SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 16(6), 517–529 (JUNE 1986)

Integrating Distributed Array Processing into EMAS 2900

MICHAEL W. BROWN
Edinburgh Regional Computing Centre, University of Edinburgh, Edinburgh EH9 3JZ, U.K.

SUMMARY

This paper describes the integration of initially one, and later a second, ICL Distributed Array Processor (DAP) into a dual ICL 2972 installation running the Edinburgh Multi-Access System (EMAS). The alterations made to each level of the operating system are briefly described, along with the methods for mounting some of the relevant ICL DAP compiler software. Software enhancements to enable multi-programming of the DAP and a locally designed method for DAP data area expansion are also described. A brief résumé of the user facilities available is included, along with operational policies and some utilization figures.

KEY WORDS EMAS ICL 2900 Time-sharing Parallel processing Array processor

THE DISTRIBUTED ARRAY PROCESSOR

The ICL Distributed Array Processor (DAP)[1] consists of 4096 bit-processor elements (PEs) in a 64 × 64 array, each able to perform simultaneous calculations. Each processing element has its own local store of 4096 bits, making a total of 2 megabytes in the DAP store. In addition there is a master control unit (MCU) which decodes the instruction stream and initiates simultaneous PE execution.

The DAP store may be regarded as a three-dimensional array of bits consisting of 4096 planes, each of 64 64-bit rows (or columns).

In order to be loaded with program and data, and for execution to be initiated, the DAP is connected as a block of main storage to a host computer system, which in the case of the MSI 64 × 64 DAP must be a P series ICL 2900 installation.

This unique architecture provides, for particular problems, substantial computing power. In addition, when the machine is not being used as a DAP, the 2 megabyte DAP store is available to the host computer system as a further 2 megabytes of main store.

THE EDINBURGH MULTI-ACCESS SYSTEM

EMAS 2900 is a general purpose time-sharing operating system for ICL 2900 series machines. It is a re-implementation of a system designed for the ICL System 4–75 machine. The original system was developed at the University of Edinburgh and has been adequately described elsewhere.[2] The re-implementation exercise for 2900 has also been described,[3] but some of the aims and basic structure of EMAS will be described here, along with the configuration of the dual 2972 (later 2976) hardware to which the DAPs were attached.

0038–0644/86/060517–13$06.50
Received 19 July 1984

The prime function of EMAS is the support of a large number of interactive users with the secondary function of providing a reasonable batch throughput. EMAS is a virtual memory system—files are accessed through virtual addresses as if they were an extension of real memory. In addition, as much use as possible is made of shared code and data. EMAS also enforces fairness by attempting always to give quick response to small interactions, while keeping the CPU occupied with the larger computations. The system runs with a minimum of operator intervention, has effective error recovery features and the ability to run on a reduced configuration, at some cost to the total performance of the system, in the case of the breakdown of parts of the hardware configuration.

The central part of EMAS is the Supervisor. This consists of a minimal resident global controller which contains the scheduler, the active memory and paging managers and the device drivers, and many separate incarnations of a local controller. The function of the local controller is to manage the virtual processor which is available to each user and to provide various services to the user process.

The next level in the fundamental structure of EMAS is the paged Supervisor called Director. Director maintains the file system and virtual memory access, and provides a procedural interface to the system facilities.

There are several paged system executive processes which exist when EMAS is running. The volumes manager maintains the back-up and archive tape file stores. The spooling manager controls the slow devices and RJE queues, and the batch job queues. The file transfer executive gives full NIFTP–B(80) file transfer protocol (FTP) facilities, and the mailer executive provides full JNT MAIL facilities.

It is perhaps significant that the EMAS system is almost entirely written in the high level language IMP[4, 5] as is all of the locally written DAP support software, and that the object code format and system interfaces are quite different from the manufacturer's VME system.

