CERN-Data Handling Division
DD/83/12
June 1983

FILE COPY

EXPERIENCE WITH A VIRTUAL MEMORY BASED
DATA ACQUISITION SYSTEM

A. Bogaerts, M. Dehnert, C. Eck,
A. Lacourt, J. Ogilvie, J. Petersen

(Presented at the Conference on Real-Time Computer Applications
in Particle and Nuclear Physics,
Berkeley, California, 16-19 May, 1983)

EXPERIENCE WITH A VIRTUAL MEMORY BASED DATA ACQUISITION SYSTEM AT CERN

A. Bogaerts, M. Dehnert, C. Eck, A. Lacourt, J. Ogilvie, J. Petersen

CERN, DD Division
CH 1211 Geneva 23

SUMMARY

This paper describes a Data Acquisition System which has been specifically designed to take advantage of modern operating systems. It is modular, structured as a set of independent tasks communicating via a shared data area. The design is based on the concept of circular buffers with associated data producer and (parallel) consumer tasks.

By using privileged tasks in time critical areas, a fast and efficient system has been obtained: interrupt latency of less than 100 microseconds, and data transfer speeds essentially limited by hardware (CAMAC DMA or magnetic tape recording).

The tasks may be distributed over different processors. For example, 16/32 bit multi-processors with a shared multi-port memory are used to implement systems where powerful data reduction and/or monitoring tasks are required.

The system is in use at over 25 high energy and nuclear physics experiments at CERN and in other European laboratories.

INTRODUCTION

The Data Acquisition System^{1,2} (DAS) described below was primarily intended for high energy physics experiments at the CERN accelerators. It is implemented on NORD^{3,4} computers, comprising a family of 16 and 32 bit processors. The system has been in use since 1976 and has found wide acceptance also in other European laboratories, in such applications as high energy, nuclear and plasma physics. In this report, the data acquisition system itself will be emphasized. Since several applications have been described elsewhere^{5,6}, they will only be mentioned as examples and further references will be given.

THE COMPUTER SYSTEM

DAS runs on a NORISK DATA family of 16 and 32 bit computers. A typical hardware configuration consists of a ND-100 (16 bit) processor, with a logical address space of 2 x 128 Kbytes (program and data space) and physical memory sizes up to 32 Mbytes. The hardware includes a memory management system for the implementation of a virtual memory (page size 2 Kbytes), optionally 2 Kbytes of cache memory and an interrupt system, providing 16 hardware priority levels. Each level has its own register file, including a program counter.

Task context switching efficiency has been further enhanced by micro-programming time critical parts of the operating system. Integer calculations are done on 16 bit words, floating point calculations on 32 or 48 bit words. The CPU (model CX) has a Whetstone rating of 330.

Such a system may be coupled to a 32 bit ND-500 series processor by a shared multi-port memory. The ND-500 hardware comprises a memory management system offering a virtual address space of 2×4.2 Gbytes (program and data space) and two identical cache memories of maximum 64 Kbytes each for instructions and data. Floating point calculations may be done on 32 bit words (Whetstone rating of 1850 on model 540) or 64 bit words (Whetstone rating of 1200). The ND-500 has been designed to use the ND-100 as I/O processor. There are various multi-port memory systems available. The most advanced model (MPM-5) provides a twofold interleaved 64 bits wide data path into the ND-500 and has a DMA bandwidth of 20 Mbytes. There is no practical restriction on the number of ports.

In addition to the standard range of computer peripherals, two types of CAMAC interfaces are available: a Single Crate Interface allowing the use of a maximum of 16 crates, or a System Crate Interface which may connect a maximum of seven standard CAMAC branches of seven crates each. DAS may use either of these interfaces. There is a wide choice of computer communication hardware, either offered by the computer vendor (based on RS 232 serial lines, HDLC or Ethernet) or developed locally (CERNET⁷, interfaced through CAMAC).

The principal software component on which DAS is based is a virtual memory, multi-mode operating system which offers a time-sharing service for program development and includes real time and inter-task communication features. Real time programs run under supervision of the operating system and are scheduled for execution according to their software priority. The typical interrupt response and context switching time is approximately one millisecond.

