DATA TRANSFER IN ON-LINE SYSTEMS

V. Zacharov
University of London Computer Centre

Abstract

The problem of transfer of data in both directions between experimental equipment and process systems on the one hand, and hardware processors on the other, is an important one. This fundamental question is discussed in the context of contemporary practice, where the principal processing element is the minicomputer. Although several interface conventions will be considered, practice is dominated by the CAMAC system, and the main emphasis will be to review recent developments in that system, particularly in the area of distributed configurations. The impact of new microcircuit technology on the way in which data transfers are performed is only beginning. The present discussion will try to assess this impact and to identify the main changes that are expected to occur.

1. Introduction

In the overwhelming number of cases, the digital data generated by instruments, detectors and transducers in experimental science, in engineering and in process-control applications, is, in raw form, useless. It is usually of considerable volume, it is generated at rates much too high for direct human absorption and, inevitably, some of it will be contaminated or destroyed by some form of noise or spurious data. In short, such data is neither useful information for those human beings involved nor is it of the right form to be used as input to some kind of automatic control system.

Nowhere is this situation more evident than in experimental particle physics where, for example, many modern spectrometer experiments may generate data at rates in excess of 10^6 bits/sec over running periods of months, but where the final result can usually be summarized, if not actually by one number, at any rate by a very small number of graphs or histograms.

Thus, before data can be of any value in all these cases, it has to be processed in some way or other. Usually the processing will result in a reduction in volume of source data by several orders of magnitude, and this can only be achieved by an automatic computational process; but in other cases the problem may only be one or sorting or formatting, in which case some kind of computational process will again be required. Even if all that is required is temporary buffering or more permanent storage of data, there will still arise the question of some sort of data transfer and processing.

So we can see that, in the majority of real cases, the data generated by various detectors and other data sources has to be transferred into some other device or system, and this device must be capable of digital processing or computation. But this problem is further increased by two other realities: first, the data is not usually generated in the right place and so will have to be transferred in any case; second, it is usually generated on time scales that require the data to be removed very quickly indeed, either because the data source has to be restored to a form appropriate to record more data, or because some result (or feedback) is required from the data already generated. This second consideration is, in fact, one which imposes one of the most stringent demands on the data transfer system; the data transfer problem becomes indeed a "real-time" one.

Corresponding to the all-pervasive real-time data acquisition problem just described, there is also the problem of data output. Here also, even in the event that the data is required for visual inspection by some human being, the requirements are often very demanding. The human being not only may wish to see a presentation in graphical form, often dynamic and with some dimension (like colour) artificially introduced, but he will certainly require the information on a quite different time scale from that of input data generation. But, for process-control applications, the output data required for transducers and so forth will generally be of totally different form from the input data. And because output data is frequently required, both by humans and by control equipment, on time scales dictated by the need to ensure that the course of the experiment or process is a correct one, the output data-transfer requirement becomes like that for data input, a real-time one.

In summary, for so many problems associated with data generation and output, the first problem that arises is that of data transfer. And because of the nature of the data and the associated process, this problem is generally one of real-time data transfer. The purpose of these talks will be to examine this problem and to see what success there has been in finding relevant solutions. Since the problem is of such universality and pervasiveness, the question of finding conventional or standard solutions becomes of the greatest importance, and particular emphasis will be placed here on this aspect.

2. The System Problem

The first question that we need to examine is between what device or system component the data should be transferred. There are very many different kinds of data source and also a wide range of sinks or acceptors. Data can be processed by very many different processors or computers. So an important question is whether there are many different types of data-transfer problem or not.

Actually the number of different categories of receptors of data is not very great, so let us examine each of them in turn to see if there are any common features in solving the problems of data transfer to or from each category. As we shall see, we can reduce the complexity of the problem considerably.

2.1 Data Links

For the sake of completeness in our survey of data transfer we need to include data links, because one thing we can do with data is just to transfer it somewhere else. Of course, such links do not in themselves help us much except in removing data from one physical location to another, more convenient, one and sometimes in providing some short-term buffer storage. But, even if one could readily transfer data from detectors or other sources into data links, the problem is only translated to that of data transfer at the other end of the link.

The range of data links is enormous, and there is certainly no difficulty in transferring data along appropriate links at the highest rates of data generation encountered. They have been extensively described elsewhere and are only mentioned here to emphasize one point, namely that the use of links for data transfer introduces additional problems. It is not just that noise and digital errors are introduced by data links, but that their use inevitably leads to problems of increased complexity both in hardware and in logic. Even in the simplest case, the use of digital data links will introduce the necessity of an error-recovery strategy and a communications protocol,no matter how primitive these both may be. In more complex situations, there will also be the problem of commutation and multiplexing.

Thus the existence of data links does not really help us solve the fundamental problem of data transfer, namely, that we have to get the data processed. Indeed, data links introduce additional problems which further enhance the need for a data-processing capability in the system.

2.2 Storage

There are two categories of storage media that we need consider: short-term buffer storage and more permanent mass-storage. In all practical situations the former always exists, even if only in the form of cables. But nowadays it is rare indeed to find any system generating digital data without some form of active storage, and the main purpose will be for so-called "de-randomizing" and for obtaining from short bursts of data at high peak rates longer data blocks at more modest transfer rates.

So, like data links, buffer storage, however useful, does not really solve the basic requirement of data transfer which is to move data into or out of an "intelligent" processing device. And also, like data links, the use of buffer storage introduces additional requirements to process the data. Modern buffer storage is nearly always available in some conventional form, into which data can only be inserted provided it is properly formatted both physically and logically. And so the raw data from detectors and other data storage has often to be processed in some way even to be inserted into buffer storage. It is very uncommon indeed in modern systems for buffer storage to be unassociated with some form of processing element, and the most usual arrangement by far is for the buffer storage actually to be the memory of a minicomputer. Once again, the problem which remains to be solved is how to transfer data into intelligent devices.

In the case of mass storage the argument is different, since any computer system can have such storage attached, for example, magnetic tape units. So, if data can be transferred on-line, say to magnetic tape, then the tape can be subsequently transferred off-line onto a processing system. But the problem of data transfer onto magnetic tape, or for that matter onto any other mass storage device or medium, remains. Here we are faced with two problems: first, that of formatting the data to the form required by the storage medium; second, that of matching the data rates. For both these problems we require together both a buffer and a processing device, and by far the best way of solving these associated problems is to use a minicomputer. Once again we see that the problem of data transfer, even in the case of intermediate storage, reduces to that of transfer to a processing device, usually a minicomputer.

2.3 Large Computer Systems

If we set aside for the time being the pre-processing of experimental data, associated for example with event selection in elementary particle interactions, the overwhelming bulk of data processing tasks could be handled in principle at least by large computer systems. Such systems have adequate processing power, they have a wide range of peripheral devices, such as mass storage, and they are quite capable of the formatting and display required for visual presentation to human beings. So we should ask why cannot the data transfer problem be solved by moving the data into and out of large computer systems.

This was indeed one of the early approaches to on-line data acquisition and control in a number of large centres engaged in both high-energy and nuclear-structure physics, where experimental equipment was coupled to large (at that time) computer systems by means of special hardware directly coupled to an I/O channel. Later,special hardware was partly replaced by hard-wired couplers or data adapters provided by the manufacturers of the large computer mainframes, a typical example of which was the IBM 2701. It was quickly realised by many, however, that this approach was not the best,for a whole range of different reasons.

The operating systems of large computer systems are not designed to handle real-time problems of the kind encountered in experimental science. On the contrary, such operating systems are intended to utilize system resources (particularly central processors) very efficiently, or to give a rapid response to multi-access terminals operating at human speeds, or sometimes both. In addition the architecture of large systems was developed so that input/output was mainly along channels designed to match the requirements of conventional peripherals such as tape and disk drives. All these properties of large systems taken together made them well suited for transfers of large blocks of data in a highly scheduled and ordered manner, that is to say, kiloword blocks transferred at 1-10 millisecond intervals, but very badly matched to real-time requirements of data transfers in small blocks at microsecond time scales. Moreover, large computer systems are almost incapable of operating in a demand-handling regime; at the very best, the processing of external interrupts on a large system only results in a tremendous degradation of performance.

There are other aspects of large computer systems we should take into account, such as the fact that they are inefficiently used for the short-word integer data processing characteristic of data transfer tasks; but, in summary, they are very ill-suited to the environment we are discussing.

The way out of the problem of transferring data to and from large systems is to introduce an intermediate computer system between the data sink or source and the large computer, to make an impedance match, so to say. Such an intermediate computer system is termed a "front-end" computer, and invariably comprises one or more minicomputers. The functions of the front-end system can vary, but usually include buffering, multiplexing, formatting and sometimes even a certain amount of pre-processing, mainly involving integer arithmetic. The front-end processor will also handle all those computer tasks associated with communications and with servicing interrupts.

Thus, yet again, we see that the problem of data transfer between large computer systems and experimental or process-control equipment reduces to that of transferring the data to or from a front-end computer, that is to say, a minicomputer. In actual practice there may well be more than one minicomputer, either to separate certain functions or to provide a back-up capability for enhancing reliability, but this is not of importance for the present discussion. Also we need not be concerned here with the nature of the connection between any large computer and its front-end, since this is a problem that is faced very seldom, usually only once in the life of the large system, which is typically five years or even more.

2.4 Special Purpose Processors

In a growing number of cases, particularly in experiments in high-energy physics, there is an intermediate stage in the chain of data-transfer system, from data source to data acquisition buffer and processor. This intermediate stage is concerned with enhancing the quality of the data by pre-processing or filtering; the aim is to reduce the data as much as possible in quantity and in average transfer rate, and eliminating as much spurious data as possible. In practice the intermediate pre-processor is a special-purpose system, generally made (and indeed designed) just for the requirements of a particular data source or experiment.

