DAD, The C.S.I.R.O. Operating System

B. J. AUSTIN, T. S. HOLDEN AND R. H. HUDSON
C.S.I.R.O. Computing Research Section, Canberra, Australia

The design and implementation of the C.S.I.R.O. operating system, DAD, is described in detail. This system is designed for the Control Data 3600 using a large drum backing store and is intended to allow the integration of a remote console (display) subsystem into a conventional job stack environment.

The use of the drums, the buffering of input and output on slow peripherals, and the execution of normal job stack work are described. The display subsystem is described only as it integrates into the rest of the system. The techniques found useful in the development of DAD are given, and an assessment is made of the validity of various design decisions.

Performance figures based on several months of operation are tabulated.

1. Introduction

This paper is intended to give details of the design and performance of the C.S.I.R.O. operating system, DAD. A preliminary description was given by Austin and Holden [2] in a paper written before the design had been finalized and tested in practice.[1] (See also Pearcey and Kerr [1], and Kerr and Karoly [3].) The system has now reached some stability, although further refinements and facilities are currently being added. The object here is to give a full description of the design, and some performance figures based on several months of operation.

The writing of the C.S.I.R.O. Drum and Display (DAD) Monitor was undertaken in an effort to use to best advantage the equipment at our disposal. This comprises a Control Data 3600 computer with a 32-thousand-word, 48-bit core store, eight magnetic tape drives, three line printers, a card reader, a card punch, two incremental plotters and two reader/punch paper tape stations, together with one million words of high speed drum backing store, six small remote console keyboard displays, and a 19-inch CRT display. The equipment, other than the last three types listed, has been operating since August 1964, but the drums and displays were not delivered until March 1966. Work on the DAD system commenced in December 1964.

The last three I/O devices mentioned are grouped together to emphasize their importance in the DAD system. DAD uses the drums as a large I/O well, and the method by which this has been implemented has provided an environment for the effective use of the displays. This paper does not set out to describe in detail the display subsystem (DAVE), but rather deals with I/O processing, job stack handling, and the control and usage of peripherals.

Prior to the introduction of DAD, the CDC-supplied monitor, SCOPE, was used. Users of the system, both in Canberra and in other parts of the C.S.I.R.O. network, had been developing programs under SCOPE since August 1964, and therefore a high degree of compatibility had to be achieved between SCOPE and DAD from the users' point of view. Furthermore, as the manpower resources were comparatively meagre, it was imperative that only the minimum rewriting of the system be attempted, and therefore the smallest possible modifications have been made to the compilers, assemblers, and other large complexes so that they would work under the DAD system. Thus, within the framework of compatibility with SCOPE, the aim was to design a monitor with an efficient I/O buffering scheme into which a display subsystem could be successfully integrated. The aim was also to produce a system which would ease the burden on the operators. This has been achieved by incorporating into the monitor the scheduling of job executions. Furthermore, the marshalling of magnetic tapes and other data required by programs, and the handling of equipment failures, are such that the operators can attend to their various tasks as convenient, without halting the system as a whole.

2. Allocation of Core Storage

The core store is allocated to various functions of the monitor in four blocks, viz.:

(a) Resident, which handles all interrupt processing and requests from users at execution time, and also I/O buffering.

(b) Display area, which is a small area used by all display programs on an overlay basis and by the display monitor DAVE.

[1] DAD has been the standard system at the C.S.I.R.O. installation since June 1966, and has recently (February 1967) become the standard system of the Computer Center at Santa Barbara, California.

(c) Store used by the current job stack program.

(d) The rest of the store, which is used for buffering information to or from the drums.

Note that only one job from the job stack is in the store at a time. It was desirable that the maximum store space available to this job (the "main job") be not less than the available space left by SCOPE, in order that compatibility be achieved. This consideration has forced us to write the resident part of DAD with an accent on brevity. We have in fact failed to meet this target by about 300 words. DAD resident occupies 4500 words; DAD plus the display area (b), 7300 words. It will be seen that there is no attempt to run more than one main job at a time. We feel that the 3600 does not have sufficient hardware facilities, in that the bounds protection is not complete since it can be by-passed by input from peripherals, and there is no built-in relocation feature.

3. Use of the Drums

The heart of the DAD system lies in its use of the drums. The drum store is used up in segments of 256 words at a time, and the buffer area (d) is similarly subdivided. The buffer area can be thought of as an extension of the drums with the property that immediate access is possible. The drum segments not in use are shown in the Free Space List, which is a bit pattern held in core.