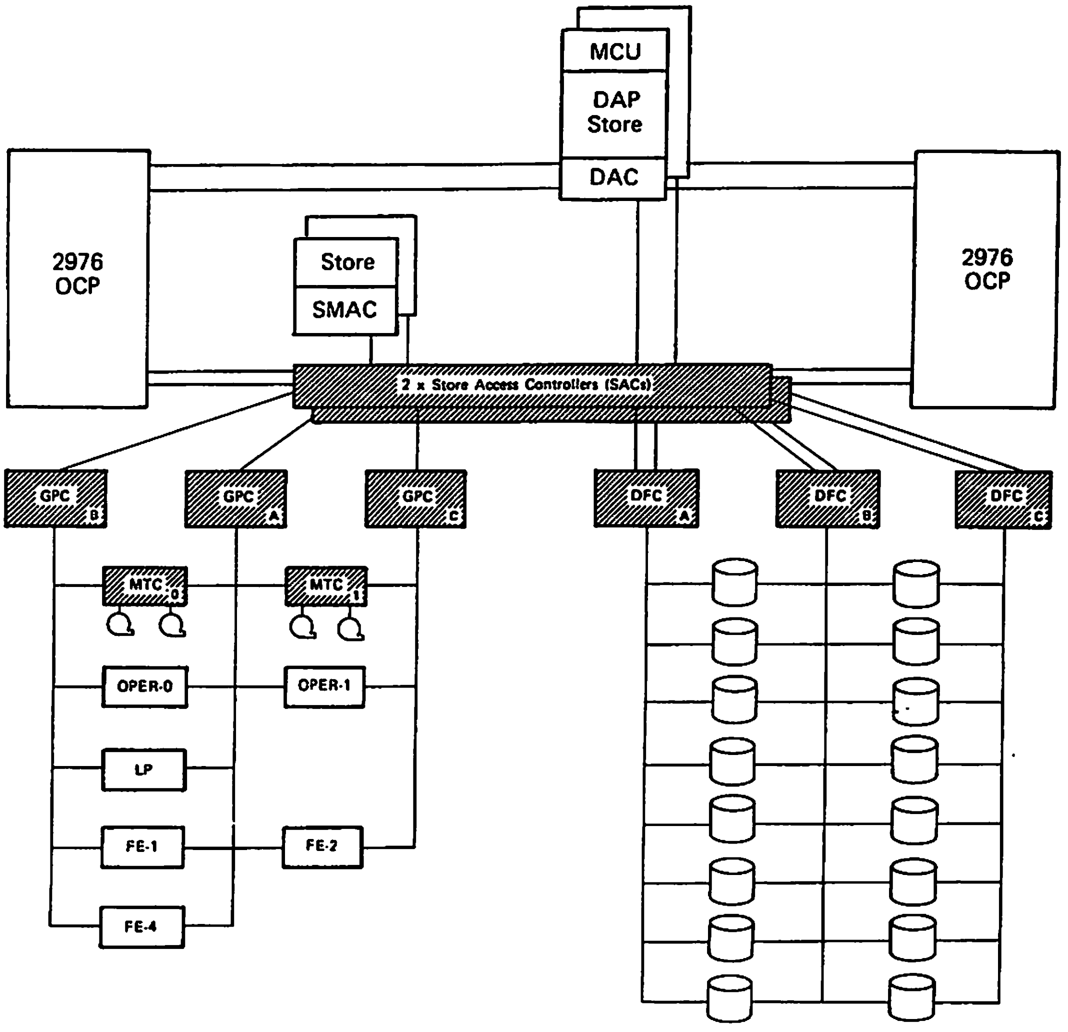
Figure 1 shows the current 2976 configuration.

ACQUISITION AND BACKGROUND

The first ERCC DAP was mainly financed by a grant from the U.K. Science and Engineering Research Council (SERC) to the Physics Department at the University of Edinburgh, but also by a software contract between ERCC and ICL. The substantial support given by SERC was in response to proposals for projects in subnuclear physics and condensed matter physics. The prime movers behind the proposal were Professor Wallace, Dr. Pawley and Dr. Bowler of the Physics Department who wished to attempt large-scale computer simulations of the molecular solid state,[6] but were hampered by lack of computational resource. The original programs were developed on the DAP installation at Queen Mary College (QMC), London, but communications difficulties and poor turn-round prompted the initial proposal that a DAP be sited at the University of Edinburgh, and be attached to the dual 2972 system running EMAS.

The initial proposal made at the end of 1981 was for a second national DAP to be installed at Edinburgh, but the weighty and detailed proposal failed in December as QMC's DAP was at that time underused. The proposal was recast to request the DAP merely as an Edinburgh resource and, in the end, this was accepted.