A discussion of pre-processors is outside the scope of the present talks, and is extensively discussed by others [1], but the data-transfer problem is not greatly affected by the existence of such systems. Data transfer into pre-processors is a problem that in general has to be solved in a way that differs for each new system. It is very rare indeed for special-purpose processors to be capable of complete reduction of data, and so the problem still arises in any case of further data transfer from the pre-processors to the next component in the data-transfer chain, nearly always a minicomputer.

For the purposes of the present discussion, we shall regard special-purpose processors as being a part of an experimental system. In high-energy physics, for example, we shall think of pre-processors as part of the event triggering system, yielding a more refined "signature" for wanted events than would otherwise be possible. So our problem here is, once again, how to transfer data to a minicomputer.

In one particular respect, the discussion of pre-processors is different from all the other components considered so far, namely that pre-processing can represent one aspect of the introduction of "intelligence" into the data-transfer chain between source and main data-acquisition processor. This is something that we shall come to briefly when we consider the present status of modular systems and the question of introducing "distributed intelligence".

2.5 Microprocessors

The whole issue of microprocessors is extensively discussed elsewhere [2], but is one which is raised here in the context of data transfer to ask if the question of data transfer to minicomputers is any different from that of transfer to microprocessors. The question is a complex one, which we will mention again later, but at this stage we can say the following. Firstly, if the microprocessor is just one component of a system whose architecture serves the function of a minicomputer, then the problem of data transfer here is identical to that of a minicomputer. The fact that certain manufacturers (such as DEC) have chosen different I/O structures in the microprocessors they intend to replace their own minicomputers is an irrelevancy, since it does not change the qualitative nature of the data-transfer problem.

Secondly, if one or more microprocessors are components of a special-purpose system, such as a pre-processor, then the problem of data transfer into such systems is, as stated before, a special one, outside the present discussion. The problem of transfer from the special system to any subsequent minicomputer remains however.

But one aspect of microprocessors still remains, namely, the case where a complex microprocessor system may be all that is required for the complete reduction of data from some experiment or instrument. In that case, the problem of data transfer into a system synthesized of microcircuits is raised. Unfortunately, the disparity of different microcircuit components is so great, and the rate of evolution of these components so rapid, that it is impossible at this stage to make any general recommendations. There are not even any *de facto* industry I/O standards for microcircuits, far less any interconnection conventions, and we must wait rather longer for the emergence of general solutions.

3. Minicomputer Data Transfers

We have seen from the foregoing discussion that the most important problem introduced by the need for data transfer is that of getting data in and out of minicomputers, whether it be for the purposes of control or because of experimental requirements. This has been the situation in most experimental science (and certainly for high-energy physics) over the last ten years or so, and it is one which is likely to continue for a considerable number of years more.

Thus the first question to ask concerns the input-output (I/O) structure of ordinary minicomputers in widespread use, and to see to what extent this structure is appropriate for data transfer. Necessarily, the environment we are considering is a real-time one with on-line data transfer. To answer our question we need to examine the principal components of the I/O

structure, which are:

i) Program-controlled I/O
ii) Direct memory access
iii) Demand handling
iv) I/O commands and software

We shall only review these briefly, for a more comprehensive survey will be treated elsewhere[3], and in particular we shall be concerned here only with the main options most generally met in typical and popular minicomputers.

3.1 Program-controlled I/O

This is the facility whereby data may be transferred berween certain registers of a minicomputer processing unit (CPU) and an external device by means of execution of certain instructions in the minicomputer command repertoire. Nearly always the external device is not coupled directly to the registers, but couples by means of an intermediate unit, generally termed an interface unit or device controller (Fig.1). The registers are usually special ones used only for I/O, but sometimes they are general-purpose CPU accumulators. In certain cases the special registers are chosen to be selected (and fixed) memory locations.

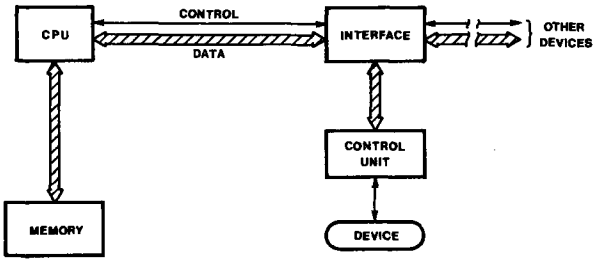

Fig.1 Program-controlled I/O configuration

Data transfers under program control occur when certain commands are encountered during a running program. Generally more than one instruction is necessary in order to transfer a single data word, several different commands being necessary usually for the purposes of device status checking, actual word transfer and waiting for device readiness. Even with very modern minicomputer architecture, program-controlled I/O is a technique that is generally slow, particularly for transfers of blocks of data words, since such transfers usually require even further instructions for counting and testing.

3.2 Direct Memory Access (DMA)

To overcome the limitations of program-controlled I/O for block transfers, many different techniques and configurations have been used, but by far the most usual is to transfer words to or from memory directly, with no program execution except possibly for setting up or termination of the transfer (Fig.2).

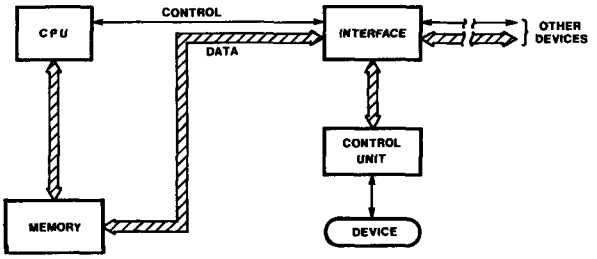

Fig.2 Direct Memory Access (DMA)

There are two essential aspects of most DMA systems. The first is that the operations that need to occur each time a word is transferred, such as counting the number of words, sequentially incrementing the memory address, and checking whether the required number of words in the block has already been transferred, are performed other than by CPU instructions and usually by hardware. The second is that transfers can occur at any time in synchronism with a memory cycle, and so have only to wait for the end of execution of a current CPU or memory reference instruction. This latter facility, known as "cycle stealing" or "data-break", implies that the CPU is generally only temporarily halted for a single memory cycle for each transfer of a word along DMA channel. Indeed, in certain minicomputers, the memory may be partitioned into segments with several access paths. In this case DMA transfers may actually occur into one memory segment, while the CPU can continue execution without interruption if it only requires access to other segments.

Because of DMA hardware the data block transfer speed can occur at a rate limited only by the memory cycle time, typically nowadays 300-500 nanoseconds per word. And because of the data-break mode, the time necessary to initiate block transfer can be very short indeed. In DMA transfers, data words in a block are stored in contiguous memory locations, so the steps necessary to initiate such transfers are to set up in appropriate registers a word count (data block length) and starting address in memory of data-block storage. There will also generally be the need to signal to the supervisor (hardware or software) that a transfer is about to commence (Fig.3).

Clearly, at the end of any block transfer, there will usually need to be an appropriate termination sequence of action, in order to restore the relevant minicomputer states back to those which obtained before block transfer. In many cases, this is done by generating an interrupt to the CPU with an appropriate transfer to the supervisory program to signal that DMA has been completed.

It should be emphasized that Fig.3 represents only one of many different variations of DMA procedure. For example, when a DMA sequence is proceeding, but no data is available, instead of initiation of a recovery sequence as in Fig.3, there might simply be a loop to wait for more data when

ready. It is important to note, however, that in nearly all DMA variants there is no effect on the status of any running program; the current address (CA) referred to in Fig.3 is the address of memory and has nothing to do with any program counter or program address register.

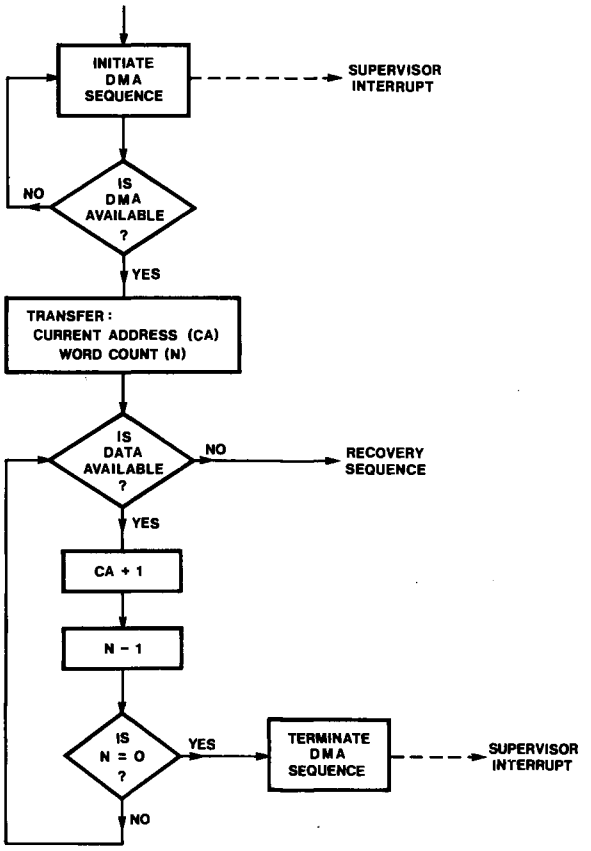

Fig.3 DMA flow-chart

3.3 Universal Bus

The path between memory and CPU in a minicomputer is generally termed a Bus, and it comprises data, address and certain control lines. On the other hand, both for program-controlled I/O and DMA, the intercoupling paths are not so simple; they are called channels. In this case, there is a fairly well-defined difference between the two in the way in which data is transferred. For the memory bus, transfers are extremely straightforward, being achieved in a single cycle simply by the presentation of the relevant memory address and a definition of the mode (read, write, data width, etc.). Data transfers along I/O channels require, however, a definite sequence of distinct actions or steps (Fig.3 shows one example of such a sequence); indeed, what is required here is a "protocol", albeit a rather elementary one.