Under SCOPE, a programmer referenced all input or output via "logical units," which were basically magnetic tapes but could be other devices which produced or received data records in a serial manner. In the DAD monitor, this concept has been taken over and generalized by providing a means for storing and accessing serial data strings on the drum. These data strings are called "documents." A document is stored as a linked series of 256-word segments, which are allocated dynamically as the document is created. The linking structure contains both forward-pointing and backward-pointing addresses, so that movements through the document in either direction are possible. The programmer is provided with the facility for accessing more than 50 documents in one program and also means whereby he can name a document, save it on the drum, and retrieve it later. This last feature makes an effective display system feasible.

The programmer may view DAD documents as magnetic tapes and need have no knowledge of the fact that the serial access properties are simulated by the system. Facilities are provided to write records and to read records (forwards or reverse) with the same ability as the 3600 hardware to write record markers, skip information, and chain from one control word to another. Further facilities are provided which allow backspacing, skipping forward or backward to a file mark, and rewinding to the beginning of the document. A high degree of compatibility has been achieved between the facilities provided in magnetic tape hardware and those simulated by DAD on the drum, and it is possible to develop a program on small quantities of data using a drum logical unit, and to change to magnetic tape for production purposes, merely by changing one control card. In addition, the linked structure of a document makes it easy to add facilities not possible with magnetic tapes, viz., the insertion and deletion of records. These properties of drum documents do not appear to be utilized by many of the users, but they have been most effectively employed in many parts of the system itself.

The chief system documents are the Main Documents List (MDL), the Output List (OL), and the Execution List (EL). In fact each of these consists of more than one document. A detailed description will be given in later sections of the paper, but an outline of the functions of MDL, OL, and EL is appropriate here. MDL contains the directory of all documents currently on the drum. OL lists all documents awaiting output, and the peripheral type required. EL lists all documents awaiting execution, and gives the maximum CPU time to be allowed for each.

The part of the DAD system which simulates hardware properties for drum documents is re-entrant; that is, it can deal with many active documents simultaneously. This adds quite considerably to the programming required but is well worthwhile. The maximum number of documents that can be handled at any time is limited only by the size of various tables and has been set at ten. This number appears to be sufficient.

Documents may be given names and may be saved on the drum and retrieved later by referring to the name. The list of all documents currently on the drum is kept in the Main Documents List (MDL), which is itself a document. Each entry gives a document name and the drum address at which its first segment starts.

Programmers using the C.S.I.R.O. network are identified for accounting purposes by an eight-letter charge code. Document names contain the charge code of the programmer who "owns" them. There is a limited check placed upon one programmer attempting to use the documents of another—he may read them, but not alter or destroy them.

The exact layout of the linking structure is worth mentioning. It has already been stated that drum space is allocated in 256-word segments, and that each segment contains a forward and a backward link. There is also a field giving the number of words in the segment which are actually used. This is only different from 256 for the last segment of a document or where insertion or deletion operations have occurred. The logical records are also given by a linking structure. Successive records are separated by a word which gives their lengths, so that track can be kept of the record structure whether the document is being scanned forwards or backwards. This extra word also gives certain other information which is used to simulate hardware properties such as even/odd parity. Records are held without regard to the boundaries of the 256-word segments and just spill over to the next segment as required. Thus, all segments comprising a document are fully utilized except when insertion or deletion operations have been carried out.

The programming required to achieve the above is complicated, but was to a large extent tested out by simulation long before the drums were delivered. Some bugs remained, however, and showed up only when considerable drum activity occurred.

A linking system such as this has two characteristics when things go wrong: (a) the conditions which cause the error are ill-defined and hard to reproduce, and (b) the system carries on past the occurrence of the error for a considerable distance, stopping only when the links point to a nonexistent drum address. It seems, therefore, that the design of the system should include some means of recognizing a wrong link word at once. For instance, any unused bits in the word could be adjusted to make the word an exact multiple of, say, 13 and the system could stop at once on failure of a link word to pass this test. This would make debugging much simpler and also pinpoint hardware failures more rapidly.

The allocation of segments to a programmer is not under his control but is "optimized" for him. To be precise, an attempt is made to choose segments so that forward movement through the document is made as fast as possible. Switching tracks takes no appreciable time on the drums, and so all segments with the same angular position are equally accessible. However, no use is made of this fact. The first attempt is to link a segment to one which is 5msec ahead around the drum, this being the minimum spacing for which the reading of both segments with a single drum revolution can be guaranteed. If this particular segment is not available, any available segment is chosen. In practice, it turns out that this simple algorithm optimizes successfully for 10–30 segments before breaking down.

