ESSENTIAL FEATURES OF THE CERN SPS DISTRIBUTED CONTROL SYSTEM *)

J. Altaber, V. Frammery, C. Gareyte, P. van der Stok

CERN, European Organization for Nuclear Research, Geneva, Switzerland

Abstract. The SPS accelerator presents a considerable industrial control problem with the additional complication that the control procedures are never fixed. Right from the beginning it was decided to base the control system on a distributed network making use of an interpretive language for the control processes. The success of these decisions can be seen from the fact that, over the last six years, the system has grown to a network of more than 50 computers spread over a ten square kilometer site, all the time controlling an ever-changing accelerator complex. This paper will discuss the major elements of the strategy used and explain the reason for their choice. Microprocessors have become very popular in the field of industrial control and the SPS control system is going to integrate this trend with little difficulty. The paper will show that the SPS approach is ideally suited to the construction of a real-time control network making use only of microprocessor based units.

Keywords. Distributed control system; network; multi-programming; multi-processing.

INTRODUCTION

The SPS machine

An accelerator of the SPS type is essentially a ring vacuum tube in which protons are guided by magnets whilst being accelerated by radio-frequency field. The process is cyclic, protons being injected into the machine, accelerated up to the desired energy and finally ejected along beam lines and taken to experimental areas. The SPS machine is largely housed in a tunnel about 30 metres below the surface and there are more than 10 km of tunnel. To prevent the control apparatus from radiation damage the majority of it has been located in surface buildings spread along the machine over a site of more than 10 km square. The control requirements are concentrated in these surface buildings and involve more than 60,000 parameters.

A special characteristic of the SPS is that it is for both, a day to day production plant and an experimental device. In effect the behaviour of such a machine is never well known and it is under continuous development, new equipment being added, new control strategies being tried. Indeed this ever changing environment is reflected by the fact that the SPS machine which started operation in 1976 is currently being turned into a proton-antiproton collider and is scheduled to become the injector for the future electron- positron collider LEP.

The impact on the control system

The framework described above has two main impacts on the control system. Firstly, the control system for a machine of this size has to be more than just a collection of controls transferred from the various apparatus to a control room. It is necessary to find a means of analysing the vast amount of data available and presenting it to the control crew in a suitable form for easy assimilation, and to carry out settings and measurements. As, on the other hand, the distance between the control room and most of the surface buildings is considerably more than one kilometre, it is essential to concentrate and process some of the data at the place where they are produced. This implies that computers have to be integrated into a distributed control system.

Secondly, the very large number of control procedures required and the fact that the control strategy is constantly changing with the additional requirement of being "instantly" modifiable, imply the provision of appropriate programming tools to allow physicists and engineers themselves to implement the control procedures they need. It was early recognized that a compiled programming language was unsuitable for these requirements, but that an interpretive programming scheme offers the necessary facilities. In effect interpretive languages are easy to learn owing to their

*) Presented at the 3rd IFAC WORKSHOP
Beijing, China, 15-17 August 1981

interactive nature; they are extremely flexible, as they allow the operator to take action during the running of the program whose source code is instantly available for inspection and modification.

The consequences

All these considerations on the process to be controlled and its impact on the control system have produced a method by which the global control problem has been cut up into units of a practical size and solved separately. This in turn implies the construction of a real-time control network with its associated difficulties. The paper shows how to overcome these difficulties, the methods discussed being:

- the control procedures are split into suitable groups such that much activity occurs within the group and a restricted amount between the groups,
- the real-time problem has been solved by the use of special purpose hardware monitored by the computers but not requiring information flow through the network,
- the information flow between distributed groups is further reduced by use of local pre-processing and post-processing whenever possible,
- the construction of the network is such that there is no privileged position in it.

DISTRIBUTED PROGRAMMING

Task distribution

The tasks have been carefully distributed among the computers to minimize the relationship between the computers. There are parts of the SPS, such as the radio frequency or the beam extraction systems which are functionally grouped in a surface building and they can be controlled by a dedicated computer. There are others, such as the vacuum system or the beam instrumentation, which are spread all along the tunnel and which are more conveniently controlled by a computer housed in the surface building containing the associated equipment. Such a computer will deal with the things which are common to the area of the tunnel it supervises.