It says much for the faith that SERC have in the ERCC that they were prepared to invest a significant amount of money towards the installation of a piece of hardware and



Key:

OCP	– Order Code Processor	GPC	– General Peripheral Controller
DAP	– Distributed Array Processor	MTC	– Magnetic Tape Controller
DAC	– DAP Access Controller	OPER	– Operator Console
MCU	– Master Control Unit	LP	– Line Printer
SAC	– Store Access Controller	FE	– Front End
SMAC	– Store Multiple Access Controller		(Communications Processor)
DFC	– Disc File Controller	Mbyte	– 1,048,576 bytes

(Shaded boxes represent control units)

Figure 1. ERCC dual 2976 configuration

its associated software that were completely untried and unknown in the EMAS environment.

The second DAP was acquired by the Physics Department directly from ICL. Its arrival was coincident with a significant upgrade to the ERCC dual 2972 installation, with the upgrading of the processors to 2976 standard and the supply of further on-line disk space.

DAP SOFTWARE IMPORTATION

A DAP program exists in two distinct parts. One part, called the 'host' program is written in standard FORTRAN and, as its name suggests, it runs on the host 2900 mainframe. The other part, called the 'DAP' program, is a subprogram written usually in a special dialect of FORTRAN called DAP FORTRAN, which is called as a subroutine from the host program and actually performs the DAP processing. DAP FORTRAN[7] looks much like standard FORTRAN, but for the inclusion of vector and array operations and the absence of I/O statements.

ICL implemented compilation and support software for the DAP to run under their VME (virtual machine environment) operating system. The ICL DAP FORTRAN compiler does not compile directly to DAP machine code, but instead to a macro format of APAL (array processor assembly language)[8] called AMF (APAL macro format). The AMF resulting from a DAP FORTRAN compilation then needs to be assembled by the ICL APAL assembler, resulting in the production of CIF (consolidator input format). The file of CIF then needs to be input to the ICL consolidator which is used to link modules of CIF, satisfying external references in a cascade fashion, to produce a file of OMF (the VME object module format). This is the sequence of operations required under VME; under EMAS a further stage is required to convert the module of OMF into an object file conforming to EMAS standards.

OMF copies of the ICL compiler, assembler, consolidator, subroutine library, run-time diagnostics, AMF macros and message text modules were obtained from ICL—a total of 273 files— and each was put through an OMF-to-EMAS object file converter. The converter is an established EMAS utility and has been used in the importation of all ICL software mounted on EMAS to date. The EMAS object file is totally different in layout from its ICL OMF counterpart, the code and data, shareable and unshareable parts all being laid out in separate areas as required by the EMAS loader. The results of conversion were object files conforming fully to the EMAS standards, each with code entry points, data areas and code and data references.

ICL compilers use a common interface to the VME system called Compiler Environment (CE). CE is a standard VME product and consists of an extensive procedural interface to utilities which create files, create and extend workspace areas, handle listing control, interpret compiler options, read source records and output OMF records, etc. CE is totally VME specific, but to support ICL compilers on EMAS a partial CE implementation had to be written. Fortunately, much of the work in implementing a partial CE on EMAS had been done for the exercise in importing the ICL COBOL compiler, although extensive alterations and additions were required to this basic framework to support the DAP compiler and assembler.

The ICL consolidator uses a different standard ICL interface to the system called Compiler Target Machine (CTM). Once again it was necessary to produce a partial

CTM interface on EMAS, and once again part of the work had already been done for the COBOL run-time support.

The importation of the consolidator was more difficult than that of the compiler and assembler, particularly with regard to handling input and output OMF-type data records. In addition a mechanism had to be devised to enable the consolidator to search for and satisfy references from the standard ICL DAP subroutine library and for the appropriate modules to be located and input. All the possible subroutine entry points are held in a table which indexes into a second table which holds the name of the subroutine module. On successful conclusion of the search for a reference, a currency (in VME terms) or stream (in EMAS terms) is set up, and the module records are input directly by the consolidator.

The search keys for the standard ICL subroutine library, and an extensive subroutine library supplied by QMC, are built in to the consolidator support software on EMAS. User-written libraries may also be searched, in which case the user enters the name of such a library onto a library search list maintained in a special option file held for each user.