The servicing of drum read/write requests is also optimized. The drum hardware includes a facility for ascertaining the angular position of the drums, which are not locked in phase.

At first, an algorithm was tried which favored system or display program drum requests, as an attempt to improve efficiency, especially in the response time of the displays. This was found not to be of much use in the latter respect and, furthermore, introduced some logical difficulties when some requests had to be serviced in the order they were made. The present optimizing algorithm is thus quite simple—the request from the queue which can be serviced first is taken. In sampling the angular position of each drum a small constant is added to make sure that the best request can be selected and started before its crisis time occurs.

All operations on documents require that the information on the drum be brought into the store. The buffer area (d) mentioned above is used in units of 256 words. A table is set up giving the drum address corresponding to each buffer and showing whether the information in the store is an exact copy of what is on the drum, i.e., does not need to be written back. The number of buffers available to the system depends on the program in core and whether the display system is active. A buffer is written onto the drum in general only when the buffer is required for another drum segment, and this means that a whole series of requests from the programmer may be honored without the need to wait for a drum transfer. A maximum of 32 buffers is used, but it seems that for most purposes 16 is more than sufficient and 5 is satisfactory.

4. Transfer Rates for Drum Documents

Table I gives the transfer rates of one drum logical unit for writing and forward reading as a function of record size. Figures for the magnetic tapes (Control Data Model 607) are given for comparison. The transfer rates are given in thousands of words per second.

Table II gives the transfer rates when there are 10 logical units reading or writing simultaneously, as a function of the number of buffers available. This rather extreme test is noticeably affected by any other activity which happens to be occurring in the system, and it was difficult to obtain consistent results. Hence the anomalous values around 16–18. The transfer rates are given in thousands of words per second.

5. Use of the Drums as Random Access Devices

The original design of DAD did not provide for the use of the drums in any way except via documents. It has since been decided to incorporate a random access facility

TABLE I. Transfer Rates with One Drum Logical Unit

(*in thousands of words per second*)

Record size	Drum documents		Magnetic tapes	
	Write	Read	800 bpi	556 bpi
10	2.5	2.5	1.7	1.7
50	11.5	9.8	2.4	2.4
100	13.8	10.8	6.8	5.6
200	15.2	12.2	9.4	7.4
500	16.2	12.8	12.0	8.5

TABLE II. Transfer Rates with Ten Logical Units

(*in thousands of words per second*)

Number of buffers	Write	Read
2	0.2	0.3
4	0.3	0.5
6	0.3	0.5
8	0.4	0.7
10	1.0	1.3
12	1.3	1.3
14	1.7	1.4
16	1.7	2.0
18	2.5	1.4
20	2.5	2.5

into the system. This allows a programmer to request a number of consecutive segments which can then be addressed directly. The random access feature is made subordinate to the document (serial access) use of the drum and the programmer's request is rejected if the required number of consecutive segments cannot be located in the Free Space List. No attempt is made to reorder the linkage of existing documents in order to clear a suitable area. However, a random access logical unit of 32 thousand words is always provided. This can be used by the programmer, and is also used by the system during compiling/loading and during recovery after abnormal program termination.

It appears that not many of the users of the system employ the random access facility, but it is used heavily by the system itself. In addition to the above cases, the nonresident system programs of DAD (DAVE, INTERJOB—see Section 8) and compilers (FORTRAN, COMPASS) in absolute form, and the library of common subroutines in relocatable form, are held as random access records on the drum. The display subsystem DAVE, makes use of the feature to provide a rapid overlay scheme for display programs.

A random access facility is thus too valuable to discard, and it should be incorporated into a document-oriented system such as DAD. A complete integration of random access and a dynamic allocation scheme is difficult to achieve, but fixed areas used by the system can be allocated at start-up time quite easily. The gain in efficiency which results from making frequently used system programs into random access records is very great.

6. I/O Buffering Program

The I/O buffering program is permanently in core, and carries out three main functions. First, the label on any reel of magnetic tape is read as soon as it is mounted, so that the system is always up-to-date in its knowledge of the tapes available to it. Secondly, all input peripherals are serviced, and data read from them is formed into documents on the drum. Finally, documents flagged for output are sent to the appropriate peripheral. The first function is relatively trivial, and will not be discussed further. The input and output functions are more important. An understanding of these aspects requires some knowledge of the 3600 hardware.