The operating system allows a restricted number of privileged tasks running on reserved hardware priority levels of the ND-100. They do not have access to all the facilities of the operating system since they may run at an even higher priority level. Their advantage is a very much reduced context switching time. This is improved by approximately a factor ten, and the interrupt response time is better than 100 microseconds. DAS makes no distinction between ordinary or privileged tasks. In the following the term task will be used for both.

The segment is a fundamental concept in the operating system. It is an area which is contiguous in virtual memory, and contains code or data. During program execution, segments may be paged on disc. It is also possible to fix a segment in physical memory. This may improve the efficiency and is essential in the case of DMA transfers to the segment. A segment contains always an integral number of pages. Its maximum size is 128 Kbytes for the ND-100 and 128 Mbytes for the ND-500. In most cases, only one task is loaded per segment. It is possible to link a task to a data segment, even dynamically, and to identify ND-500 segments with ND-100 segments. DAS task communication is based on the use of shared data segments.

DAS STRUCTURE

DAS is a modular Data Acquisition System, structured as a set of independent tasks which communicate via a common data area. DAS could be implemented to run under any multi-tasking operating system which provides only very few, generally available facilities:

- the possibility to activate and suspend the execution of a task from another task (the supervisor task)
- activation of a task via external interrupts
- task communication using only one common data area.

DAS consists of a set of (circular) buffers and a set of tasks. All data is structured into records or "events" and has to flow through buffers. Data is put into or taken out of a buffer by tasks which are called "producers" or "consumers". No assumptions are made about the relative speeds of these tasks, which need therefore to be synchronized. For example, a producer has to be suspended when a buffer becomes full, and, conversely, a consumer has to wait when a buffer becomes empty. This application is slightly more complicated than the classical consumer/producer problem⁸, because DAS has to allow for several consumers of the same data. For instance, in many applications in high energy physics experiments, it is not sufficient to record data onto magnetic tape. It is necessary to provide event display and monitoring tasks which are all consumers of the same data. Furthermore, certain tasks may only consume a fraction of all the data. This leads to the notion of parallel consumers, i.e., a set of tasks which may consume the same data asynchronously. An event may therefore only be deleted from a buffer if each of these parallel consumers has received the data or has at least been considered if entitled to only a fraction.

Various solutions of the producer/consumer problem have been suggested⁹. All of these can easily be extended to our problem. The solution chosen is the use of a monitor: all tasks wishing to produce or consume data have to invoke the DAS supervisor as summarized in table 1. The supervisor has been implemented as a privileged task.

Each task may write into several different buffers (at present maximum 5) and may read from several different buffers (at present maximum 10). Each buffer may receive data from any number of producers, which have to proceed serially. If several producers request buffer access at the same time, they will be queued. This restriction has to be imposed to allow for variable length events, whose size is not known beforehand. The number of parallel consumers that may be connected to each buffer is limited (presently, to 10). This is illustrated in figure 1. These constraints are rather arbitrary and are imposed only to limit the size of certain tables. A task may at the same time be a producer and consumer. A typical example of such a "transducer" is an event filtering task.

Consumers may be coupled to a buffer in various ways:

- a consumer may take all the data flowing through a buffer (typically, a task which records all data on magnetic tape).

- a task may only temporarily be connected to a buffer (typically an event display task, which is only connected to a buffer when a display has been requested).

- a task may run at its "natural" speed, requesting as much data as it can handle within the limits imposed by the operating system (e.g. priorities may regulate the competition for CPU utilisation). This is essentially the same mode as the previous.

- a task may get a user specified fraction of the data (typically, a monitoring program which samples data).

- a task may run freely but is guaranteed a certain minimum fraction. This mode is obtained by switching between the two previous strategies.

It is possible to change the mode for each task individually even during a run.

DAS CONFIGURATIONS

Data buffers and tasks are the components used to build actual systems. They are obtained by defining the number and size of the buffers, the number of tasks and their interconnections. For each task and buffer, there exists a descriptor, called Task Control Block (TCB) and Buffer Control Block (BCB) respectively. These contain the characteristics of the tasks and buffers such as the task name, the buffer size and memory address, the task/buffer interconnections as well as workspace to hold buffer pointers and statistics about the amount of data flowing through a buffer. Tasks and buffers may be addressed using logical task and buffer numbers, in much the same way as logical unit numbers are often associated with physical I/O devices. The task itself has no direct knowledge about the characteristics of the buffers it uses nor the topology. If several buffers are accessed within one task, they are addressed as the first, second etc. input or output buffer. The TCB contains the necessary information to translate these indices into logical buffer numbers. Thus the topology is defined in the configuration, not in the task. This offers a great deal of flexibility, since those tasks which can be made independent of the data such as magnetic tape recording, CAMAC reading and data-link servers may be used in many different configurations without changes.