As is described below, the user interface to the DAP software on EMAS is totally different from ICL's interface—it was designed to conform to EMAS standards and policies. The final stage of the consolidation process on EMAS is the conversion of the OMF file produced by the consolidator into an EMAS-type object file. A modified version of the OMF converter was incorporated into the consolidator support software, and is called automatically on successful termination of the consolidator.

The only run-time support software that was imported into EMAS was the ICL DAP diagnostic package. ICL diagnostics follow a different approach from those of EMAS, and their interface to the system is via another standard ICL product called OPEH (object program error handler). A partial OPEH interface had to be written, this time from scratch, as OPEH-type diagnostics had never before been implemented on EMAS. Fortunately the OMF diagnostic records are not discarded during the object file conversion process, but are stored in a known place at the end of the EMAS object file. Except for some simple procedures to be written to interface directly to the DAP diagnostic procedures, all that was required in the incorporation of the ICL DAP diagnostics into the EMAS environment was the supply of a set of procedures to access the stored OMF diagnostic records.

The only other standard ICL DAP software imported was the DAP simulator. This is a software emulation of the DAP and can be useful for testing short sequences of DAP program without the need for an actual DAP to be available. The simulator operates very slowly indeed, but has been found useful in testing library subroutines.

DIRECTOR LEVEL

Under EMAS, access to the DAP from the user level is controlled by the paged supervisor, Director. The user program, through the DAP access commands supplied at user level, initiates DAP activities via Director. Director sets up necessary virtual storage mappings, moves the DAP program and data into the DAP, and requests Supervisor to initiate DAP execution.

A more detailed description of Director's role in DAP processing may be found elsewhere.[9]

The higher level software can request Director to initiate a number of DAP-related activities:

(a) claim contiguous DAP area
(b) release claimed DAP area
(c) start the DAP at a supplied program counter
(d) stop the DAP (forced termination)
(e) request DAP status information.

Director allows a series of DAP programs to be active at any time, and each is given a time-sliced share of the actual DAP. In practice this allows a short interactive program to overtake a long batch job, whereas in theory the two jobs interleaved take the same total time as if they were run sequentially. Supervisor maintains a DAP service queue, and notifies Director when programs are to be swapped. Currently, the time-slice is about two minutes, although this can be altered.

Director maintains a 'virtual' DAP for each process, consisting of a 2 megabyte file in each user's virtual memory and, in conjunction with the scheduling controlled by Supervisor, it moves in and out of the actual DAP store the 2 megabyte virtual DAPs for each process on the DAP service queue. When the user level software makes a DAP claim call to Director, the request is put on a queue to be serviced at an appropriate time. There is an overhead incurred by the swapping of different user programs, and the DAP is idle in the interval between time-slices, but the throughput degeneration is slight and is an acceptable penalty for the resulting increased flexibility. With only one DAP in service the allowed concurrency limits were 4 interactive jobs and 1 batch job; with the introduction of the second machine this was raised to 4 interactive plus 1 batch job per DAP. The limits can be altered dynamically by operator command.

Director is also responsible for recording DAP usage (for accounting purposes) and for enforcing DAP processing time limits. Currently the access scheme allows for short interactive access and longer batch access throughout the daytime prime shift. Both limits are increased, automatically, throughout the evening, and gradually decreased throughout the early morning. Higher limits are set at weekends than during the week. These policies encourage the running of long batch jobs at night and the weekend, it being desired not to run long jobs at prime time during the day in case the total mainframe user load requires the release of a DAP for use as main store, or the system is shut down over lunch-time for maintenance. The interactive and batch limits are altered automatically throughout the day, but they may be overridden by operator command to suit requirements as necessary.

SUPERVISOR

It is the resident Supervisor of EMAS that controls the DAP at its most basic level, and is ultimately responsible for the allocation of DAP store to users as a DAP, and for the starting, scheduling and stopping of DAP processing. A more detailed description of the Supervisor control of DAP processing may be found elsewhere.[9]

When the EMAS system is loaded, if the DAPs are included in the configuration as recorded on the system reconfiguration unit, then Supervisor will recognize the presence of the DAPs and include them in the system configuration, using them as 2900 store initially.

The DAPs may be brought in and out of the configuration, and their status altered from store to DAP and vice versa, by simple operator commands. In addition, Supervisor controls the dynamic reconfiguring of the DAPs between acting as 2900 store and acting as array processors, according to the host system load and the demands on the DAP resource.