The C.S.I.R.O. installation has a large number and diversity of external devices connected directly to the computer. There is no satellite machine, and communication between computer and peripherals is achieved by means of data channels, of which there are five. A data channel can be directed by the central processing unit to initiate a transfer of information, and it will complete the transfer in an autonomous fashion while the central processor does further I/O or computational work. The end of an I/O operation is signalled to the central processor by means of an interrupt. It is possible to have this interrupt occur at a variety of phases of the I/O operation, and also on the occurrence of certain types of error condition.

The channels work with 48-bit words as far as the computer is concerned, and transfer 12-bit bytes to or from the peripheral equipment. After the last 12-bit byte has been transmitted, the channel becomes not busy (i.e., free to service a further I/O request for the same or another device). The external equipment itself, however, remains "busy" and is therefore unable to undertake a further operation for an additional period. The time that the equipment remains busy is usually of the order of many milliseconds. This distinction between channel-busy and equipment-busy is fundamental to the design of the DAD system.

It is also important to appreciate the difference between those devices that have a buffer and those that do not. The card reader has a buffer which holds a complete card image, and although it takes 50msec to read a card the buffer can be emptied by a data channel in about 1msec. Thus the card reader occupies virtually no channel time, even when operating at full speed. Similar remarks apply to the printers and card punch, which are also buffered devices. By contrast, the unbuffered equipments, which are the magnetic tape units, the typewriter, the incremental plotters, the paper tape stations, the displays and the drums each occupy a channel practically full-time. There is an option on some devices which allows the transfer of characters rather than words, and this would in principle make it possible to use these devices in a manner which would occupy almost no channel time. In practice, however, the amount of interrupt processing and general dead time in the monitor that this would involve make the scheme quite unworkable in that all the central processor capacity would be utilized to service interrupts.

In attempting to maximize the throughput of the peripherals, we are immediately concerned with the best use of the data channels. There are two aspects to the problem—firstly, how to allocate all the peripherals between the channels, and secondly, how to service I/O requests. One channel has been kept for the buffered devices (card reader, printers, and card punch), so that these important peripherals will never be held up by unavailability of a channel. Furthermore, the transfer rate of the drums is so high (two million characters per second) that a specially built fast data channel is exclusively reserved for them. The remaining three channels are used by the alternative connections to the buffered devices and by other I/O devices. It has been found effective to have the plotters on different channels.

Requests to the monitor for I/O operations are placed in a table which has one entry per I/O device. Besides the specification of the operation required, each table entry contains bits indicating what channels are physically wired to these devices. When a channel is free, a scan is made for requests which can be honored by that channel. This involves locating a request, checking the status of the device and, if all is well, initiating the operation. "Busy"

status causes the monitor to try the next request in the table. Error conditions such as "not ready" or "paper out" similarly cause the monitor to try another request, and a message is given to the operators at 10 minute intervals on the console typewriter. *In general, failure in one device does not cause the system as a whole to stop.* (An exception is the console typewriter—failure there causes complete stoppage with an indication in the console lights.) This tends to ease the burden on the operators.

Requests for operation on some of the unbuffered devices (paper tape reader and punch, plotters, and typewriter) are "partitioned," i.e., cut up into shorter transfers. This reduces the channel busy time and also allows other devices an occasional quick opportunity for service by the channel. A series of one word transfers to a character device such as a paper tape punch occupies the channel for $\frac{7}{8}$ of the time. A series of two word transfers uses $\frac{15}{16}$ and so on. Hence the cutting up of long transfers into shorter ones provides a small but worthwhile opportunity for other use of the channel. If a scan down the request table fails to locate an operation which can be started, the channel is left unused. In principle, in the 3600 hardware there is a means whereby an interrupt can indicate to the monitor that a peripheral device has become available for a further operation—the interrupt on "Ready and not Busy." In practice, it has been found impossible to use this facility, chiefly because of the difficulty of avoiding the interrupt when it is not wanted, i.e., during data transfers on another device on the same channel. Instead the system relies on repeating the whole scan at frequent intervals, in fact at every entry to the monitor. A time interrupt every 100msec is used to cause an entry into the monitor and a scan of the request table. This means that the printers and card reader will run at not less than 600 records per minute, which is not full speed but is satisfactory. If the main program is causing entries to the monitor in addition, the printers and card reader approach full speed. This scheme is not very elegant, and it is also inefficient. A scan of the request table can take 2–3msec if many requests are pending. When a great deal of peripheral activity is going on, many interrupts occur every second, and hence there are many request table scans. In this way, more than 50 percent of the central processing unit's capacity can be used in servicing interrupts. A means of limiting the minimum time between scans to 8msec has been added, and this has given an adequate throughput potential with reasonable overheads (see below for details).

