EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN - SPS DIVISION

CERN LIBRARIES, GENEVA

CERN SPS/83-14 (ACC)

CM-P00064380

REPLACING MINI-COMPUTERS BY MULTI-MICROPROCESSORS
FOR THE LEP CONTROL SYSTEM

J. Altaber, M.C. Crowley-Milling, P.G. Innocenti and R. Rausch,

Abstract

At the CERN SPS accelerator we have used a control strategy involving the use of a network of multi-tasking computers, each having a functional or a geographical task. Control is performed by executing interpretive programs in various computers, some procedures requiring the collaboration of several computers. The experience being very successful, the same lines will be followed for the LEP control system, making use of present day technology. Each multi-tasking computer will be replaced by an assembly of microprocessor based modules with each performing a unique, single stream type of task. Each module will contain a micro-CPU, private memory, and I/O driver for communicating via messages, the medium being a multi-master parallel bus. The paper describes the module tasks which have been identified and the relationship between modules. The improvement in performance, minimisation of software effort required, as well as the reduction in cost of such an arrangement are also discussed.

Paper presented at the 1983 Particle Accelerator Conference, Santa Fe,
21-23 March, 1983.

Geneva, March, 1983

REPLACING MINI-COMPUTERS BY MULTI-MICROPROCESSORS FOR THE LEP CONTROL SYSTEM

J. Altaber, M.C. Crowley-Milling, P.G. Innocenti and R. Rausch,
C.E.R.N., CH - 1211 Geneva 23, Switzerland

Introduction

The control system for the CERN SPS machine has pioneered a control strategy which has expanded the usual central control methodology not only spatially among computers but also organisationally among people. Each fragment resulting from this "controlled explosion" is a mini-computer containing a message exchange package which loosely couples the fragments, a multi-tasking monitor and an interpreter. A small number of professional programmers have constructed a system which numerous engineers and technicians have been employing both for the day-to-day operation of the machine and for developing experimental multi-computer control procedures.

The same strategy will be used for the LEP machine but a further expansion will be implemented breaking down the mini-computer, basic construction block of the SPS control system, into clusters of loosely coupled micro-computer based units. Apart from the availability of the technology, this expansion is feasible because of the underlying constructional principles of the SPS system which will be transported intact to the LEP system. The paper will describe the problem of organising the clusters analogue to a mini-computer running tasks interacting with real-time events; it will be shown that simple hardware and software solutions exist for the problems that arise, and in addition it is possible to organize a uniform data flow through the system.

The essential organisational levels

The LEP control system from a constructional point of view will be organized into three levels centered on Process Control Assembly (PCA) made from the micro-computer based units :

- network : file and program exchange between PCA,
- process : execution of program within a PCA,
- equipment : access to equipment connected to a PCA.

Network

At the higher level, the network will be organized in an arrangement offering a single route for a datagram flowing from source to destination PCA. Such an arrangement guarantees the sequencing of the datagrams through the network, provided that any specific queuing in the network is made as a FIFO. For the SPS control system a multi-star, datagrams switching, store and forward architecture has been used. The same architecture can be extended to cover the LEP network solving de facto the necessary bridging between the LEP-SPS control system. Although this solution is very attractive we are looking into other possible architectures which both on time scale and performance could satisfy our requirements and comply with the standards which have been agreed since the SPS system was designed.

The SPS-LEP network can be described very easily in terms of the OSI-ISO model as shown below :

7) Application	Nodal EXECUTE(N)
6) Presentation	Empty
5) Session	Assign session number, form data into block
4) Transport	Add transport header forming packet
3) Network	Add routing header, form into queues, deal with flow control
2) Data link	Send packet as a frame
1) Physical	Transport frame

Fig. 1

The two upper layers (application and presentation) are handled by the process level and being kept identical to the SPS system they will guarantee the compatibility between SPS-LEP systems, provided the necessary gateway exists.

The session, transport, network, data link layers are dealt with by the network level. The physical layer will be based on an integrated communication system described in Ref. 1.

Process

The process level deals with the execution of programs under the supervision of a real-time monitor. There are two essential decisions which have been transported from the SPS control system.