When Director is called to initiate a DAP start, it passes to Supervisor the required initial values of various DAP image store registers. Supervisor then activates the execution of the DAP program by writing to the DAP image store register CTRL. Supervisor also sets a value in the IT register, which times out roughly every 8 seconds; the DAP then interrupts the processor, and Supervisor resets the timer.

Whenever the DAP stops, it should interrupt Supervisor. A series of image store registers are then read by Supervisor, which passes them back to Director, which in turn passes them back to the DAP loader at user program level. The user process may then continue.

Sometimes, for unknown reasons (usually following a hardware failure), the DAP may be started, but it promptly stops without ever interrupting Supervisor. To combat the subsequent hang up, Supervisor now includes code to deal with a DAP timing out in this fashion.

Extra complication was inevitable with the acquisition and installation of a second DAP in late 1983. Supervisor introduced the concept of 'logical DAPs' and now does all its handling of multiple DAPs on the basis of logical DAPs rather than physical DAPs. Supervisor Logical DAP 1 is the first DAP configured in as a DAP—it can be either of the physical DAPs. If both DAPs are in configuration it is immaterial on which DAP a particular program will run—indeed if there are a number of processes each running DAP programs simultaneously, a particular program may run on different DAPs in consecutive time-slices. Despite there being two physical DAPs, and two batch DAP services (see the section on 'Spooler'), Supervisor maintains a single DAP service queue.

DAP PROGRAM LOADER

The DAP program loader consists of routines called from the DAPRUN command and it operates at the user, rather than a privileged, level. Its role is to claim the DAP if available, to load all the required areas of the DAP program block from the DAP program object file into the virtual DAP maintained by Director, to call the EMAS loader to load up and enter the user's host program object file and to control entry to the DAP when a DAP subroutine is called from the host program. In addition, the DAP loader handles host contingencies generated by, or on behalf of, the user process when a DAP program is running, and processes any DDE (DAP data expansion) transfer requests (see below) from DAP programs.

The DAP loader is in two distinct parts, the first concerned with the claiming of the DAP and the loading of the DAP program. This part is in a procedure called LOAD DAP which is called directly from the DAPRUN command. LOAD DAP initiates a DAP claim request to Director. If this is successful, Director returns to the loader the virtual base address of the virtual DAP. All addresses for the loading of sections of the DAP program block are calculated from offsets from this address.

With the DAP claim successful, the loader then proceeds to zero out the entire area claimed. The DAP program as produced by the consolidator is in a number of sections, such as for code, read only data, read/write data, control, workspace and stack. The internal addresses in the code are fixed up by the consolidator and are relative from the values to be set up in the code base and limit registers. The consolidator produces, within the object file, a table with entries for each area and their offsets in the program block that will reside in the DAP.

Each of the DAP program areas is then moved into the appropriate area in turn, according to the relevant displacement and length entries in the load table. Once this process is complete, all that remains is for the host program to be loaded and executed.

On EMAS, program loading is simply a matter of satisfying external references and making necessary address-word relocations. The DAP program is explicitly loaded by the subsystem loader in order that its external entry points (the DAP program ENTRY SUBROUTINES) can be located and entered by the calling main program.

Finally, the subsystem loader is called to load and enter the host program.

The second part of the DAP loader is concerned with the actual entering and calling of a DAP subroutine from the host program.

When the host program calls on a DAP subroutine, a 32 byte instruction sequence is entered and executed. All this sequence does is to place a number at a known offset from the 2900 CTB (common table base) register, the number being different for each subroutine in a DAP object file. A call is then made to a short procedure called ICL9PAJDAPSTART. On EMAS this procedure decodes the stored number and uses it to calculate an offset within the DAP object file which contains the entry point address for the appropriate subroutine. A call is then made to the DAP loader procedure RUNDAP with this entry point as parameter.

RUNDAP has two main roles: first it initiates a DAP start request to Director; secondly on termination of the DAP subroutine it decodes and analyses the stop code returned by the DAP via Supervisor and Director, and executes the appropriate action. There are basically three types of DAP stop to be considered, with different actions to be followed in each case.

1. Successful stop. Return is made to the calling program.
2. TRACE stop. Diagnostics are called to process the monitor TRACE call. The DAP is then restarted at the point of stopping.
3. Diagnostic stop. Some kind of error has been deteced, so diagnostics are called followed by a return to command level.