The I/O buffering program (BACKGROUND) is written almost completely in a reentrant fashion. Each branch of the program controls one peripheral device. Control passes to BACKGROUND in the first place by means of a time interrupt occurring every two seconds. The similarity to the technique employed on ATLAS [5] will be noted. At this time every peripheral not already in use is examined. If a peripheral is found to be in a "not ready" condition, it is ignored, but if "ready" is signalled BACKGROUND attempts to use the device. ("Ready" means that no fault condition exists and that the operator has pressed the "Ready" button.)

If the device is an input peripheral, "ready" indicates that the operator has loaded some data (cards or paper tape) for input. The first record is treated as a control statement or "header" giving the document name which is to be attached to the data following, and specifying what translation process is required to turn the data into internal BCD code. Punched cards can be read as 20-word "binary" cards (i.e., untranslated) or 10-word "BCD" cards (translated from Hollerith code by the card reader hardware). Paper tape can be read in six modes, which include parity checked/unchecked, 1 or 6 or 8 characters per machine word, and translation from the code used by our paper tape preparation equipment. The last option and packing six characters per word are performed by software, while all the other facilities are provided in the paper tape hardware.

After reading the header, input of the document is initiated. Since a paper tape document may have to be physically separate from its header (because it is a different width), reading stops before commencing the input of a paper tape document. The operator must indicate that the paper tape has been mounted by pushing "Not Ready" followed by "Ready" about two seconds later. Input then continues until a special control statement, "end of document," is read, or until some error condition indicates that the input should be abandoned.

When BACKGROUND discovers that an output device has become "Ready," it searches for a suitable document to output. Documents awaiting output are queued in two serial lists on the drums, (OL1 and OL2). Each list contains the names of documents awaiting output, and the type of device to which they are to be sent. OL1 is always scanned before OL2 and therefore is the Priority Output List. A document goes into OL2 if its length exceeds 50 256-word segments. This criterion is crude but works quite well. The output capacity of the system is greater than the computing power can cope with, and thus the above priority scheme is not critical. By the same token, it would have been more sensible to reverse the scanning procedure, i.e., to look for a suitable device only when a document for output was discovered. The system was designed on the assumption that the output peripherals would always be kept busy, and is somewhat wasteful in the situation that actually exists.

When the system discovers a suitable document in OL1 or OL2, output commences. Information within the document gives the external code into which the document must be translated, if this is relevant.

The system pauses (awaiting the operator's "Not Ready"–"Ready" signal) before plotter output, directing the operator via the console typewriter to set the pen position. It is also possible for the programmer to select 1-part, 2-part, 4-part, or special stationery on the line printers, and the system keeps track of the type of paper mounted on each printer. If a document requires reloading

of a printer with a new type of stationery, the system gives the required type on the console typewriter and pauses awaiting "Not Ready"–"Ready." During these pauses the system as a whole does not stop; input and output continue but no fresh output branches will be activated. Again, this eases the operator's task by allowing the required action to be performed more or less at leisure.

Output, once started, continues autonomously. Error conditions, such as paper running out on a printer, cause an operator message and stoppage of the branch concerned. The operators can terminate a runaway output operation by typing in a directive on the console typewriter. They also have an option which allows them to terminate an output operation saving the document on the drum. This allows, for example, rerunning of a plot where the pen has run dry. When all of a document has been output, the peripheral is returned to the pool of available devices In general, the document is removed from the drum, but it can be saved if the programmer wishes. Confusion between successive decks coming from the card punch is avoided by directing the operator to clear the punch at the end of a card output document, and pausing until he has indicated by "Not Ready"–"Ready" that he has done so.

It has already been stated that BACKGROUND is almost completely reentrant. It has not been possible to make it completely so, because various nonreentrant bottlenecks exist. For instance, two branches may almost simultaneously require to use the typewriter to output error indications or directives. In this case, the second branch must cease all activity, and becomes "held-up." The held-up branch attempts to restart at the next two-second interrupt. Another bottleneck which causes nonreentrant behavior exists in the system queues (e.g., OL1 and OL2), and the treatment of the loading of new stationery types on the printers has been written in a nonreentrant fashion for reasons subsequently found invalid. It seems that we have made a basic error in design philosophy in not trying to reduce nonreentrant code to a minimum. It was assumed that the type of paper on the printers would not be changed often. This is largely true, but if the operator happens to overlook the message to change paper, all output devices gradually become idle. A similar error has been made in the input half of BACKGROUND, when in some circumstances the system refuses to process any fresh documents, and in various situations in which display users cannot log into the system because system lists must temporarily be frozen. The above examples can be divided into two categories—nonreentrant behavior implicit in the design or in the hardware, and nonreentrant behavior introduced as a convenient way of dealing with circumstances that it is hoped will occur infrequently. Our experience indicates that the latter class should be avoided entirely, and the former cut out wherever possible.