In certain minicomputer architecture the memory bus and I/O channels are combined so that all elements or modules of the whole system communicate and transfer data along a single common highway or bus (Fig.4). The best known of such examples is the DEC Unibus.

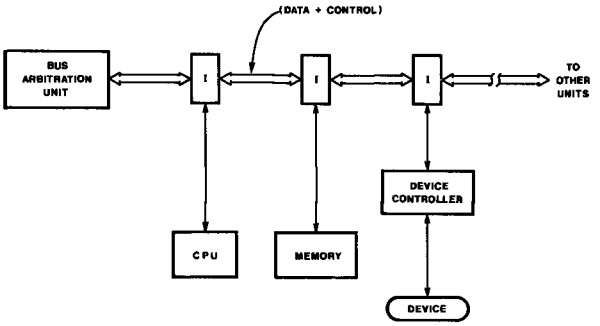

Fig.4 Universal minicomputer bus

Such common buses have several advantages over the other forms of architecture we have considered. For example, the addresses of memory locations, CPU registers and also registers in peripheral and external devices can be regarded as homogeneous, filling together the minicomputer address space. Here, contents of all locations in the address space can then be used as operands in any of the CPU instructions, including peripheral register contents. In common highways, the distinction between program control and DMA vanishes, since both become merely options of the general transfers along the bus. Also it becomes possible to transfer data between two devices directly without any intermediate step involving the CPU. There is, however, one important disadvantage generally, namely, that only one single transfer can occur at any given cycle along a common bus.

Since all data, address and control lines are shared in common by all units coupled to the common bus, it is clear there cannot be more than one paid of addresses present or more than a single data word at any one time. This problem has been tackled in many different ways, for example, by only allowing a single unit coupled to the bus (perhaps the CPU) to supervise all transfers; in other words, there will only be a single bus "master". More commonly however, the problem is resolved by having a special unit (or special software) to select which pair of modules coupled to the bus is allowed to be in communication at any given instant. Such a unit is called a bus-arbitration unit as shown in Fig.4, but it must be remembered that the problem is a complex one, which requires for its resolution a knowledge of the states of all devices and units coupled to the bus as well as the ability to respond to every control condition that might arise. As we shall see, this problem is related to that of demand handling, particularly in an environment of interrupts with differing priorities, and it cannot generally be resolved by hardware alone.

So we see that, whatever the disadvantage, at least the common bus has the important advantage of more or less common data-transfer protocol for all external (and, for that matter, internal) device controllers coupled, irrespective of their speed or application. And in addition, as seen in Fig.4, the interface (I) for coupling to the bus is the same for all units. However, before we can say whether or not the common bus of a minicomputer meets all

the requirements imposed by data transfers, we must examine the remaining aspects of demand handling and of software.

3.4 Demand Handling

Under program control and possibly even for DMA it would be possible to accomplish data transfer under the complete supervision of the CPU. However, this would certainly necessitate the frequent polling of external devices to see whether they required any attention, but also to discover when any particular transfer was complete. At the very best, polling can be a very inefficient process, but in a real-time environment it is generally impossible for polling to satisfy the data-transfer requirements. In practice it is essential to respond very rapidly indeed to any demands of external devices, and this is done by introducing external interrupts.

The form of interrupt scheme implemented in different minicomputer architecture varies enormously. In certain systems there is only a single external interrupt line which signals to the CPU that some external demand exists, and it is then up to the supervisor to identify which of the external devices requires servicing and what action needs to be initiated. In other schemes there is both an external interrupt line and a separate set of lines used to signal to the CPU (or the supervisor program) that some action related to DMA transfers is required. Such "service" interrupts can be used, for example, to meet the supervisor requirements of Fig.3, and they can result in very rapid responses to external conditions.

Whatever the precise form of demand handling, interrupt schemes usually have an associated means of introducing a priority in the response. Clearly this is of great importance in real-time applications, since a higher priority real-time process may need to be initiated even while a lower priority demand is being serviced as a result of some previous interrupt.

Here again, there have been many different ways in which a priority interrupt system has been implemented. In some minicomputers it has been done simply by physical position of a device as coupled along a data channel; the further away from the processor, the lower the priority. In other minicomputers, the priority is introduced by having several different interrupt lines each with a different priority. Sometimes the priority is determined by a "status" code on a set of interrupt lines and, yet in other systems, sometimes different priorities may be determined by some combination of all the techniques mentioned.

In general, in any interrupt involving the CPU, there will have to be a break in the running program to service the interrupt if it is of higher priority. In such cases it is normal to save the contents of all the relevant registers in the CPU and elsewhere and then branch to the appropriate interrupt service routine, either a general one which then has to identify the interrupt source or one special to the already identified interrupt source.

Clearly, with a common bus architecture, where every data transfer as well as transfers of control information have to contend for access along the common path, the question of demand handling assumes the greatest importance. In particular, the time necessary for handling any demand involving transfer of bus mastership must be minimized. One way in which this has been done is to introduce so-called vectored interrupts, so as to avoid the time-consuming task of identifying the source of an interrupt. In vectored interrupt systems, the interrupt source itself provides (either directly or indirectly) the address in memory of the location of a routine specific to service that particular source. Usually the interrupt "vector" which contains the service program address also contains the "status" of the interrupting device. In certain cases of vectored interrupts using a common bus system, such status information can be useful in establishing that a transfer may be initiated between two coupled devices without interrupting the operation of a third.

We have seen that the priority interrupt structure of a minicomputer can be quite complicated and that the range of different implementations is very great indeed. Modern techniques using register stacks and micro-processing, while helping to reduce the interrupt handling time, also serve further to increase the complexity. There is a constant introduction of new features in minicomputer systems, designed to speed up interrupt handling, but many of them introduce dangerous compromises. One example of this is automatic saving of CPU register contents by hardware, leading to an actual loss of time in certain cases when the CPU is not required to service the interrupt.

Certainly there is very little homogeneity in the interrupt handling schemes offered by different minicomputer manufacturers, and there is very little stability even from one single manufacturer. In the case of micro-computers the situation is particularly bad, and we have seen certain manufacturers actually introducing new common bus structures on the micro-computer systems they offer to replace their own minicomputers. But, in any case, before we can complete our conception of the usefulness of minicomputer I/O and demand-handling systems for data transfers, we also have to consider the associated software.

3.5 I/O Software

It is not the purpose here to review all the programming techniques developed for handling input and output to minicomputers, but rather to illustrate the scope of the problem. The first point to make is that the repertoire of I/O functions in any given modern minicomputer is usually quite small, but it differs greatly from one manufacturer to another. Moreover, even within one single minicomputer, there may be different I/O commands for the different types of I/O; for example, there could be one group of commands for DMA, another group for program-controlled I/O and so forth.

But of course, if the range of commands necessary for I/O differs from one minicomputer type to another, the spectrum of techniques used for demand handling is even broader since, as we have seen, these techniques depend not only upon the I/O commands available, but also the hardware associated with the data channels or highway as well as the nature of system supervisor.

In recent times we have seen a certain simplification of I/O software due to the growing use of common bus structures. Thus, not only is the command repertoire in these cases relatively simple, because I/O data transfers are just special cases of general register-register transfers, and because arithmetic and logical operations on both

full words and bytes can apply equally to data as to CPU operations, but the vectored interrupt scheme allows the introduction of much more modularity into the handling of I/O demands. Nevertheless, in spite of all the recent developments in minicomputer architecture, and in spite of the long experience now in dealing with real-time problems, it is fair to say that the problem of software driver for any new on-line device is one that generally has to be solved anew.

3.6 Is the Minicomputer Suitable for Data Transfers ?

Having reviewed briefly the I/O structures of modern minicomputers, we come back to our basic question as to their suitability for data transfers in a real-time environment. Well, the first thing to say, in all fairness, is that one probably could choose one range of minicomputers, and it would certainly be quite adequate at the technical level to solve most I/O problems. Indeed, this is exactly what has happened in an enormous number of cases throughout the world. Unfortunately, the solution to our problem is not quite as simple as just defining a certain range of minicomputers as a standard.

The first point is that different minicomputers have different properties, and some may be more suitable for certain tasks than others, even with the same external devices coupled on-line. The second point is that coupling of an external device to a minicomputer may depend upon the system configuration and upon the software or operating system available, so that even the same device coupler or software driver may not function correctly when moved from one minicomputer to another, identical in components but different in configuration. In yet other cases there is no separate device driver or module (either hardware or software) to transport from one system to another, because they are embedded in a more complex unit concerned with coupling many devices together. This question of lack of either "upward" or "downward" compatibility, as it is termed, is an enormous and frequently encountered problem, where all the work necessary (sometimes many man-years) to couple some device to a minicomputer has been performed, but the result is not in a form that can be transported, even between minicomputer systems of the same manufacturer.

So the problem reduces to that of compatibility and modularity, which is the only way to minimize the enormous task of data transfer. Manufacturers themselves have realized this by trying to offer "standard options" for I/O coupling, for example, by providing hardware modules with well-known electrical conventions such as TTL, and also by providing complete packages, both hardware and software, for coupling certain peripherals. Unfortunately this gets us nowhere, not only because we would be constrained to one manufacturer of minicomputer, but also to that range of external devices which that manufacturer felt to be of interest. In reality we need to be concerned with many different types of minicomputer and, in any case, with a vast and ever-growing number of external devices.