A fourth possible type of stop, that in the case of a DDE transfer call (see below) is directed to a procedure that deals solely with initiating DDE transfers.

USER VIEW

It was a deliberate policy that the user interface to DAP commands would follow EMAS philosophies rather than those of VME to allow for consistency and ease of use by established EMAS users. On EMAS it is not necessary for the user to assemble separately the AMF produced by a call of the DAP FORTRAN compiler—the EMAS DAPFORTRAN command does this automatically if the compilation phase was successful. There is a DAPASSEMBLE command for the assembling of user-written APAL, although users are not encouraged to use it.

The compiler and assembler are called with two obligatory parameters, for the source filename and the output filename. In addition an optional listing file may be specified. Control of the various ICL-designated options may be effected by special DAP PARM and DAP OPTION commands. This method was chosen in preference to the ICL type parameter keywords as the standard EMAS compilers control their options similarly.

The interface to the consolidator has been altered to look like that of the EMAS program linker, again for continuity.

A special DAPRUN command was introduced, which takes the names of the host program object file and the DAP program object file as parameters. A third parameter is used to specify the maximum DAP time for the program run in minutes.

Various utility commands were supplied, such as the DAP PARM and DAP OPTION commands mentioned before, and also a DAP STATE command which returns the complete physical and access status of the DAPs in configuration.

Host programs are written in FORTRAN 77 and compiled using the ERCC-produced FORTRAN 77 compiler. The complete ICL DAP subroutine library is available as well as an extensive subroutine library obtained from Queen Mary College (QMC). In addition, users have access to on-line documentation.

The DAPs are available to both interactive and batch users, with the bulk of the available DAP time being used by processes running in batch mode. The ability to access the DAPs interactively is useful for the testing of short programs before running a full program in batch mode.

DAP DATA EXPANSION—DDE

The ERCC DAPs each have a DAP store size of 2 megabytes. This size limits the amount of data that can be processed, as the entire DAP program (input data, output data and control information) has to fit into the DAP store. It was not financially possible to extend the size of the ERCC machines to 8 megabytes, the solution adopted at QMC, but it was desirable to increase the amounts of data which could be processed without the overhead of multiple subroutine calls. ICL produced a block transfer system (BTS) for use with the DAP under VME which allows asynchronous background transfers of data to maximize throughput, but on investigation BTS was found to be too VME specific to be imported into EMAS because of its many low-level dependencies.

After consultation with the Physics Department users, it was decided that initially a data replacement strategy could be devised, whereby data would be removed from the DAP store after it was processed and fresh data brought in to replace it without the DAP being released and completely unloaded and reloaded each time. The system, called DAP data extension (DDE), was designed so that whereas initially only synchronous transfers would be allowed, with the DAP being suspended but not released while the transfers were taking place, it would be comparatively easy to extend the system later to allow asynchronous transfers to take place while the DAP was processing other data.

Changes were required at the Director level and in the DAP loader, and APAL subroutines had to be supplied to intercept the transfer request calls from the DAP subroutine. Data to the DAP is passed from and to the host via FORTRAN COMMON blocks. Under conventional operation there is a one-to-one relationship between COMMON blocks in the host (containing DAP data) and COMMON blocks in the DAP. DDE breaks this relationship, and allows a large number (currently 62) of host COMMON blocks to be associated with a single DAP COMMON block. The only restriction on the COMMON blocks is that they must be multiples of an EMAS page (4096 bytes) in size. When a FORTRAN program is run on EMAS, COMMON areas are assigned space in the non-shareable GLA (general linkage area) by the loader. The DDE host COMMONs could not easily be assigned such space, as the starts of separate GLA areas are not generally page aligned and the transfer mechanism can deal only with exact numbers of pages from virtual memory page boundaries. Instead, the DDE COMMONs were separately assigned space on page boundaries in a special unshared area used only by a DAP program running under DDE. This facility was made possible by recent developments to the EMAS subsystem loader which permitted flexible use and assignment of program data areas. Each host DDE COMMON area is assigned a name, like a conventional COMMON, and a reference number to identify it to the DDE transfer system. Likewise, the DAP DDE COMMON areas are assigned reference numbers. The assignment of reference numbers is done in the host program, the transfer requesting in the DAP program being done on the basis of these reference numbers.