7. BACKGROUND Overheads

BACKGROUND incurs overheads in three ways. First, there is the scan of peripherals awaiting "ready" status. We have found that this costs less than 1 percent in CPU overhead. Secondly, once an available peripheral has been found, a search is made in OL for documents awaiting output, and thirdly, when a document is found its output will commence. Table III shows the percentage overhead as a function of the number of simultaneous input or output operations occurring. The entry for zero I/O operations gives the cost of the OL scans for 6 available output devices. The percentage is calculated from $100(X - Y)/X$, where Y is the time required for a long instruction loop when no interrupts are occurring and X is the time required for the same loop running under DAD with various amounts of BACKGROUND activity.

8. Job Execution

Job execution is under the control of a part of the DAD monitor called INTERJOB. INTERJOB is nonresident and is brought from the drum as a single absolute record at the end of every program run. INTERJOB will form recovery dumps after abnormal run termination, produce accounting information at the end of a job, and unload magnetic tapes and delete drum documents as required. The important task of deciding which job to run next is then started.

The names of documents awaiting execution are held in two special lists, EL1 and EL2—drum documents accessible only to the monitor. Scanning is rather similar to the scanning of the Output Lists OL1 and OL2, in that EL1 is always looked at first. INTERJOB takes the first entry from EL1 and finds out if the document given exists and is not being operated on by some other part of the system (BACKGROUND or the displays). If so, the document itself is scanned to find out what magnetic tapes and drum documents must also be available, this information being contained on control cards which must be the first records of the document. In the case where some required item is missing, a message goes out to the operators directing them to fetch the missing tape or document, and the job is then abandoned, but left in the Execution List. INTERJOB then proceeds to try successive entries in EL1 and then in EL2 until a job is found which can be successfully started. Once more, this system was designed to help the operators.

Entries are placed in EL1 or EL2 by three means. Firstly, documents being input by BACKGROUND may have as their "header" record a special control statement indicating that the document name is to go into the Execution List and giving the expected running time in

TABLE III

Number of I/O operations	BACKGROUND overhead (%)
0	3
1	8
2	12
3	20
4	25
7	50

minutes. Most documents handled by BACKGROUND, especially card reader documents, are of this type.

Secondly, display users may type in an "EXECUTE" request, giving the name of a document already on the drum and an estimate of the running time. Thirdly, jobs arrive in Canberra from the subsidiary computer centers at Sydney, Melbourne, and Adelaide, as magnetic tape job stacks, and are unloaded and formed into drum documents by a special part of the system, which also puts entries into EL1 or EL2.

The decision whether to put a job in EL1 or EL2 is made purely on the basis of the estimated running time supplied by the programmer. If this is 4 minutes or less, a job goes to the end of EL1; otherwise, to the end of EL2. The effect of this crude system is to give program development and display usage a better turnaround, but it is fairly common to find that EL1 has an hour's work queued up. Because of this, a further level of priority is being developed ("BREAKIN"), in which a low-priority program can be suspended temporarily while a high-priority program is run. BREAKIN will be particularly useful in improving the response of the system to display users.

To illustrate the path of a simple job through the system, consider a FORTRAN job consisting of a card deck containing control cards, FORTRAN source language cards and data cards. As the card deck is input through the card reader, the background program interprets the header record, forms the rest of the deck into a drum document, and closes the document when the "end of document" control card is found. The information in the header is used to form the MDL entry and to enter a request for execution into EL1 or EL2. At some subsequent time, the request reaches the top of the queue, and INTERJOB will then start the job by interpreting the first record of the document. This will be the FORTRAN control card, and INTERJOB responds by calling in the FORTRAN compiler and passing control to it. The job will then proceed through the various stages of compilation, loading and running as under the previous monitor, SCOPE, except that scratch units (such as those used by the compiler) are drum documents rather than magnetic tapes, and output from the job is formed into another drum document. When the job terminates, all logical units used by the job are released by the monitor and an appropriate output request is entered into OL1 or OL2. After a further period (often almost immediately), the background program services the output request by unloading the document to a printer and, when finished, the document is released. Many variations to this simple sequence of events are possible. The output and input documents could have been saved for inspection and editing by means of the displays (see later), the data could have been a separate document input from paper tape or produced by a previous job and saved, or the job could have used magnetic tapes, and so on.