Modifications occur frequently in high energy physics experiments. These may require different DAS configurations, as illustrated in figure 2. Figure 2a shows a typical example of a simple DAS configuration, consisting of a CAMAC reading task, one buffer and three consumer tasks (magnetic tape recording, event display and sample analysis). Figure 2b shows the introduction of an event filtering task before the data is recorded on magnetic tape. This configuration is obtained from the previous one by the addition of the filtering task and a second buffer without any changes to the other tasks. It is also easy to undo these changes if it is later decided to migrate the event filtering task to a micro-processor.

DAS configurations are built using an interactive program. They are stored as disc files but have to reside in memory when DAS is active. It is possible to have several alternative configurations, for example main data taking, calibration runs and playback from magnetic

tape may in part use the same tasks in a different topology. Examples of simple DAS configurations are given in figure 2.

Whenever data formats and procedures on data can be standardized, it is possible to provide standard tasks. This is for example the case for the CAMAC read-out task (where facilities exist to specify the experiment dependent CAMAC part), magnetic tape recording, event displays, an event distributor to time-sharing background users (convenient for program development) and data link drivers. Other tasks are of course highly specific. It is often possible to provide skeleton tasks. This may be supplemented by additional standards on event formats¹⁰, memory management schemes¹¹, histogramming^{12,13}, CAMAC programming¹⁴, external data formats¹⁰ and graphics I/O¹⁵. In these cases, it is possible to provide examples, libraries, documentation and sometimes complete support programs.

DATA BUFFERING AND DATA FORMATTING

Data buffers have to be memory resident to allow direct (CAMAC) DMA transfers and access by privileged tasks. The mechanism used is the "fixing" of segments in a contiguous area in physical memory. Buffers and even events may be so large that they exceed the virtual address range of the ND-100. For historical reasons, there are two types of buffers and events: "small" and "big". Small buffers cannot exceed 128 Kbytes, and all buffer address calculations are done in 16 bit integer arithmetic, using virtual addresses. Big buffers may have any size, and address calculations are done in 32 bit integer arithmetic using physical addresses. Similarly, small events cannot exceed 128 Kbytes, and they may be held in either small or big buffers. Big events can have any size, but can only be held in big buffers.

Data formats have been kept as simple and general as possible. Events have to be structured as logical records, where the first word indicates the size (a 16 or 32 bit integer). The next word may be used as an event identifier, and the DAS supervisor allows for event selection according to this type. For many DAS tasks, this is sufficient. It is possible however to adopt additional rules and conventions, which allow the use of more advanced data structures and support packages in monitoring, data reduction and on-line analysis tasks. This is left to the application programs and the data producers are responsible for further formatting. There exists a CERN standard for recording of data onto magnetic tape¹⁰, the formatting of data is done by the tape writing task.

Data in a DAS buffer may be accessed using two levels of subroutine calls. At the lowest level, only pointers to the data are passed over to the task. A consumer task which wants to access an event calls a subroutine (ZMON) which invokes the DAS supervisor. If data is present in the buffer and if the task is entitled to more data (depending on the mode), the next event which has the correct type is locked in the buffer and pointers to the event are returned to the requesting task. When the task has finished processing the data, the supervisor has to be invoked once more. This is similar for data producers, except that they can get access to a free area in the buffer which has to be filled with events.

At the next level, data is copied from the task into the buffer or vice versa. Three routines are available:

ZPUT which copies part of an event, a complete event or several events from a local data area into a buffer

ZGET which copies an event from a buffer into a local array

ZPGET which copies part of an event into a local array.