The oft-quoted example that, if there are N different minicomputer types and M different kinds of peripheral, then there need to be N x M different coupling adaptors or interfaces (to say nothing of software) is actually a considerable underestimate, since even the packaging or power requirements or some other factor may prevent the transportability of a given implementation.

The conclusion here is that to rely upon the I/O system of a single manufacturer would not effectively solve the data transfer problem, because it would be too restrictive; and to use many different I/O systems directly would be too expensive in terms both of costs and of manpower. The only alternative is to define some other widely accepted intermediate standard for coupling external devices, and we shall now review what has been done so far to introduce such a standard.

4. Standard Options for I/O Systems

There has been an enormous number of different attempts to introduce some kind of standard for I/O, but they fall mainly into one of two classes.

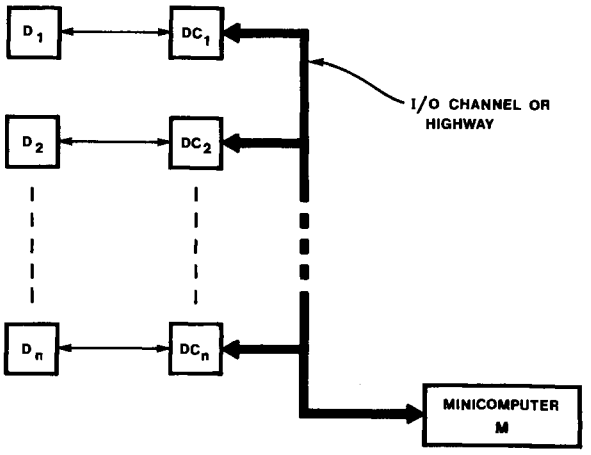

Fig.5 Normal method of device interfacing

The normal way of coupling devices is shown in Fig.5, where it can be seen that there are n device couplers for n devices on a single minicomputer. They all need to be developed separately for each new device and for each new minicomputer, and there may even have to be two different controllers or couplers for program control and for DMA. The first type of standard I/O solution is shown in Fig.6 where an intermediate standard interface is introduced, or rather a set of standard I/O ports. In this way, although it is still necessary to have n different device couplers (I), there is only one type of system coupler (IP) between the standard interface and minicomputer I/O highway. Thus the device couplers do not change in moving from one minicomputer to another, and the system coupler unit need only be developed once for each minicomputer (although there still have to be n of them for n devices).

The second type of option to implement a standard is not to have a standard set of ports, but rather to have a new standard I/O highway and to translate to this highway from the minicomputer data channel or I/O Bus by means of a single system converter or adapter.

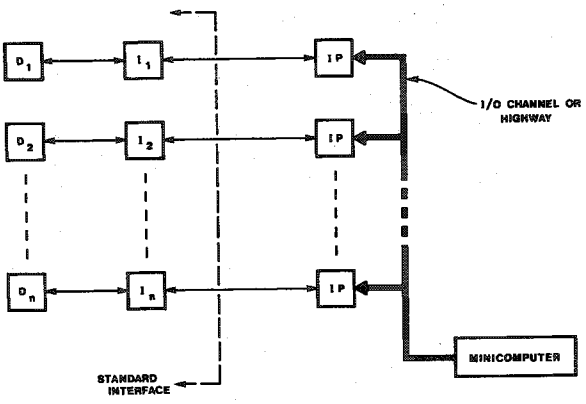

Fig.6 The standard interface

This solution is shown in Fig.7, where (IM) are interface modules which need to be implemented once only in the life of the device (D), and the system converter needs to be developed only once for each minicomputer. As we shall see, there can be several different configurations developed from the basic idea shown in Fig.7.

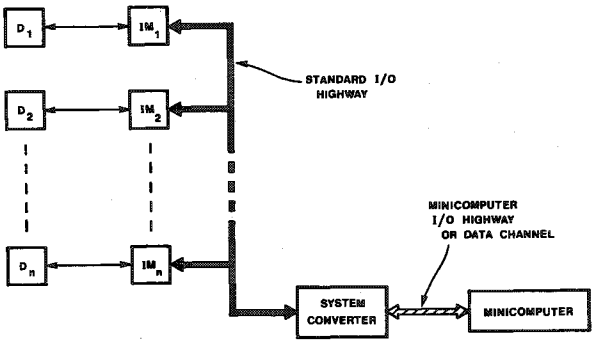

Fig.7 The standard highway configuration

Let us now look at some actual examples of standard solutions.

4.1 British Standard Interface

One of the best known implementations of the structure shown in Fig.6 is the BS 4421 system [4], originally devised in the UK by the National Physical Laboratory but later elaborated and defined as a British Standard in 1969. This system is based on the notion of data sources and acceptors which couple to each other across a "standard" interface, which defines the following:

i) Physical format
ii) Electrical signal specifications
iii) Logic levels and line allocations
iv) Data transfer mode and control

As can be seen from Fig.6, there will need to be $2n$ real adapter units to match n real (but different) devices to a given minicomputer, only n of them will be identical. At each of the unions across the interface, at ports, there is complete standardization of the data transfer at the level of an 8-bit byte.

BS 4421 defines a physical connector with 18 or 34 contacts depending upon whether single or twisted-pair lines are to be used. The lines are allocated for data, error detection and control, the data being transferred in 8-bit, byte-serial mode with a single parity bit. The signal levels are defined in a way appropriate to discrete components (such as transistors) and are different for data and for control.

There are seven control lines, with two used to signal the ready states of source or acceptor, one to signal if the byte parity in transmission is valid, and another one to define whether or not parity is being used. The other three control lines are used to control data flow, two for "hand shaking" and one to signal the last byte of a sequence in block transmission. The way in which hand shaking is achieved using the two "Acceptor Control" and "Source Control" lines is shown in Fig.8.

There is very little to say about BS 4421 except that it has not come into widespread use and is unlikely to do so. First of all, the data width and signal specifications are wholly inappropriate to modern technology and to most modern requirements. Secondly, the hand shaking mode together with the narrow data path make high speed transfer of long blocks extremely difficult to achieve. Finally, BS 4421 says nothing about demand handling, multi-source addressing or priority arbitration. In summary, it is no more than was originally intended, namely, an interface standard - and not a highway or data-bus convention. Even for data acquisition BS 4421 is not particularly useful, and it is really best suited to certain communications applications.

4.2 The MEDIA System

Quite different in concept from BS 4421 is MEDIA [5], an acronym for Modular Electronics Digital Instrumentation Assemblies, which is a system marketed by GEC Ltd under licence from ICI. It is of comparatively recent availability, having been released commercially only in 1975. The system is fairly simple in conception but rather restricted in implementation. The design of MEDIA owes much to the original intention, which was to implement a more or less self-contained system for industrial process control.

The MEDIA system adopts a wide range of different conventions, of which the principal ones are as follows:

i) Physical formats for modules, crates interconnection plugs
ii) Two types of highway, and associated intercommunication scheme
iii) Internal logic levels, addressing schemes and control sequences