Besides the main system documents referred to earlier, INTERJOB uses four further logical units, COD, ACM, OCM, and OIL. COD is a drum document which contains all the charge codes currently valid, and is used to prevent a job from starting which has an unknown accounting code. COD can be updated by the operators at any time during the day. Accounting information is output on ACM, which is a 110 char/sec paper tape punch, online. Each job produces about 20 characters of output, giving the charge code, date, and the time used which is in fact the central processor time plus a surcharge for input and output. Operator messages are output on OCM, which is the console typewriter (15 char/sec), also online, and information (rather than directives) for the operators is printed on the Operator Information Log (OIL), which is a 150 line/min printer, online. These three devices all cause nonreentrant behavior to some extent, but the typewriter is the slowest, and the nature of its messages causes it to be the most serious from this point of view. It would appear that a system like DAD needs a fast device such as a cathode-ray tube display to convey messages to the operators.

9. Display Subsystem

Operation of the DAD system is basically controlled by means of the system documents MDL, OL1, OL2, EL1, and EL2. The display subsystem (DAVE) is able to access these lists, and a display user may make requests which result in changes to them. The "EXECUTE" request has already been discussed. In addition, a display user may make "PRINT", "PUNCH", and "PLOT" requests which put document names into OL1 or OL2 and eventually cause the required output to occur. The display user may also "DELETE" a document, which removes the document's entry from MDL and deletes the information from the drum. Finally, there is a most useful request,

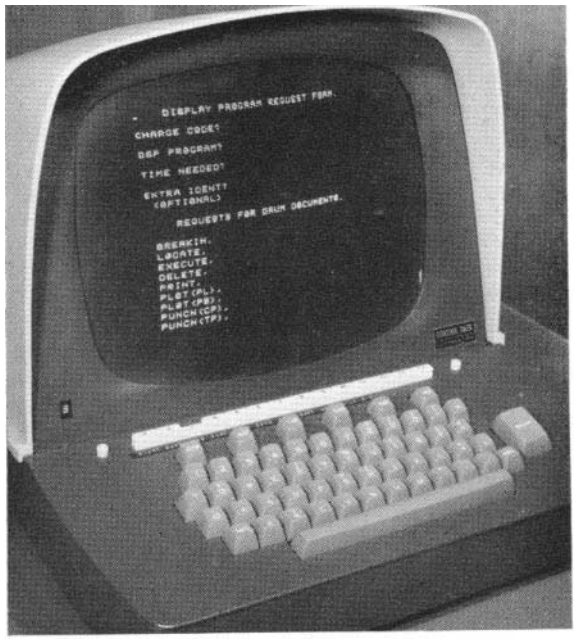

FIG. 1

"LOCATE", which ascertains whether the specified document is on the drum or being used by some other part of the system, and the estimated time until the document will commence execution if it appears in EL1 or EL2. These requests make it possible to observe and influence the operation of DAD from the remote consoles, and make the displays a well-integrated part of the whole system.

In addition there is provision for a display user to call in a display library program by name. The initial request form which appears on the display screen is illustrated in Figure 1. The user may respond by typing in his charge code, the name of a display program and the time required, in the upper portion of the screen, or by requesting an operation on a drum document in the lower portion of the screen. Display programs are special library programs that generally require only a few tenths of a second to process a console interrupt. They may be written in FORTRAN or in assembly language with the primary restriction that they require no more than 2000 words of core store for execution. A random access overlay scheme is available for lengthy programs such as the FORTRAN interpreter.

Display programs are run under the control of a small optionally resident display monitor (DAVE) which is responsible for swapping display programs and servicing their I/O requests (via DAD). A nonresident portion of DAVE initiates display programs and provides recovery dumps if they terminate abnormally. Display programs have priority over the main job but they, in turn, can be interrupted by the background program.

Currently there are twelve display library programs, ranging from simple demonstrations and engineering tests (e.g., NIM, CORESNAP, KEYTEST) to a general purpose statistical package (STATIST), a FORTRAN interpreter (INTERP) [4, 6], and a general purpose document editor (CIDER) [3, 6]. The latter program may be used in connection with the testing and debugging of main jobs. After the program under test has run, its output may be inspected and the input document modified by using CIDER. A new EXECUTE request is then made, and the job runs again. In this way a display user may have many test runs in the course of a single day without having to resubmit a card deck.