For the central control area the same distributed approach has been used, a number of computers performs specified duties: general purpose console, alarm analysis, library...

This structure presents considerable advantages. Each computer performs a well defined task, the system can be built up progressively and the operation or modification of one part does not act directly on other parts. Moreover the same type of computer can be used everywhere,

with obvious savings both on hardware and software. This is demonstrated by the fact that the control system which started in 1976 with 24 computers has been progressively upgraded to more than 50 computers without major modification of the system.

The real-time constraints

Making use of a network of computers, the question immediately appears how the real-time constraints are satisfied. The answer to this question is twofold. Firstly, having many central processor units close to the equipment guarantees a better response of the system to individual interrupts from equipment. Secondly, the machine real-time stimuli can be pre-planned and they can be handled through the following phases:

- a planning phase, determining what will be done at the critical time and when that time will occur,
- the time critical phase,
- a checking, or data collection phase which may be not necessary.

It is only the second phase which is speed critical and for which a network operation may be inapplicable. Therefore this phase must be executed by a hardware independent of the computer system, and this hardware must be under total computer control. A concrete example of this is the measurement of the beam position at a specified time in the cycle. The timing hardware of the computers involved in this measurement is set up from the computer, the timing hardware will then trigger the hardware of the beam position measurement; the results of the measurements can be finally fetched at leisure.

How to use the distributed system

There are basically two ways to use a large distributed system: the traditional way is to ensure the existence of a large central data-base. This enables the data handling to be decoupled from the data gathering and setting process. The data base carries an image of the actual and desired states of the process. The application programs used for operation act on the data base rather than on the actual system. A scanning or refresh mechanism is supplied to ensure that the data-base corresponds at all times to the state of the process, and that, conversely, the process corresponds as nearly as possible to the desired state. With this method one has three distinct functions in the network: the data acquisition and setting where computers mostly act as data concentrator and multiplexer, the data-base handling and the data crunching.

This method of moving all the data towards the data-base has some important weaknesses: there is a time-lag between a change occurring in the process and the appearance of this change in the data-base. Enquiries made during this time will refer to obsolete data, and may even result in inconsistencies between the status in separated locations. This defect can be reduced by increasing the frequency with which the scanning is done. In practice a system of this kind draws a compromise between freshness of the information and saturation of the data-links and the process controllers. This is a pity, since much of the information gathered in this way will never be used.

The other method which has been used for the SPS, is to allow the database to remain distributed, and only move information when it is required elsewhere. In this way, the data-links between computers are kept idle until they are needed for some process; the information, while instantly available, is kept where it is unless it is needed for some processing elsewhere. The fact that the data-bases are kept near the equipment to which they refer means that they are more likely to be correct, and that no network resources are normally taken up in refreshing them. This in turn means that the network traffic is kept to a low level, and that when information is needed, the full bandwidth is available for instant response.

This latter consideration has been one of the constant concerns all along the project implementation and most of the system design decisions have been made on these grounds.

THE NETWORK OPERATING SYSTEM

The control strategy defined above entails the definition of the following layers for the operating system contained in every computer:

- connection between computers
- process
- connection to equipment.

At the higher layer, the computers are connected through private high speed serial data-links. The network is organised into the simplest arrangement which offers a single route for messages flowing from source computer to destination computer. For the SPS a star configuration has been chosen, the transport algorithm being based on a store and forward mechanism for data-grams of a maximum fixed length. The network was initially a single star and has subsequently developed into a multi-star configuration.

At the lower layer, the equipment to be controlled is joined to the computer via

CAMAC as a primary interface, then through a specially developed serial highway multiplex system (MPX). The essential property of the MPX system is that it provides a one-to-one correspondence between a piece of equipment and its MPX interface module.

The intermediate layer deals with the execution of processes and consists of the combination of an interpreter and a multi-task real-time executive.