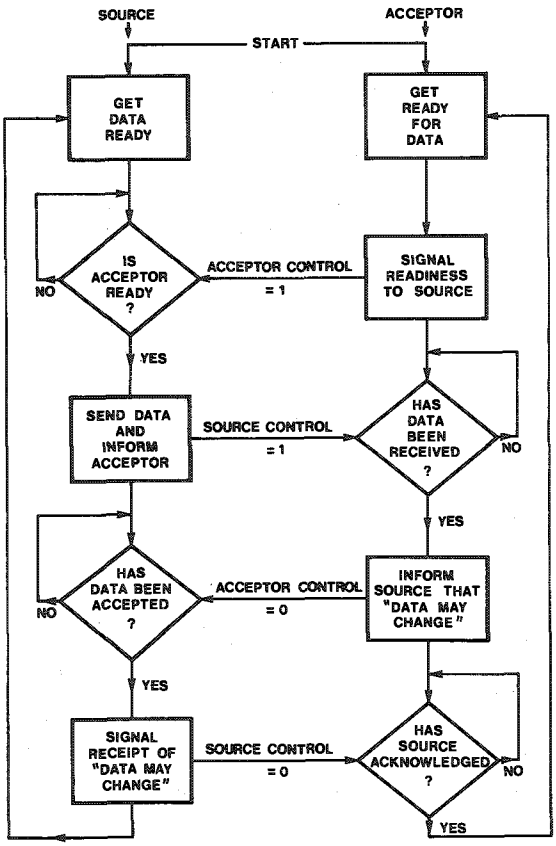

Fig. 8 "Hand shaking"in BS 4421 data transfer

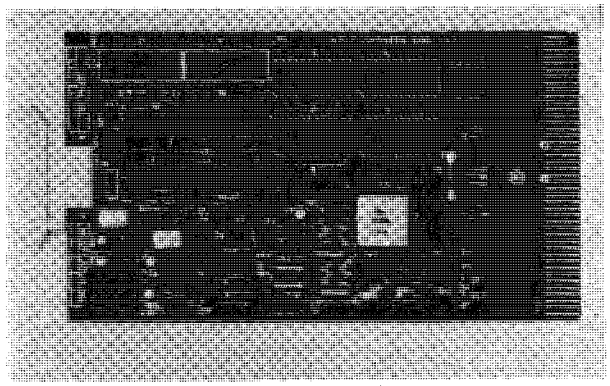

Fig. 10 Typical MEDIA module

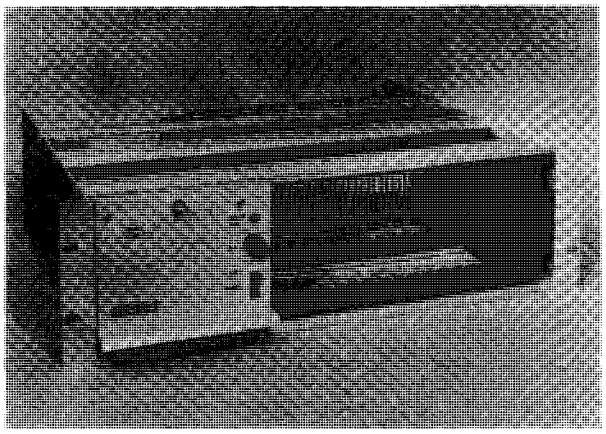

Fig. 11 MEDIA crate with power supply

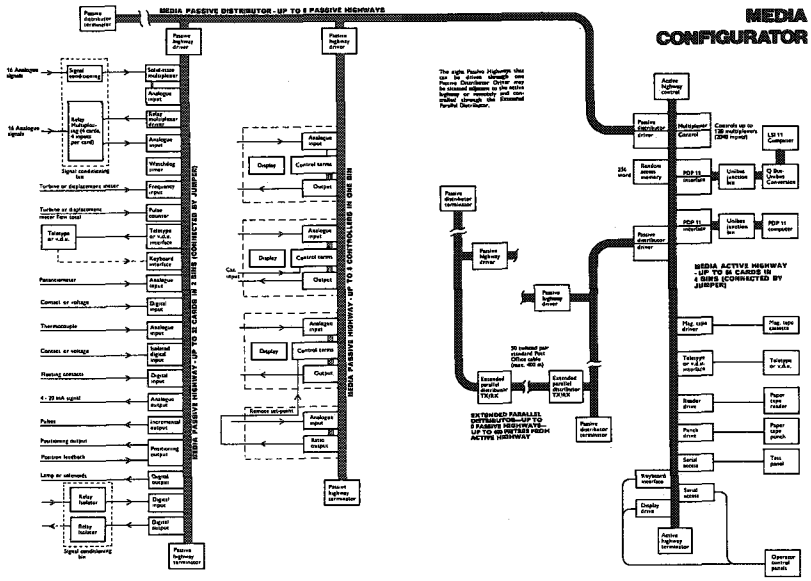

Fig. 9 MEDIA System

All these conventions are not stipulated rigorously in the manner of an international standard, but appear rather to be defined by the actual implementations of different units and system components by the single manufacturer involved.

Fig.9 shows a representative MEDIA configuration. It can be seen that two highways are involved, termed passive and active respectively, and that the two are intercoupled. The number of lines, the logic levels, the addressing structure and control are implemented separately for both highways, and both highways are essential to synthesize any system other than a very rudimentary one.

MEDIA modules, a typical one of which is shown in Fig.10, fall into one of two categories and plug into a bin or crate (Fig.11). Passive modules, with only a single address, can only send or receive a single 16-bit word on receipt of an appropriate command, and no passive module can generate a command explicitly, although a data word can clearly be interpreted eventually as an indirect address or implicit command. Active modules, which can only plug into an active bin, can generate commands, but have to contend for access to the active highway. Resolution of contention is by means of a hard-wired priority unit and by a rather inflexible time allocation scheme. The introduction of program control and processor capability is by modules on the active highway which either contain "intelligence" or which interface to a remote minicomputer.

In assessing MEDIA as a contender for standard I/O Highway in place of the I/O Bus of some minicomputer, it must be said that the system certainly offers something not available from most minicomputer manufacturers, namely, the means to couple signal sources and sinks (both digital and analogue) without bothering about many of the details. Also MEDIA is certainly modular and also not slower than most minicomputer I/O data channels. But that is about all we can say on the positive side. It is not that MEDIA is not useful, for manifestly it can help solve certain process-control problems very effectively. However, MEDIA is not a standard and, in its present form, unlikely ever to become one.

The plain facts are that there is no complete and explicit specification of the MEDIA Highways, nor of the associated protocols, and there is no definition of such things as the arbitration scheme on the active highway. These all exist of course in one particular implementation by a particular firm, and no doubt work exceedingly well, but there is nothing to tell an independent designer what rules he should follow if he wishes to develop either modules or interfaces. The question of demand handling is entirely unspecified in MEDIA except at the active highway level, to which the overwhelming majority of applications modules can only couple. With the exception of certain special-purpose process controllers, there appears to be only one MEDIA interface to a minicomputer, namely, a set of modules to enable a PDP-11 to be coupled.

Thus we must disregard MEDIA any further as a candidate for standard I/O system. Indeed, even if someone wished to use MEDIA and to construct alternative modules or controllers, he would presumably first have to obtain a licence from the two firms concerned !

We have to look further, to see if there is a more generally specified I/O standard. Such a standard is the IEEE 488 specification.

4.3 The IEEE 488 Bus

This specification [6], published in mid-1975, is the outcome of work by the Hewlett-Packard Company to develop an interface standard for laboratory instruments. The result of the work was an implementation (called the HP-IB), which is now a standard accepted not only by the IEEE but also by ANSI and the IEC.

The IEEE 488 Standard defines completely a data bus and, in electrical, mechanical and logical aspects, any port on the bus. Not only are these specifications complete and unambiguous, unlike the MEDIA system, but also they are defined in a way totally independent of any device attached to a port and, particularly, independent of any processor or minicomputer. Considerable attention is given to the signal specifications, and the highway protocol is also defined.

Fig.12 shows the bus structure and different categories of devices coupled to ports on the IEEE 488 Standard. There are 16 lines in all in the highway, of which eight are for data, transferred in bit-parallel, byte-serial mode. Three lines are used to regulate the transfer of each data byte in a "hand shake" manner, while the remaining five lines are for control purposes.

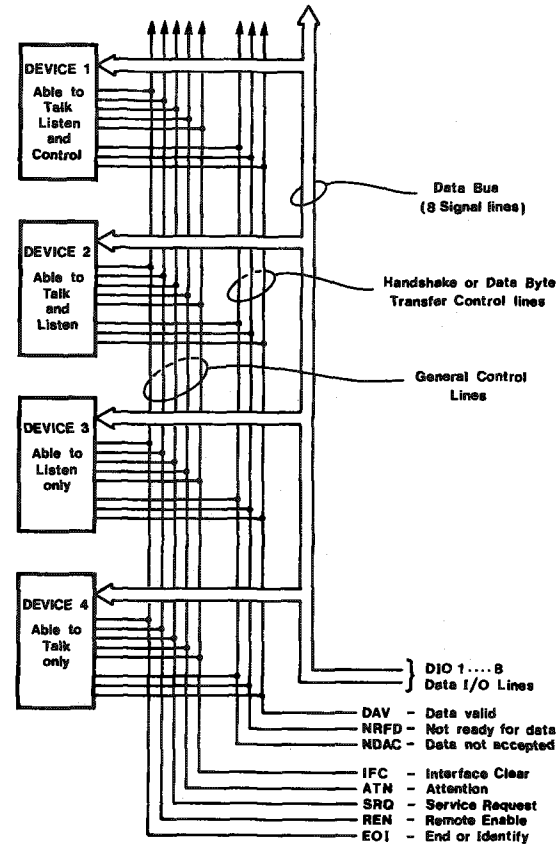

Fig.12 IEEE 488 Bus structure

The hand shake principle here is similar to that used in the BS Interface, only three lines are used, one (DAV) by the sender and two (NRFD and NDAC) by the acceptor. The three lines are used to signal as follows:

DAV	-	Data valid
NRFD	-	Not Ready for Data
NDAC	-	Not Data Accepted

Fig.13 shows a flow diagram for a typical hand shake procedure, where it should be noted that there can be more than one Acceptor but only a single Sender (or "Talker", as it is termed in the vernacular of IEEE 488). With more than one Acceptor, since the data transfer is asynchronous, the cycle time is governed by the speed of the slowest device.

The other control lines serve different functions, but most important is the ability to define any given coupled device as one of three types: a Talker, which sends data over the bus, a Listener, which receives data, or a Controller, which directs flow of data and defines which devices may "talk" and which should "listen" at any given time. To do this, there is an addressing scheme which uses seven of the eight data lines, together with an appropriate signal on one of the control lines. The eight data lines are bi-directional.

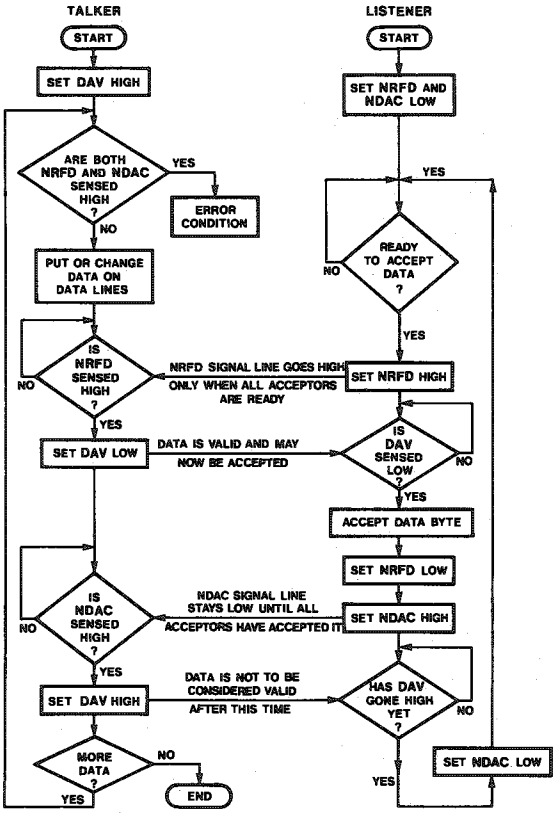

Fig.13 Hand shake procedure in IEEE 488

The actual bus is a multi-core cable to which are coupled defined connectors (Fig.14) allowing both chain and radial configurations. Signals along the cable are also defined with TTL logic levels. Although the address space allows 31 addresses (or more if two successive bytes are used), in practice only 15 devices are recommended because of electrical limitations.