Additional statements are required in programs using DDE to declare the special DDE COMMONs and to initiate data transfers.

Instructions have to be given to DDE to allocate the host and DAP COMMONs; this is done in the host program. Instructions to initiate the actual transfers are given in the DAP program.

After an initial trial period with the simple synchronous DDE, an improved asynchronous version, allowing transfers to be carried out while DAP processing is continuing, was introduced into service. Return is made to the DAP program so that it can continue processing until it reaches a DDE AWAIT call. If the transfers are completed by that time processing will continue without any pause.

Usually a DDE transfer is a bidirectional movement of data, from DAP to host (processed data) and from host to DAP (new input data). Facilities are available for a transfer to be unidirectional, either to or from the DAP. If total two way data replacement is not required, then unidirectional transfers can be achieved in half the total transfer time.

The DDE transfer initiation procedure DDE TRANSFER is written in APAL, and is consolidated into the DAP object file. It generates a special stop code according to the reference numbers passed as parameters, and on receipt of such a code Director makes a call on a DDE control routine within the DAP loader, rather than making a conventional return. This control routine decodes from the stop code the identity of the host and DAP COMMON areas involved, works out the virtual memory code addresses for the source and destination of each area to be transferred and makes a call on Director to initiate the transfer.

DDE has been in service since October 1983 and has been extensively used by members of the Physics Department to increase the size of the models they are working on.

SPOOLER

It was envisaged that the bulk of DAP usage would be processes running as batch jobs, and so it was necessary to extend the batch job scheduling and selection methods to cope with jobs that needed access to the DAP.

DAP batch jobs are submitted to EMAS in exactly the same way as other batch jobs but with the addition of one extra scheduling parameter, namely the 'DAP requirement' (DAP minutes).

Batch jobs are scheduled and initiated on EMAS by a paged system executive process, Spooler, which maintains the relevant queues and has a pool of job control streams that can be dynamically adjusted to initiate jobs of differing requirements (job priority, job OCP time etc.). One of these streams is adjusted to service only jobs requiring the DAP.

When a DAP batch job is submitted to EMAS it is added to the relevant queue and the operators are informed of the job's DAP requirement. In normal circumstances this is for information only. A DAP job is then scheduled and when at the head of the queue, subject to the availability of the DAP and the job's satisfaction of the current DAP control parameters, is started automatically. Spooler checks first on the availability of the DAP, and then on the current batch DAP time limits (controlled by Director) before selecting a job from the queue and requesting Director to start the process. For abnormal circumstances there is an operator manual override to this automatic sequence.

Further complication was introduced with the arrival of the second DAP. Spooler now maintains two job streams solely for DAP use, but these are activated only when both DAPs are in service. The two job streams, or 'services', do not in any way reflect the physical allocation of DAP to user job. It is Supervisor which allocates logical DAPs (which are mapped directly to physical DAPs) to an active process. Different DAP time limits are imposed on the two services, so that currently when both DAPs are operating one stream will handle short jobs, and the other longer jobs.

DAP IN SERVICE

The first ERCC DAP was delivered in March 1982, and physically installed in the 2972 machine room on 5 April. By 12 April, all the physical connections were made, although there were some problems in interfacing the DAP to the dual 2972 processors. It is believed that a DAP had not been previously connected to a dual 2900 configuration, and further complication arose as EMAS treats a dual processor as a symmetric configuration so that the DAP had to be capable of interrupting either processor. It was some time after that before the DAP was first available to the configuration, initially as store only. During May it was made possible to reconfigure from store to DAP, but the initial test versions of the EMAS DAP software were not available until June. Testing and development continued throughout the summer, aided by the flexibility of the dual 2972 installation whereby the system can be partitioned into a reduced service configuration with one processor and half the main store, and a parallel development configuration with the other processor, the other half of the main store and the DAP.

The DAP entered full user service only one day later than the target set in the spring, on 12 October 1982. Initially, there was no multi-programming or time-slicing of the DAP, no dynamic reconfiguration between DAP and store and diagnostics were not available for the DAP FORTRAN compiler until late October. The Supervisor and Director modifications to support time-slicing and dynamic reconfiguration were tested during December 1982, and entered service in January 1983.