Process

For the SPS control system, a specially tailored interpreter called NODAL has been constructed. It is a re-entrant package which executes programs and single commands. A Nodal program is made up of computer independent elements consisting of a program line or a named element (variable, array, string...). The identifier of a program line is its line number with values ranging from 1.01 to 99.99; the number to the left of the dot denotes the group number to which the line belongs. The identifier of a variable, array, has its name determined by maximal 6 ASCII characters.

The interpreter runs under the supervision of a multi-tasking real-time executive. This executive organises the tasks to be run in a computer into classes according to their nature. There is a fixed number of classes for each computer and, at any time, there is only one task active on each class. The usual class arrangement includes: surveillance, interactive, remote, operator intervention.

The first class allows the running as a background activity of local surveillance tasks which survey the proper operation of the local equipment; this allows a further reduction in network traffic as only faults and exceptions which cannot be handled locally are sent through the network.

The interactive class provides, by means of a local terminal, the commissioning and the debugging of software and hardware in situ, a vital facility.

The synchronisation between tasks is performed by means of an independent timing system which is distributed to every computer. If tasks have to exchange data, this can be done through common data files or through the distributed data-base.

This combination of facilities has resulted in the construction of control procedure as a tree of real-time tasks, each individual task doing a limited amount of work in investigating some aspects of the process behaviour and reporting on it. The task can then schedule another task for immediate or conditional execution.

Network

Following the trend of minimizing the consumption of network resources, and of making use, whenever possible, of local processing capabilities, network facilities have been added to the programming language.

The Nodal interpreter has been equipped with statements allowing a piece of source code together with the essential parameters to be sent from a Nodal program for interpretation in a remote computer. The piece of Nodal code, made up of computer independent Nodal elements, is executed by the remote computer as a remote task on a dedicated class. The source of such a network transaction is a Nodal interpreter, the target is an identical interpreter; this means that the piece of code can be sent out into the blue to the remote computer: it will know what to do when it gets there.

Additional facilities available in Nodal enable the code obeyed remotely to remit its results to the sending task, as well as allowing the latter to re-synchronise itself on the arrival of the results. It is therefore possible to run a multicomputer program in one computer, parts of which are executed in various other locations, the whole being effectively a single multicomputer program. This single multicomputer program, resident as it is in one computer, has all the desirable characteristics of an interpretive program with local interaction. Run-time diagnostics and on-line editing are available for all parts of such a multicomputer program, even those parts which are run remotely.

The weak point of a remote-execute scheme of this kind is that the master program must contain explicit references to the computers involved, and the author must know which computer owns which specific piece of hardware to be addressed. Since, at the SPS, programming is done by engineers or technicians familiar with both the equipment and the computer layout, the responsibility for citing computer references explicitly has not given major difficulties. More than six years of experience has shown this to be adequate.

In addition to its simplicity, this scheme has two outstanding advantages for the saving of network resources. Firstly, as the program is sent towards the data, data reduction and packing can be performed and advantage can be taken of the block transfer mode of the network. Secondly, as the remote task mechanism exists in every computer, it allows a non-hierarchical arrangement where every computer can be a master for a given task and, at the same time, slave for another one; this is a very valuable feature for an ever extending network like the one at the SPS.

Equipment

Having established a method for writing control programs running in one place in a network and using data gathered "live" from a distributed data-base, we can now discuss how the individual local elements of this distributed data-base are organised.

The simplicity and flexibility of an interpreter make it possible to write application programs with a minimum of effort and risk, but a complete implementation also requires the interpretive language to be furnished with real-time facilities and hardware interface links. The devices used for this purpose in the existing "real-time" languages usually require the user to address the hardware in absolute terms from the application level. This results in programs that are difficult to write and impossible to read, and it is to avoid this problem that the Data-Module concept has been developed.

Few of the devices under control are unique. There are, for example, many pumps, many power supplies and many function generators. There is a Data-Module or interface routine for each type of equipment, which does sufficient house-keeping on the hardware items under its control to enable a very simple and user-friendly interface to be presented to the high-level program writer. The interface consists of a subroutine call with two parameters. The name of the subroutine is the generic name of the equipment involved, the first parameter being the sequence number of the item addressed, and the second parameter being the particular property of that item. For example:

MAGNET(6,CUR)

refers to the current flowing in the sixth magnet.