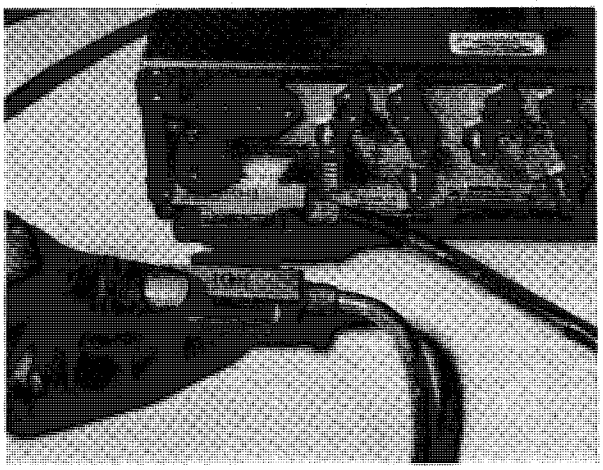

Fig.14 IEEE Bus connector

Any device can be coupled to a port of the Bus, provided it has a mating plug and can respond to the relevant signals. The address of the device is usually determined by a set of switches manually, but there are no other restrictions on the physical format of device. There is the facility for any device, including a Listener, to generate what is called a "Service Request", that is to say, a demand, and a Controller has functions which respond to such requests. Moreover, there can be transfers of "status byte" between devices.

Controllers can be programmable or otherwise, and there can be more than one, although clearly only one controller may be active at any one time. There is no explicitly defined arbitration scheme for passing control from one bus master to another, although certain primitive commands are available which would serve the purpose. The cycle time of the Bus can be as high as 1 Megabyte/sec, so manifestly there is no difficulty in using a minicomputer as a bus controller.

How can we assess the IEEE 488 Bus as a solution to our problem of standard for data transfer to minicomputers ? Well, there can be no doubt that it certainly serves the purpose for which it was intended, namely, to inter-couple laboratory instruments and to control their data flow by means of some"intelligent" controller. Unfortunately, that is about as far as the standard does go, although it is fair to say that a number of firms in addition to Hewlett-Packard are beginning to make their instruments with an IEEE 488 port coupler, so that they may either be used manually or on-line.

So far as experimental data acquisition is concerned, or for that matter industrial process control on any large scale, the IEEE 488 Bus is not very appropriate for a number of different reasons.

First of all, the number of devices that can be coupled is very small and, in any case, the address space is also small and without any sub-address or multi-register option within a given device. Secondly, the data width is too small; 16 bits would have been much better. The demand handling facilities are rather weak and block transfer is more or less overlooked. Finally, although the system is, in a certain loose sense, modular, the modules are totally undefined.

The merits of the IEEE 488 Bus are its low cost and device independence. For the purpose of a general data transfer standard, alternative to a minicomputer I/O Highway, these qualities are unfortunately inadequate. We still need a standard which has a wider data path, which can handle block transfers with ease, and which has defined modules. There appears, so far, to be only one candidate to fulfil this need - CAMAC.

4.4 CAMAC

CAMAC started off as a convention for modular electronics agreed by ESONE, a committee of European national laboratories mainly concerned with nuclear and elementary-particle physics and with nuclear energy. The genesis of this convention was in about 1968, but many of the features evolved from the earlier NIMS standard and from the Harwell 7000 system developed by the AERE. By 1969 this convention was published to define the mechanical, electrical and logical characteristics of a crate and data highway, and this was later extended to becomes a full European standard in about 1972. At this time also it was adopted as a US standard of the AEC, but it has since become the standard IEEE 483-1975.

Up to the present time CAMAC has continued to have additional features defined, some of which have also become accepted as IEEE and IEC standards, but it is true to say that the original definition of crate with dataway has undergone little change. The present status of CAMAC is that a very wide range of different features have been formally defined in a manufacturer and indeed device independent way. There is a vast body of literature devoted to the standard itself and to its applications.

It is not the purpose here to review all the aspects of CAMAC, for which purpose readers are referred to an excellent bibliography [7], but instead to view it in the context of our present discussion of data transfer. Therefore, let us summarize the principal attributes of CAMAC, which is concerned with the coupling of devices to an "intelligent" controller.

All devices in CAMAC are coupled by means of modules of standard format. These modules, which may be the devices themselves in certain cases, can occupy any one or more of 24 stations in a standard crate, at the rear of which is an interconnection highway called a Dataway. The Dataway is an 86-line highway with two 24-bit data paths, and each crate has a Controller, coupled to the two end stations 24 and 25 or a crate. The CAMAC system defines two such Crate Controllers (Type A and Type L), but there may be others as well. Thus, for example, the CAMAC system may stop there for certain applications, since a Crate Controller may be "intelligent". Fig.15 shows a Crate and typical module. Along the Dataway the addressing and command structures are completely defined, and there is also a comprehensive range of features to allow demand handling. Important aspects of the Dataway are that each module may have many sub-addresses, and that the status of each may readily be determined.

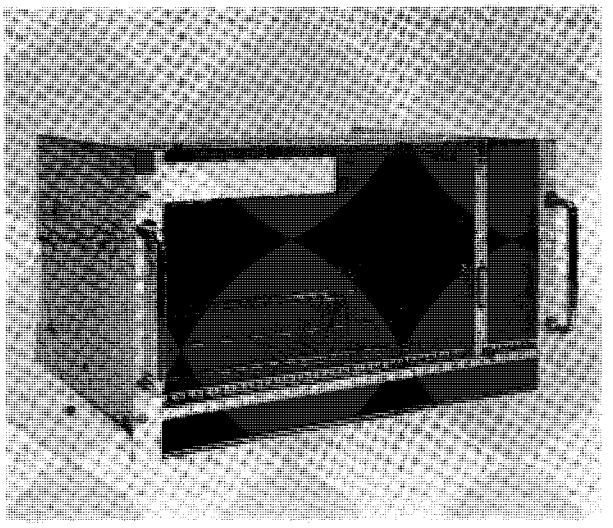

Fig.15 CAMAC Crate with single-width module

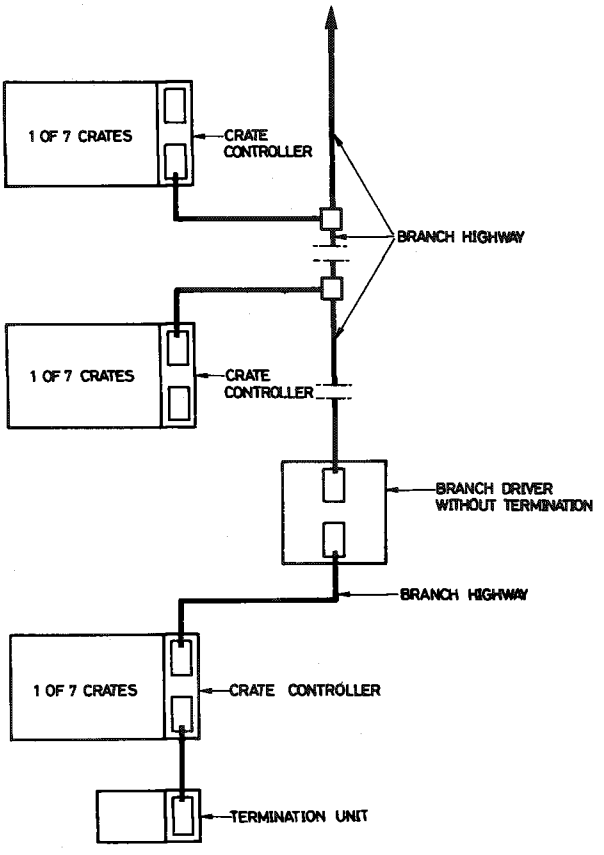

Fig.16 CAMAC Branch

Crates may be used singly, or they may be configured and intercoupled in a number of ways. The most common interconnection is the chain configuration shown in Fig.16, in which crates are coupled by means of a parallel branch or highway. One such highway is defined within the CAMAC standard, which allows up to seven crates to be coupled by means of the Type A controller. In the CAMAC Branch, the Branch Highway is completely defined, as is the port to which there must be connected some intelligent controller or branch driver. Another interconnection configuration is by means of a serial highway (Fig.17), and the CAMAC standard defines such a highway, operating in either bit-serial or byte-serial mode, with the corresponding serial crate controllers (in place of Type A). Clearly, there may be other methods of crate interconnection also, and these include data-link modules for CAMAC - CAMAC coupling, but only the Branch and Serial Highways are defined standards.

Thus CAMAC defines the electrical, mechanical, signal and logical specifications for all the principal interfaces between external equipment and an intelligent controller, and even defines standards for external analogue signals. Detailed information is given in the specifications about such aspects as timing and multi-address operation.

Particular attention has been paid,in formulating the CAMAC standard, to the question of block transfer of data words. Thus already the command repertoire of CAMAC and the Dataway organization make block transfers of any general kind readily facilitated. However, in a separate specification, CAMAC does illustrate several conventional methods of block transfer which make use of one of the dataway status lines, the Q-line. In this way, conventions are defined both for sequential data transfers from a single sub-address and for transfers from contiguous sub-addresses.

