer’s file system,” IFAC Realtime Programming Workshop, Baton Mass.; August 1975.

[Schof 75] J.D. Schoeffler, M. Haelsy and C.W. Rose, “Microprocessorbased communication and instrument control for distributed control systems,” Proc. National Electronics Conference, pp. 270-276; October 1975.

[Schof 76] J.D. Schoeffler and C.W. Rose, “High speed microprocessorcontrolled instrumentation and communications systems,” IEEE Transactions on Nuclear Science, Vol. 23, No. 1, pp. 3855; January 1976.