The DAP loader and Director modifications required for the DDE implementation were designed and implemented over the summer of 1983, and Supervisor, Director and Spooler modifications for the incorporation of the second DAP were completed during the same period.

The second DAP was delivered and installed during September 1983, and testing started at the end of November. There were the inevitable problems in the interconnection of a second DAP into the configuration, and the DAP had to be given special on-site modifications to enable it to interrupt the processors on Port 5 rather than Port 4. Although in trial service for some of December 1983, it was not until January 1984 that the second DAP was commissioned and fully accepted into the configuration.

When there was only one DAP, the policy was to have the machine configured as store for much of the day shift on most weekdays during term time. At all other times, the machine was available for a DAP, though running as store if there was no demand for use as a DAP. With the introduction of the second DAP it is hoped to have one in service as before, with the other in use as a DAP whenever the system is available.

DAP RUNNING FIGURES

Table I gives the total DAP running times for the year 1983. It must be noted that the figures for December include a proportion of running with dual DAPs. Separate figures are not maintained for each machine, as programs may time-slice between DAPs under certain conditions.

Table I

Year	Month	Total DAP time Hours.Mins. Secs	Users	Program runs	Sub-routine calls
1983	January	317.51.11	12	1254	23,204
	February	254.31.53	13	1087	3769
	March	391.41.48	14	1097	3539
	April	460.26.25	21	1209	2309
	May	476.25.52	18	1547	3363
	June	516.15.04	21	1537	5132
	July	510.46.45	22	1502	4901
	August	361.30.42	19	1263	9464
	September	421.21.25	18	1251	6634
	October	467.40.10	20	1317	11,118
	November	257.37.07	23	2024	7955
	December	334.56.02	22	1443	16,005
	Total	4771.04.24		16,531	97,393

CONCLUSIONS

This paper described the successful integration of two DAPs into the ERCC EMAS configuration. The incorporation of the hardware, and the writing of the new support and driver software was completed in a comparatively short period with surprisingly few problems. Even the importation of alien software which existed only as VME object files was completed with comparative ease.

The usage figures given for 1983 show that very large amounts of DAP time have been used, a trend which is continuing with extensive use of the machines overnight and at weekends as well as during the day. On many days in 1983 the first DAP was in use for almost 100 per cent of the theoretical maximum time. The future of DAP use at Edinburgh looks encouraging, with current and projected work giving maximum use in the months ahead.

ACKNOWLEDGEMENTS

Many people have been involved in making the Edinburgh DAPs a success, although special acknowledgement must go to Peter Stephens for the design and implementation of the alterations to the EMAS Supervisor, to Keith Yarwood for the design and implementation of the enhancements to Director and to John Henshall for the enhancements to Spooler. Also to Professor D. J. Wallace of the Physics Department, and the late J.G. Burns of ERCC who were instrumental in the acquisition of the DAPs for ERCC.

Finally, thanks are due to Geoff Millard, the Compiler Group Manager, who designed and implemented the ERCC FORTRAN 77 compiler and who released the author for eighteen months to enable this project to come to fruition.

REFERENCES

1. R. W. Gostick, 'Software and algorithms for the distributed array processor', *ICL Technical Journal*, **1**, (2), (1979).
2. H. Whitfield and A. S. Wight, 'The Edinburgh Multi-Access System', *The Computer Journal*, **16**, 331–346 (1973).
3. P. D. Stephens, J. K. Yarwood, D. J. Rees and N. H. Shelness, 'The evolution of the operating system EMAS 2900', *Software—Practice and Experience*, **10**, 993–1008 (1980).
4. P. D. Stephens, 'The IMP Language and Compiler', *The Computer Journal*, **17**, 216–223 (1974).
5. F. Stephens and J. Murison, *The IMP80 Language*, ERCC reference manual, 1982.
6. K. C. Bowler and G. S. Pawley, 'Molecular dynamics and Monte Carlo simulations in solid-state and elementary particle physics', *Proc. IEEE*, **72**, (1984).
7. International Computers Limited, *DAP: FORTRAN Language*, ICL TP 6918, 3rd Edition, April 1981.
8. International Computers Limited, *DAP: APAL Language*, ICL TP 6919, 1st Edition, July 1979.
9. P. D. Stephens and J. K. Yarwood, 'Providing multi-user access to distributed Array processors', *Software—Practice and Experience*, **16**, 531–539 (1986).