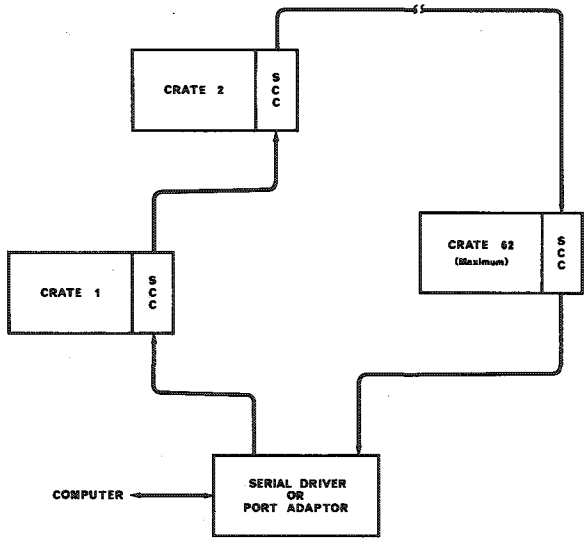

Fig.17 CAMAC Serial Highway

Unlike all the other conventions we have considered, CAMAC has evolved to a higher level in an extremely important respect, namely, that there exists such a thing as "CAMAC" software [8]. As a result of the very extensive installation of CAMAC hardware systems, and the ease of transportability of modules and indeed of complete sub-systems, there has been a very strong motivation to develop transportable software. This has mainly taken the form of defined languages, but also there have been a number of high-level software packages. Thus, for example, there is a version of BASIC with additions of CAMAC Macro-commands, and there is even a certain move to use FORTRAN with conventional CAMAC-communicating sub-routines.

There can be no doubt that CAMAC can and indeed does perform the function we were seeking, namely, to be the intermediate standard highway system between minicomputer and terminal equipment. CAMAC is of course not restricted to minicomputers, nor indeed to the sole application of data transfer, but it is evident that it serves eminently well for the purpose of data transfers to and from minicomputers. This situation is attested well by the fact that CAMAC is used now in at least 20 countries and there are more than 70 manufacturers who supply CAMAC components. It is also probably true to say that there is no minicomputer of any general availability for which there is no CAMAC interface or driver available.

Thus, as a standard alternative to any particular minicomputer highway, CAMAC has served extremely well and has so far not imposed any particular limitation on either system configuration or on data throughput. We now have to examine the future, to see to what extent CAMAC will continue in its present form and how it might evolve.

5. CAMAC: The Present Status

It is clear that CAMAC can satisfy most data transfer requirements into minicomputers, and hence into larger systems also. And for this reason, it is not surprising that it has been used on a very large scale, not just in nuclear and particle physics but also in industry, medicine and in space research. We can see that this situation has come about because, on the one hand, CAMAC makes the exploitation of minicomputers in data-transfer activities very much more effective and, on the other hand, CAMAC does not impose on minicomputers any great limitation. Indeed, it can be said that, without CAMAC in elementary-particle physics, we just could not have achieved in any other way the results that have been attained,because we would have required many more man-years of effort (plus cost) than was available.

So, using CAMAC, up to the present at any rate, there has been a good match between the external system requirements of applications involving data transfer and the capabilities of minicomputers. To see whether this happy situation can continue, let us review the principal attributes of such total systems.

The first question is whether the number of modules available is sufficient and whether they are sufficiently "fine-grained". Here there are no difficulties, since one can have an almost unlimited number of crates, either by using many parallel branches or by crate-crate coupling of non-standard kinds, for example, by direct coupling

between the I/O highway of a minicomputer and a special(Type U) crate controller. Similarly, it is not module size which imposes any limitation, but rather what one can put in it. Indeed, modern technology has allowed us to produce certain quadruple or octal units in single-width modules, where previously we could only have one unit. The address space of CAMAC also appears so far to impose no great restriction on modularity.

So far as data width is concerned, there appears to be no difficulty, since the 24-bit CAMAC word is nearly always more than is required. Indeed, for the overwhelming number of applications (and minicomputers) 16 bits would have been sufficient. In the coming period, we will have more and more 32-bit machines, but data from external sources will probably not require more than 16 bits for a long time yet.

The question of data-transfer rate is important, since there is a Dataway limitation both in peak and average rate to about 1 Megaword per second. This is, however, an enormous average data rate, which cannot be sustained for very long by any minicomputer . Indeed, this is a fact very frequently overlooked by certain of those who complain that CAMAC does not operate "fast enough" for their application. In point of fact there is nothing to stop CAMAC handling very much faster peak data rates into modules from experimental equipment, and indeed there are many buffer modules which do precisely that, handling peak rates of maybe 10 Megawords per second, but for relatively short blocks of data. What CAMAC cannot do (in standard form) is to remove data from buffer modules at faster than 1 Megaword/sec. So, for any application that really requires very much higher average data rates, CAMAC is not of much use (but then, neither are minicomputers !).

Concerning block transfer, as has already been mentioned, CAMAC does provide both for non-standard and conventional modes, so that data transfer under DMA into most minicomputers can be readily achieved. Actually, CAMAC, although possibly somewhat slower in block transfer than some minicomputer data channels, is considerably more powerful in the number of modes available, particularly in respect of being able to partition blocks in many data streams (for example, from contiguous modules or sub-addresses). But there is a limitation both in speed and organization, which we will come to later; this limitation has more to do with applications than with minicomputers.

Demand handling is an aspect that is extensively treated within CAMAC, and certainly imposes no limitation on any attached minicomputer. On the contrary, it is difficult for most minicomputers to handle all the interrupts that can come from a complex CAMAC system, and there usually has to be additional hardware to help, often in the form of a "grader". Having said that, it is clear that the original CAMAC specifications were formulated assuming a single intelligent controller only. This is fast becoming not the case, and already in the Serial Highway definition there has been a recognition of the problem by the introduction of a "demand message" rather than a unique signal.

The final aspect of CAMAC we need to include is cost, since there is certainly an additional expense involved in coupling. Here there can be little doubt that, above a certain very low threshold, the cost of coupling equipment to a minicomputer can be very much less using CAMAC than without, provided that all costs are properly taken into account.

6. Trends in Data-Transfer Systems

We have seen that the data transfer problem has been handled fairly well up to now, and that CAMAC has played a very important role indeed. Can this continue ?

Well, there are many applications areas where CAMAC, even in its present form, can continue to give an excellent service and indeed to grow considerably. For example, in process control, in laboratory automation and in communications, there can be little doubt that CAMAC has much to offer. However, there are two important trends which strongly motivate an improvement in available data-transfer systems, including CAMAC. The first of these is new technology, which makes possible certain developments previously impracticable; the second is that experimental requirements are becoming more demanding. Let us examine these briefly to see what they suggest for future organizations of data-transfer systems.

6.1 New Technology

The main trend which we need to note is the availability at low cost of monolithic circuitry with a moderate degree of integration (LSI). Particularly of importance is the appearance of memory, both ROM and RAM, which is both of small physical size and of high-speed (200 ns cycle or even faster), and also of processors. Included in the latter are not only micro-processors of all kinds, both word and byte processors, but also bit-slice and programmable logic arrays (PLA).

The main impact that the new technology is having is that it has become possible not just to place "intelligence" (that is to say, processor + memory) physically close to the hard-wired electronic circuitry that needs to be controlled, but also to have as many separate intelligent devices as are required. In the terminology of today, the new technology and associated reduction in cost has made possible systems with "distributed intelligence". We have also to note that the reduction in cost of memory is so great that this will also have an impact on the way processing is done, replacing hard-wired functions more and more by microcode.

6.2 New Requirements

It is of course impossible to review all the new requirements of data-transfer, but perhaps we can identify the main trends, all of which depend upon the possibilities of the new technology. Possibly the most important difference from the past that is emerging is the necessity to distribute the flow of data. Whereas it used to be the case that data was more or less funnelled into a single processor (and into a single memory), it is becoming more and more important to be able to partition data in a co-ordinated way among different intelligent devices and buffers. To coin a slogan, "distributed processing implies distributed data".

Stemming from this trend, we can see that a number of desiderata emerge in any future data transfer system. For example, we need the following:

i) Ability to accommodate several processor and memory-containing devices in a modular way

ii) High-speed (200 ns per word) data transfer between such modules

iii) A defined means for recognition and servicing of demands between modules, some of which are intelligent

iv) Incorporation of arbitration between different but inter-coupled intelligent devices

It may well be that,in order to satisfy the first requirement, we may have to choose wider data and address paths, particularly if there is a frequent need for transferring large data blocks from one memory module to another. We have to remember that modern technology allows us to put as much as 64k words of store in the size of a CAMAC module, and therefore we will need 16-20 bits of address at the module (that is, sub-address) level. But we should note also that we can always indirectly address or page.

Added to the above general requirements, we can also identify a number of more special ones which arise mainly in areas such as readout from certain spatial detectors of elementary particles. The problem here, which is a special case of encoding the results of pattern recognition, is to be able to transfer only significant data from very large arrays of data sources. For example, we need to know the addresses only of "triggered" wires in large proportional chambers, without having sequentially to scan all the separate wires.

The question of pattern recognition and pre-processing is a very important one. However, it is not yet clear whether this should be included within a data transfer system or not. Certainly, if this component could be included, there would be an immense advantage because of modularity and integration into the subsequent data-transfer chain. However, there could also be great difficulties, because there are usually several orders of magnitude difference in the data rates and volume before and after pre-processing (indeed, that is the whole idea !). So, to combine both levels of data flow into one single system could be very difficult and, maybe, should not be attempted. In CAMAC, for example, this problem has been avoided by separating data signals between the front panels of modules and the Dataway, with an order of magnitude (or greater) capability in the peak speed of handling data between the two.