Firstly, the programs will be stored in source code and interpreted for execution, the programming language being NODAL. The external and internal structure of the NODAL have been described in Ref. 3 and for our present purpose it is sufficient to say that it has been tailored to match the two environmental levels :

- Network level. Commands have been incorporated, allowing portions of the source code of an application program to be sent from one PCA to another.
- Equipment level. The language offers the capabilities to invoke equipment drivers which were called "data-modules" in the SPS. From the programmer's point of view it allows explicit addressing of equipment by generic name, equipment number and property rather than the usual interface hardware reference.

Secondly, in a classical multi-programming operating system, the number of tasks wanting to use the central processor at any one time can be very large. Priority algorithms may be used to determine who uses the CPU next, and the latency period for a task with low priority may be great. It has been decided to use for the LEP control system a method by which the tasks to be run into a PCA are organized into classes according to their nature: surveillance, interactive, machine event, remote, operator intervention. At any time there is only one task active for each class, and according to the class a task has a maximum allowable run time. Tasks which cannot be executed at once are queued on the appropriate class queue.

Equipment

The design of the connection to the equipment is influenced by two factors.

Firstly, there are in many cases rows of equipments which are controlled by a PCA, thus a connection of a multidrop type will minimize the cabling. A further reduction in the cabling can be achieved by grouping equipment geographically and functionally into crates.

Secondly, the data module concept mentioned above and which is described in Ref. 1 calls for an organization of this multidrop connection which is mainly of a master multi-slave nature. This does not imply that the equipments are dumb, microprocessors will be integral part of the equipment control; it implies that the action driven by the PCA can be of a high level nature, which implies in turn that the relationship between the equipment and the PCA can be expressed in terms of messages rather than the classical command response mode of operation.

The data flow between the levels

The relationships between the three physical levels are fairly clear from the above description. The process level talks to its two surrounding levels, network and equipment, which never talk directly to each other.

Tasks in the process level sends files, program, remitted data to the network level which in turn transmits it to the appropriate destination through the network.

The network level receives file, program, remitted data from the network and activates the appropriate element in the process level (task, real-time scheduler) to act on the data.

The process level interacts with the equipment level in the form of "data-module call" sending and reading the necessary data. In case of malfunctioning in the equipment the equipment level will activate the real-time scheduler for the launching of program which can act on the malfunction.

Thus it can be seen that, with the data on the network flowing in the form of message, the communication with the equipments being in the form of message as well, then the overall data flow between all the components of the system are of a same type and can be easily handled in a uniform way.

The Process Control Assembly

The above description shows that such a system organization is ideal for multi-processor implementation where each of the jobs to be performed is given an autonomous processing unit.

The processing units

The three levels described will be made of processing units as follows.

The duties of the network level will be performed by a single unit called the data-link unit DLU.

The equipment level will be provided by an equipment directory unit EDU which will control the access to the level, and with one or more highway branch controllers (HBC), one per multidrop highway to be connected.

Fig. 2

The process level requires two types of unit; the Process Supervisory Unit (PSU) which will play the role of a real-time monitor and several NODAL processing units (NPU) each of which can execute autonomously NODAL programs at different priority levels. This gives an additional freedom compared to the priority concept of a multi-tasking environment which can be replaced by the concept of guaranteed availability of a NPU to execute a task. As shown in the example below, a NODAL task can run on the NPU assigned to its class or on any NPU assigned to lower priority classes and which is free.

Fig. 3

Several of the tasks identified above can be carried out by the same hardware: the EDU, NPU, ASU are identical, the difference being the software which is inside. Thus recovery from a unit failure can be achieved by having an extra unit in the crate, normally idle but which can be loaded with the suitable software to replace the failing unit.

The relationship between the processing units

A suitable arrangement to construct a PCA from processing units is to use a crate into which the required units can be plugged. The units are connected by a multi-master system bus which carries the inter-units communications which is in form of message of several "words". To this end one unit must be able to reserve another unit temporarily for its exclusive use. We will call this procedure "reservation". In addition, once such a reservation is established, it is highly desirable that other units should be denied access to the reserved unit. This will be called "protection".

By supplying suitable hardware in each unit, reservation of one unit by another can be the concern only of those two units, as can any subsequent transaction between them. Only in the case of global failure or interference from outside will central action be taken, clearing the situation and restoring any necessary units to their unreserved state.