Properties may be of many kinds. The example given above refers to current, bit sensing status, setting status, hardware address, protection key, permissible limits and conversion factors may also be available. Thus the acquisition or setting of some properties may imply a hardware access, while others may refer only to a data item stored inside the data-module.

Data-module calls may be used in arithmetic expressions, as in the following example:

SET MAGNET(6,CUR)=101.6

has the effect of loading the digital-analog converter controlling the power supply for the sixth magnet (of that particular computer) with a value which will cause 101.6 Amperes to flow in the magnet. The data-module will perform the necessary tests for permissibility before allowing

the current to change, and will also apply any calibration, sequencing etc., which are necessary.

Similarly, the statement:

TYPE MAGNET(6,CUR)

will print the current in the same magnet, after causing the necessary analog digital conversions, calibrations, corrections etc., to take place.

Each computer contains a data-module for every kind of equipment under its control. Each data-module contains and manages that portion of the database appropriate to the equipment it interfaces. By having carefully designed a data-module, the application programmer can be saved from having to worry about the intimate details of the equipment. Rather he thinks of the equipment functionally, and leaves the data-module to look after the housekeeping.

An example of multi-computer program

These concepts are better explained in an example of a typical procedure. This procedure measures the profile of the extracted beam making use of a miniscanner which is moved step by step in the beam and whose charge is a direct information of the profile of the beam.

```
1.1 ASK "INITIAL POSITION=
    "IP "FINAL POSITION="FP
1.2 SET P=IP;EXECUTE(EXTR) 3.2 P
1.3 FOR P=IP+1,FP;DO 2
1.4 END

2.1 WAIT-CYCLE 6
2.2 EXECUTE(EXTR) 3 P
2.3 TYPE "POSITION="P-1 "CHARGE="
2.4 WAIT(EXTR);TYPE A

3.1 SET A=MINSCN(2,CHRG)
3.2 SET MINSCN(2,PSN)=P
3.3 REMIT A
```

Line 1.1 asks the operator for the range of the measurements to be done. Line 1.2 sets the miniscanner in the initial position by sending to the extraction computer line 3.2 together with the initial position. Line 1.3 loops over the range of measurement; group 2 is executed at each loop.

Group 2 starts by waiting for the extraction signal from the timing system. Then line 2.2 sends group 3 for execution in the extraction computer together with the next desired position for the miniscanner. When the remote task is launched, line 2.3 is executed immediately, typing at the terminal. Line 2.4 synchronises the execution of the program with the arrival of the result of the remote task, then types the measured charge.

Group 3 which is sent for execution in the extraction computer and which is executed in parallel with the main program, obtains the charge from the miniscanner in line 3.1, positions it for the next measurement and remits the charge to the main program.

TOWARDS A NETWORK OF MULTI-MICROPROCESSORS

For many reasons, microprocessors have become very attractive in the field of industrial control, and the SPS control system has already followed the trend by incorporating them at the edges of its system, mostly as "intelligent interfaces" for sophisticated equipment. Although this was very useful, it soon became necessary to organise this unavoidable transition, failing which the overall system might have degenerated to such an extent that the elegance and maintainability would have been lost. Initially some guidelines were set for hardware and software and for the manner of operation of the microprocessor subsystem in the context of the existing control system. In this paper we consider the subsequent moves.

The first obvious move is to make a multi-processor system, each element of which is itself a microprocessor. In other words, taking advantage of the autonomy of real-time tasks organised into classes, it is possible to replace each of the multi-tasking minicomputers by an assembly of microprocessors, each of which performs one single-stream type of duty:

- scheduling
- execution of interpretive real-time task
- communication with other assemblies
- communication with equipment.