Let us see therefore to what extent the new technology and the new requirements can be accommodated in existing data-transfer systems and, in particular, let us review the changes that are being made in CAMAC.

7. Current CAMAC Developments

Not surprisingly, all the considerations discussed in the last section have been well known to the CAMAC community, and many attempts have been made to incorporate features into CAMAC that, while retaining downwards compatibility, makes possible certain of the new requirements. There has also been a very active discussion about how to modify the CAMAC standard perhaps to meet the growing range of applications. Let us examine some of these developments.

7.1 New Technology in CAMAC

Monolithic circuitry has been used in CAMAC modules for a long time now, but the advent of programmable devices and LSI memory has manifested itself in several different ways. First of all, we have seen the appearance of "intelligent" modules, in which programmable devices (usually driven by ROM or PROM units) have been used to replace hard-wired electronic logic. So, for example, there are available intelligent display drivers and also peripheral couplers. Also there are intelligent communications modules, particularly Modem drivers. Thus, either singly or in local clusters, we have seen the introduction of new technology at the module level. And this has had a deep influence on the pattern of data transfer in CAMAC, and may indeed have prolonged the life of CAMAC rather than shortened it. The point here is that the introduction of module processing capability has actually reduced some of the Dataway data-flow both in speed and volume. It is indeed an aspect of the introduction of pre-processing.

Another development has been the introduction of LSI units in the Crate Controller itself. In some cases here, the crate controller has been of adequate power to control completely all the crate modules without any external minicomputer at all, the functions of the minicomputer being taken over by the "intelligent" crate controller. Once again, the CAMAC system has been actually improved by the introduction of LSI technology, since the crate with intelligent controller is, for certain applications, a very economical solution.

7.2 Distributed Intelligence

Of course, many of the developments just mentioned have only been possible because certain modules within crates have had an interconnection through lines other than the Dataway. For example, there have been memory modules and other units separate from processing modules, and these have been connected by means of an auxiliary data and address path. And so we observe once again an analogous situation to that which obtained in the case of NIMS where, prior to CAMAC, we saw the emergence of numerous *ad hoc* data highways coupled to NIMS modules. There are now many different CAMAC systems in which modules, as well as coupling to the Dataway, are interconnected using a private bus (Fig.18). Sometimes even there has been a mixture of bus interconnections, with a minicomputer bus being used in DMA mode, coupled directly to a module which drives a local private bus.

In one particular area, there has been a move to define an auxiliary bus for CAMAC, and that is for the purposes of Auxiliary Crate Control (ACC). The problem here is the introduction of one or more controlling devices in a Crate in addition to the Crate Controller (CC), for which there clearly needs to be access, external to the Dataway, to the address and interrupt lines. This problem has been resolved by the definition of a supplementary CAMAC standard[9] which deals with multiple controllers in a CAMAC Crate, interconnected by an auxiliary bus (Fig.19). In this supplement, not only is the auxiliary bus defined, but so also is the bus protocol, the interconnection standard to the bus and a revision in the definition of crate controller which still permits downward compatability.

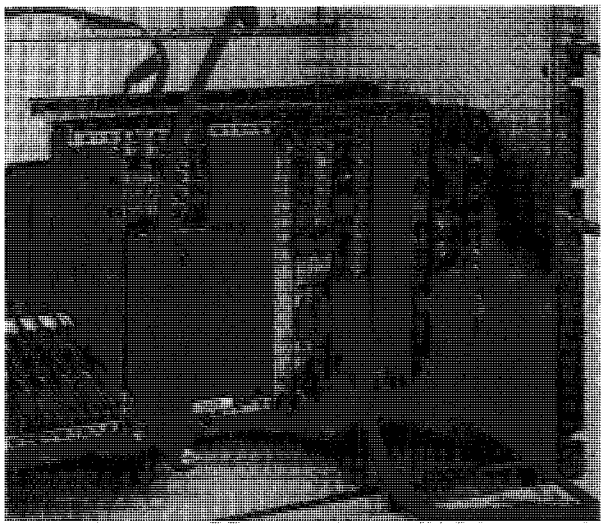

Fig.18 Example of private bus in CAMAC

It is interesting to note also that some attempt has been made to provide the means for resolving Dataway and Auxiliary Bus contention by the provision of arbitration lines. In this way a bus master can always be defined at any given time in accordance with an appropriate priority structure.

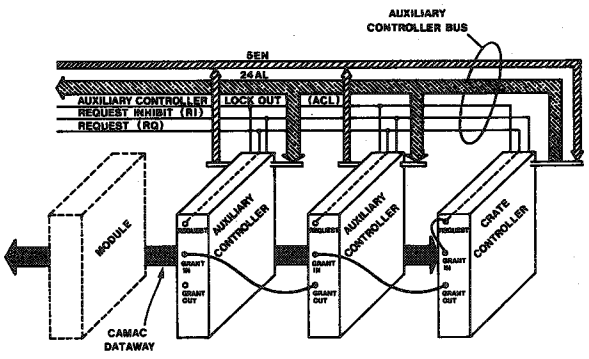

Fig.19 Multiple controllers in a CAMAC Crate

8. The Future

As always the future is difficult to predict. Particularly so in this case, since we are at a period, on the one hand, of enormous technological developments and, on the other, at a time when the investment in both software and hardware for CAMAC has been very great also. Thus there are good technical reasons for changing CAMAC, but good practical and economic reasons for not. At the same time, there still remains an immense range of problems, still unsolved, for which CAMAC can provide a most efficient and economical solution.

It is already clear that the CAMAC Dataway and associated Crate Controller impose certain speed and configuration constraints on the kind of data-transfer systems we should like to build. In any case, it is also evident that minicomputer I/O highways are also becoming much faster and, in some respects, more powerful (the Megabus of Honeywell, for example, is 72 lines wide and can have a 285 ns bus cycle). Thus, the Dataway must be replaced eventually, even if the mechanical format remains.

I believe that two quite separate developments will occur, propably concurrently. To start with, CAMAC will itself develop in two respects: firstly, there will be more and more standardization of an auxiliary bus, possibly with the incorporation of an interrupt sub-bus instead of separate lines from crate stations; secondly, there will be agreement on compatible uses of the Dataway, probably by interleaving fast data and command transfers between Dataway cycles. The second development will be to use a CAMAC-like format, possibly with a similar Dataway, but with entirely new organization and cycle time. This second development could readily be achieved already, since we have all the technological means and we know what the main parameters should be. For example, a typical specification of data highway could include:

i) Cycle time of 100 ns

ii) 32-bit bi-directional data path (with optional 16 + 16 bits)

iii) 24-bit address bus

iv) 8-bit interrupt bus

v) Control lines to permit:

   a) bus arbitration

   b) asynchronous transfers

   c) status signals

   d) block transfers in various modes

All this would be quite straightforward to do, and it would also be fairly easy to define appropriate Crate-Crate coupling, which would probably be at the level of Module-Module with a defined communications protocol for transmission over long or short distances in block-transfer mode.

If the developments just mentioned were to proceed together, then the next phase of evolution would occur in a natural way, since it would become evident fairly soon which of the two systems would dominate at any given time or, more exactly, what would be the relative balance between them. Indeed we could see a certain convergence between the two approaches, namely, the new data highway becoming identical in properties to the auxiliary bus of the old CAMAC system. In that case the transition from one system to another would occur automatically as the Dataway gradually faded from use. Such a smooth transition could be immensely beneficial since there should also be an accompanying smooth migration and evolution of software. Indeed, this could be a determining factor in the future evolution of CAMAC, since software is already the component which requires the greatest manpower effort in most modern systems.

Of course, the evolution just envisaged may not happen. And indeed there are already several different alternative suggestions to implement a completely new data acquisition scheme, such as FASTBUS [10], which has been proposed by a consortium of US Laboratory representatives. It is much too early to say which direction developments will go, but it is clear that the choice will depend finally not just on technical considerations but also on the interest and support obtained from industrial manufacturers. Let us hope that, whatever the choice, the next system developed for data transfer will be as successful as has been CAMAC.

Acknowledgment

The permission is gratefully acknowledged of GEC-Elliott Ltd. to reproduce Figs. 9, 10 and 11, and of the Hewlett-Packard Company to reproduce Fig. 14.

References

1. See, for example:
   C. Verkerk. "On-line filtering".
   Proc. 1978 CERN Summer School of Computing.

2. See, for example:
   P. Kunz. "Micro-circuits as components in high-energy physics". *Ibid.*

3. See, for example:
   L. Monrad-Krohn. "The minicomputer spectrum today and tomorrow". (Presented at this School.)

4. "Specifications for a Digital I/O Interface": BS 4421:1969, published by British Standards Institute.

5. MEDIA Reference Manual, published by GEC-Elliott Process Instruments Ltd., England.

6. IEEE Standard 488:1975.

7. H.J. Stuckenberg. "CAMAC Bibliography". CAMAC Bulletin No.13 (Sept.1975), Supplement B.

8. P.N. Clout. "CAMAC Software". IEEE Trans. Nucl. Sci. NS-24, No.1 (Feb.1977) p.484.

9. "Multiple Controllers in a CAMAC Crate". EURATOM Report EUR 6500 (1977) and USERDA Draft Report TID-26627 (1977).

10. R.S. Larsen. "Fastbus - Status of development of a standard high speed data acquisition bus for high-energy physics". SLAC Report: SLAC-PUB-2037 (Nov.1977).