The building blocks

To keep the developments strictly minimal the modules are designed bearing in mind the largest possible application for each of them. This approach permits to build larger quantities of identical elements leading to lower prices and requires for their maintenance less test equipment, test programs and documentation. The variety of failures is also limited and faults can be cured systematically.

General processing unit (GPU)

The GPU is the basic processor module implemented on a single card which will be used for PSU, NPU and EDU. This processor unit can be associated with an additional card via its private I/O port for example to extend the private memory available to a single processor or to design rapidly a specialised module to be mass produced later as a single card unit. This new module would then only contain the minimal GPU capability required for a specific fraction.

The GPU holds a shared amount of PROM and RAM memory according to the need of its function. For the memory extension the full 24 bit addressing capability of its microprocessor is available on the private I/O port connector on which the DMA mechanism is implemented as well.

As the PCA contains principally GPU's communicating with each other by means of messages each GPU may be master or slave alternatively or both simultaneously. To establish exclusive communication between two GPU's at a time without possible interference from another GPU, a protected slave I/O channel is provided on the GPU card in addition to its master access logic. The channel is hardware protected by a signature mechanism activated by a temporary master at reservation time². Only one temporary master can thus read or write to that channel at any instant. Associated with that channel there is an interrupt register holding the temporary interrupt vector loaded by the current master at reservation time.

The GPU's logic holds two interrupt vector receive sections. One for the awaited interrupt vector generated by its communicating slave channel of another GPU module and one for the non expected interrupts vector generated at random time by any other master inside an assembly. As this second interrupt section has to deal with unexpected interrupt vectors a FIFO buffer with overflow protection is implemented.

To summarize, each GPU module provides a slave channel logic and a master bus access logic permitting fully protected duplex communication amongst pairs of GPU modules.

Data-Link Unit (DLU)

The purpose of the DLU is to establish communication amongst PCA's in the LEP control network. As such a DLU has a full duplex line connection to the network. The architecture presently proposed requires the construction of switching nodes

called Message Handling Assemblies (MHA) which can be built from several DLU's. For this purpose the DLU design will provide full duplex communication capability amongst pairs over the system bus.

Highway Bus Controllers (HBC)

The function of the HBC is to control and supervise the operation of the equipment connected onto the multidrop bus. Each PCA contains one or several HBC modules under the control of the GPU having the Equipment Directory (EDU) function.

The architecture of the HBC differs from that of the GPU or of the DLU. It is mainly a slave module, receiving its commands and messages from the EDU, and becoming system bus master only for the generation of vectorised interrupt cycles. As such an HBC behaves like a protected slave I/O channel the arrangement of which is identical to the one of the GPU's I/O channel.

The protocol of the multidrop bus may be of different types depending on the particular application requirements for general purpose control of equipments in a single master/multiple slave mode the MIL 1553B standard protocol is being considered¹. For connecting measurement instruments to a PCA the IEEE 488 standard protocol is best suited.

While different types of HBC modules may be designed for various applications, for the control of LEP equipment the same command message protocol carried by the multidrop highway will be used¹.

On the system bus side the same protected I/O channel access logic is implemented allowing an homogenous message communication between HBC's, with different line protocols and GPU modules.

References

- 1) M.C. Crowley-Milling, The control system for LEP, 1983 Part. Acc. Conf., March 21-23, Santa Fe.
- 2) J. Altaber, R. Rausch, A multiprocessor bus architecture for the LEP control system, 1983 Particle Acc. Conf., March 21-23, Santa Fe.
- 3) M.C. Crowley-Milling and G. Shering, The NODAL system for the SPS, CERN 78-07.
- 4) J. Altaber, F. Beck, M.C. Crowley-Milling, R. Rausch, Suggested principles for the control of future accelerators, IEEE Trans. on Nucl. Science, Vol. NS-26, No. 3, June 1979.
- 5) J. Altaber, F. Beck, R. Rausch, Accelerator control systems without minicomputers, XIth Int. Conf. on High Energy Accelerators, CERN, Geneva, July 80.
- 6) J. Altaber, F. Beck, M.C. Crowley-Milling, R. Rausch, Truly distributed control using one microprocessor per real-time task, IVth Int. Conf. on Trends in On-line Computer Control Systems, Warwick, April 82.