A suitable arrangement is to use a crate into which the required number of microcomputers, each with its own memory, can be plugged and connected by a common bus which can be accessed for interprocessor communications within the assembly. The lowest level of the communication protocol is supplied by a bus allocation mechanism, the message level protocol could be of a packet type as in the SPS network, except that the connection between source and destination is direct and uses a parallel backplane. Such a crate will be referred to as a "multiprocessing assembly". The microprocessor with its memory will be referred to as a "general purpose processing unit".

Communication between multiprocessing assemblies can be carried out as in the original SPS system, but using the standard HDLC frame protocol for point to point connection, thus taking advantage of the HDLC drive chips available commercially. A plug-in is required in each multipro-

cessing assembly, having this hardware in addition to the processing unit which buffers and controls the message flow. Such a plug-in will be called a "data-link driver unit".

As the overall structure of the network is kept identical to the SPS one, each node of a star could consist of a crate holding data-link driver units. We shall call such a crate a "message handling assembly". Incoming datagrams will be buffered into the memory of the associated data-link driver unit, then transmitted through the bus to the memory of the corresponding data-link driver unit. In this mode of operation a truly autonomous handling of datagrams could take place, occupying only those units mentioned in the address of the datagram.

The CAMAC+MPX compound interface system of the SPS could also be replaced by similar crates which will be called "equipment control assemblies". An equipment control assembly contains the necessary I/O plug-ins to access the equipment and possibly processing units performing autonomous control on individual items of equipment driven by the crate, the software housed in such unit corresponding to the data-module. The connection between an equipment control assembly and its associated multiprocessing assembly can be performed by a "multiplex driver unit" operating in a multi-point protocol. To summarize the arguments, only three basic types of plug-ins are required:

- General Purpose Processing Unit GPPU
= microprocessor + memory
- Data-link driver unit = GPPU + point to point interface
- Multiplex driver unit = GPPU + multi-point interface

plus the family of I/O plug-ins to connect individual equipment.

The overall architecture can be traced back to that of the current SPS network, namely a connection of stars whose web can be freely adapted to match the layout of the process to be controlled, the only restriction being that there is a single path from any source multiprocessing assembly to any other destination multiprocessing assembly.

From the software side the advantages of a system of this kind are manifold, the most outstanding being that the operating system is made out of free-standing packages in dedicated plug-ins, thus making the programming safer and considerably simpler than in the case of modern mini-computers. The multi-tasking requirement no longer exists, despite the fact that true parallel processing can be carried out.

From the hardware side, the small number of distinct types of unit can only be considered an advantage for development, construction and maintenance. The advantages of mass production are particularly important when the control project is large, or when flexibility or expandability are vital.

CONCLUSION

The SPS machine is a typical example of the usefulness of a distributed approach for solving considerable control problems as it shows that the problem can be divided not only spatially among the computers but also organisationally among people. It should be emphasized that the system is a combination of a simple message exchange protocol, a conventional multi-tasking monitor and a straightforward interpreter. A small number of professional programmers have constructed a system which numerous engineers and technicians have been employing both for operating the machine and for writing their own multi-computer control procedures.

The computer field is a rapidly moving one and the lifetime of a large, expensive machine like the SPS may be greater by a factor of three than that of a computer. It is therefore increasingly important to design a control system in such a way that it can metamorphose on-line and the components can change to ones of new design without significant loss of system services.

The method outlined here is working at CERN SPS and seems to be applicable to other fields also.

REFERENCES

- Crowley-Milling, M.C. (1975). The design of the control system for the SPS. CERN 75-20.
- Crowley-Milling, M.C., G. Shering (1978). The NODAL system for the SPS. CERN 78-07.
- Altaber, J., V. Frammery, C. Gareyte, P. van der Stok (1979). Principles of the operating system for the SPS real-time control network. CERN/SPS/ACC/79-13.
- Altaber, J., F. Beck, M.C. Crowley-Milling, R. Rausch (1979). IEEE Transactions on Nuclear Science. (vol. NS-26, No. 3) June 1979.
- Altaber, J., F. Beck, R. Rausch (1980). 11th International Conference on High-Energy Accelerators. Proceedings ed. W.S. Newman. Birkhäuser, Verlag, Basel. pp. 440-443.