ZPUT is used by producer, ZGET and ZPGET by consumer tasks. All these routines use the lower level by calling ZMON. They have the advantage that buffer access is transparent, buffers are not unduly blocked due to event locking and discontinuities in the data area caused by circular buffers (data "wrap-around") are avoided. ZPGET allows copying of any part of an event. This operation may be repeated on the same event, until the user signals the last call to the routine. Its main purpose is to get hold of "sub-events" which may be located using descriptors which are themselves part of the data. If complicated data structures are used, these routines may again be hidden in higher levels of software. Most user tasks make use of higher level routines.

CONTROL PROGRAMS

Control programs are used to steer DAS and control the experiment. Their purpose is to convey commands to the system and, conversely, to present the physicists who run the experiment with the necessary information. Obviously the human interface forms an important aspect of control programs.

In the same way that the data acquisition proper has been modularized into tasks, control functions have been split into several control programs. These are not linked to data buffers, but they have their own shared data area for communication. A Control program Control Block (CCB) is associated with each control program which makes it a logical entity. As for DAS tasks, there exists an interactive configurator which produces a disc file.

A simple message system which uses the shared data area allows control programs to exchange information. DAS tasks may also send messages. This method is used to print error messages, but also for example to terminate a run when the tape writing task detects an end of tape condition or the CAMAC read-out task has transferred a pre-set number of events. In principle, any operator command may also be issued by a task or (another) control program.

Most control programs communicate with the operator. A fundamental component is the command processor, which is available as a subroutine package. This can operate either in command mode, where it expects a text string as input, or in menu mode where a choice of commands is presented in a tree-like structure. The operator can choose the mode. It has such features as command abbreviation and line editing (in command mode). It also allows the use of various touch screen devices (in menu mode), permits parameter anticipation in which case prompts are suppressed, parameter defaulting and optional display of default values (in either mode). All commands may be logged on a printer or a disc file. It is also possible to create and execute "procedure files" (disc files containing commands). Messages may interrupt the

command processor when it is waiting for command input and these may be interpreted and executed as commands, e.g. detection of the end of tape marker during tape writing may terminate a run.

Global control of the experiment is centralized in the main control program. This supervises the choice of configuration, the main data stream, the allocation of I/O devices and the use of secondary control programs. The main control program may temporarily give its control terminal to another control program, which appears to the operator as another level in the command structure. It is also possible to run secondary control programs remotely, e.g. close to the detector or on a graphics I/O station. Hooks for user extensions have been provided. It is possible to add commands at all levels of the command hierarchy and dummy routines have been inserted in all places which are likely to be modified, e.g. at run start and termination.

The standard control programs which are now available are the main control program, an error reporting program and a histogram manager. A CAMAC system debugger is under preparation. Control programs which are already in use at some experiments are a DAS status display and a magnetic tape manager.

THE HISTOGRAMMING SUBSYSTEM.

Histogramming is an important activity for many DAS tasks. The main problem encountered on 16 bit computers has always been lack of memory. With the advent of virtual memory systems and large physical memories, this has changed and the restriction which is now mostly felt is lack of virtual address space.

The histogramming subsystem on the ND-100 was designed to provide a large histogramming space to each DAS task, minimize the use of user address space and preserve the independency of tasks. Separate segments (data areas of maximum 128 Kbytes) are used for histogram data storage. A directory is also included. Each task may use four segments, giving a maximum of 4 x 128 Kbytes per task. Since each segment contains its own directory, there is no ambiguity about histogram names and identifiers unless tasks want to share the same histogram segment.

The histogramming proper is based on HMINI¹². This is a computer independent package which has been especially optimized for 16 bit computers. It allows efficient integer histogramming in one or two dimensions using 16 bit words as basic storage units and has also provisions for floating point histogramming. HMINI can also be used on 32 bit machines. In practice, the packages which were traditionally used on the off-line computers, such as HBOOK¹³, are often preferred here.

The histograms are presented by a separate control program, the histogram manager. The use of a data presenter program offers many advantages. It saves code and offers a more uniform way to present the data. It contains facilities for printing, plotting, zooming, erasing, comparing etc. Histograms may be looked at during and after a run. It is also possible to collect statistics across several runs. It is left to the task to reset all histograms automatically at the start of a run. Additional features such as histogram storage in

computer independent format and archiving are also under consideration.

Graphics I/O is based on the mini-GD3 package¹⁵.

MULTI-PROCESSOR SYSTEMS.

The modular structure of DAS lends itself rather easily to the implementation of multi-processor systems. The DAS supervisor and all tasks which are synchronized with respect to the buffers, run on the ND-100. In practice, some of these tasks may be considered as mere "shadow" tasks which in turn control another processor (e.g. a micro-processor) or a process (e.g. on the ND-500). They may also drive a dedicated data link or serve a more general network. These tasks may be implemented as privileged tasks if their response time is critical. On the other hand, existing programs such as data link drivers may with minimum effort be converted into a DAS task.

Most multi-processor configurations fall into one of four categories:

- micro-processors used for front-end data acquisition

Such data acquisition systems employ micro-processors and memory buffers integrated with the electronics trigger and read-out system. The micro-processors may filter, pre-process and compact the data. An interrupt is sent to the host computer whenever an event is ready. The shadow task on the ND-100 is in this case the standard CAMAC read-out task which is simplified since the data is already formatted in an intermediate memory buffer. Examples of such systems are the EHS experiment at the CERN SPS, which uses several ESOP micro-processors for second level triggering and data reduction¹⁶, and UA1 at the CERN proton-antiproton collider which uses four parallel fully buffered 168/E micro-processors¹⁷.

- data spying at the CAMAC level

Two DAS systems may be connected to the same CAMAC read-out chain. Usually, the two systems are not entirely symmetrical. One system is considered to be the master, which controls the main data flow and the recording on magnetic tape. The second system which may read the same data (or part of it) provides additional power for monitoring tasks. Either of these two systems may be further enhanced by the addition of a 32 bit back-end processor. In case of breakdown of one system, it is usually easy to reconfigure the installation and to continue data taking in some degraded mode on the remaining system. Another advantage of such a dual system is the possibility for independent hardware and software development (which is greatly appreciated by the software developers). Examples of such systems are the NA4 and UA1 experiments at CERN.

- shared multi-port memory systems

If CPU power is lacking, it is possible to combine the 16 bit ND-100 with the 32 bit ND-500 processor, tightly coupled by shared multi-port memory. The enhanced power of such a system has been used to implement more powerful on-line monitoring tasks (a fraction of the

events may be completely reconstructed on-line), for event filtering and for on-line calibration. Since the ND-500 is multi-programmed, it is possible to run several data consumer and producer tasks provided that they are accompanied by their "shadow" counterparts on the ND-100. Data access is very efficient since the DAS buffers are directly addressable by both processors. Examples of such systems can be found at the PETRA experiment JADE at DESY, and NA4 and UA1 at CERN.

- direct data-links and networks

Data-links and, in general, networks may be used to connect DAS systems to any other machine. In many cases, dedicated point to point connections, based on non-standard hardware and software, have been used. If standard networks are chosen, a protocol suitable for program to program communication has to be used. At CERN, CERNET mini-links are often used to connect systems. These are private point to point connections, based on CERNET hardware. The implementation of such systems requires the addition of a consumer or producer tasks to send or receive data over a link. Standard link drivers can be modified into DAS tasks. Connections based on the computer manufacturers communication hardware and software (RSC 232, HDLC, etc.) have also been used.

Examples of such systems are the JADE experiment at DESY which runs a private link for central data recording in the computer centre, and the neutrino-oscillation experiment WA18 at the CERN SPS, running a CERNET mini-link between DAS systems at two detectors located approximately 600 m apart.

CONCLUSIONS

The concept of circular buffers with associated producer and consumer tasks has been applied to obtain a family of data acquisition systems. Various functions such as CAMAC data acquisition proper, data recording, data reduction, on-line (sample) analysis as well as experiment control may be implemented as independent "tasks" or "programs". Different systems or "configurations" are obtained by combining buffers, tasks and control programs. This approach proves to be very flexible. The resulting systems may be used in a large class of applications. Evolutionary changes in the environment require usually a re-configuration of the system whereby most of the existing software modules may be retained without modification.

Certain functions of the data acquisition system may be transferred to another processor. Such multi-processor configurations are implemented by "shadow" tasks. These have a dual purpose: they are coupled to the data acquisition system in the standard way, assuring synchronisation, and drive another process or processor.

ACKNOWLEDGEMENTS

We would like to acknowledge the participation of many colleagues in the DAS development, in particular P.Baehler, M.Demoulin, M.Ferran, D.Jacobs and A.Perrelle from CERN and J.P.Fournier and B.Rothan from Saclay.

TABLE AND FIGURES

function:	issued by:	effect:	side effects:
reserve buffer space	producer task	the requested amount of free buffer space is exclusively reserved and pointers are returned to the producer	producer may be suspended if the amount is not available or if another producer reserved already
release buffer tasks space	producer task	the produced event(s) whose size cannot exceed the reserved amount becomes available to consumer tasks	consumer tasks waiting for data or producer waiting for buffer access may be re-activated
lock next event	consumer task	the next event to which the consumer task is entitled is made available and pointers are returned	the consumer may be suspended if no data is available or the consumer is not entitled to it
unlock event	consumer task	the event may be deleted from the buffer as far as this particular consumer is concerned	free buffer space may become available and producers waiting for free space may be re-activated

Table 1. Summary of buffer operations requested from the DAS supervisor by producer and consumer tasks. Producers must always issue pairs of reserve/release requests, and consumers pairs of lock/unlock requests in this order.

Figure 1a. DAS consumer/producer task.
Each task may read data from maximum 10 and
write data into maximum 5 buffers.

Figure 1b. DAS buffer with task Input/Output.
Any number of tasks (n) may write data into a DAS
buffer but maximum 10 consumers may be accommodated.

Figure 2a. Example of a simple DAS configuration consisting of one buffer and four tasks. The CAMAC reading task is the only data producer. The tape recording task receives all data, the display task only on request and the analysis task may use the remaining CPU capacity to sample as much data as possible.

Figure 2b. Example of a more complicated DAS configuration which includes data pre-processing. This configuration may be obtained from the configuration shown in figure 2a by the addition of the data reduction task. The other tasks need not be altered.

REFERENCES

- 1) "DAS User's Guide", DD/OC/ND/81-1, DD Division, CERN (1982)
- 2) "DAS Reference Manual", DD/OC/ND/81-2, DD Division, CERN (1983)
- 3) "ND-100 Reference Manual", ND-06.014.02, NORSK DATA A.S. (1982)
- 4) "ND-500 Reference Manual", ND-05.009.02, NORSK DATA A.S. (1982)
- 5) S.Cittolin, "The UA1 Data Acquisition System", presented at the Int. Conf. on Instrumentation for Colliding Beam Physics, Stanford, Calif., 17-23 February 1982
- 6) P.Baehler, A.Bogaerts, C.Eck, M.Ferran, D.Jacobs, A.Lacourt, J.Ogilvie, H.Overas and J.Petersen, "Clusters of 16/32 bit Minicomputers in High Energy Physics Experiments at CERN", Comp. Physics Communications 22, 285-292 (1981)
- 7) J.M.Gerard, "CERNET - A High-Speed Packet-Switching Network", CERN Yellow Report 81-12, CERN (1981)
- 8) E.G. Coffman and P.J. Denning, "Operating Systems Theory", Prentice-Hall (1973)
- 9) P. Brinch-Hansen, "Monitors: An Operating System Structuring Concept", Comm. ACM 17, 10, 549-557 (October 1974)
- 10) "EPIO Manual", DD/EE/81-2, DD Division, CERN (1982)
- 11) "ZBOOK User Guide and Reference Manual", DD/US/73, DD Division, CERN (1982)
- 12) "HMINI Common Histogramming Library", DD/OC/82-2, DD Division, CERN (1982)
- 13) "HBOOK", DD/EE/81-1, DD Division, CERN (1981)
- 14) D.Burckhart, S.Cittolin, J.O.Petersen and L.Tremblet, "Standard CAMAC Subroutines for On-line computers at CERN", IEEE Trans. on Nuclear Science NS-28, 5, 3845-3849 (October 1981)
- 15) "Mini-GD3", DD/US/90, DD Division, CERN (1982)
- 16) D.A.Jacobs, "Application of ESOP, a fast microprogrammable processor, in high energy physics experiments at CERN", Comp. Physics Communications 22, 261-267 (1981)
- 17) J.T.Carroll, M.Demoulin, A.Fucci, B.Martin, A.Norton, J-P.Porté, K.M.Storr, "Data Acquisition using the 168/E", presented at the "Three day in-depth review on the impact of specialized processors in elementary particle physics", Padova, Italy, 23-25 March 1983