Typically a user will spend a half hour or so creating a document or interpreting a FORTRAN program, using of the order of ten seconds of central processor time. In one recent month, display programs accounted for 25 percent of the total number of jobs and, at the same time, used less than 1 percent of the accounted central processor time.

TABLE IV. JOB STATISTICS DURING CHANGEOVER FROM SCOPE TO DAD

Month of 1966	Charged time per day (hours)	Number of jobs per day	Average job time (minutes)
Jan.	12.50	236	3.18
Feb.	15.60	255	3.67
Mar.	14.80	258	3.44
Apr.	15.80	247	3.84
May	18.50	245	4.53
June	13.25	242	3.29
July	17.75	307	3.47
Aug.	15.75	336	2.81
Sept.	11.25	357	1.89
Oct.	15.70	312	3.02
Nov.	19.60	407	2.89
Dec.	16.80	395	2.55

10. Efficiency of System

It is difficult to accurately assess what has been gained by the changeover to the DAD system. The previous monitor was not intended to run a system with many slow devices online and was obviously inefficient when dealing with our peripherals. Hence, a comparison of computer usage statistics before and after the changeover does not give a fair estimate of the worth of DAD. Furthermore, the DAD priority scheme favors short jobs and seems to have influenced users to submit more jobs of less than 5 minutes duration.

Table IV gives the number of jobs run by the C.S.I.R.O. installation, and the useful time, for each month of 1966. Note that DAD became the official monitor at the end of June, but that an increasing amount of DAD runs occurred from March to June, and some SCOPE runs occurred after June, and also that the times given are the charged times, which from October include the surcharge for peripheral use. This surcharge appears on the average to increase the actual CPU time by a factor of about 1.5.

11. Development of the System

The DAD system was developed by a very small team of programmers. Three people worked on the project virtually full time for two years, and several other people were involved for short periods. The cost in programming effort is estimated as less than 10 man-years, the smallness of this figure is attributed to the approach to system development which was adopted.

Much of the development had to be done before the drums were delivered, and this forced us to adopt simulation techniques. We also elected to develop much of DAD as a normal job, running under the control of the existing monitor. Thus, for the price of a small simulation package which imitated machine features such as the interrupt system, the full debugging facilities of the existing monitor were obtained, eliminating a lot of push button work at the computer console. When the time came to take over full control of the computer and to operate the peripherals directly, a cyclic "log" was maintained in core of all the peripheral commands with other relevant information such as the computer clock and the equipment status. At this stage, a development run normally consisted of a few minutes of testing of peripheral facilities, followed by a core dump. The log could then be studied at leisure, and it enabled us to locate program bugs very quickly and also to

pinpoint hardware faults and unsuspected design peculiarities. The log was especially valuable when odd effects showed up only intermittently. When a hardware failure occurred, it was often possible to extract a sequence of peripheral instructions directly from the log, and turn them into a simple test program good enough for the engineers to locate the trouble.

As has been mentioned, the development work was well advanced before the drums arrived. A magnetic tape was used to simulate a 32K drum, and by the time the drums were delivered, DAD was a running, if primitive, system. The code which actually carried out drum transfers had not been tested, and proved to contain one bug. We were able, however, to have DAD running on a drum one and a half hours after the engineers had declared it usable, and this demonstrates the merit of the approach.

12. Further Facilities

Work on additions to DAD is still continuing. A "SHUT-DOWN—STARTUP" system has recently been completed, whereby the contents of the drum can be saved on tape at the end of a day's work and restored the following morning. When the DAD system started operation with only 500,000 words of drum store, trouble frequently occurred due to drum overflow. Now there are one million words of drum, and the difficulty had largely disappeared until the shutdown—startup system was implemented. This has caused the drum to become populated with data that the owners have forgotten, and therefore a scheme to clear the drum of old documents is being designed. With the current job mix, which does not include many programs with huge print output, it is common for documents to last a whole day but usually to be purged the next. About the middle of this year a 100 million character disk will be delivered which will be used as a mass document store. A current weakness of the system is that if certain types of hardware or other errors occur, the content of the drum is lost completely. It is an essential design aim in incorporating the disk into the system that its contents must be preserved despite such disasters.

A further feature to be incorporated shortly is an operator option to reorder EL2. We now have some experience in the running of the system and feel that this would provide a useful gain in efficiency and operating convenience. The system has a changing set of priorities during a day's work. Priority is given to local work for the first eight hours of operation, but during the evening, work from the C.S.I.R.O. subsidiary centers in Sydney, Melbourne, and Adelaide takes first place. Also, special circumstances, such as the buildup of a large backlog of plotting, may indicate a particular strategy as most appropriate. To cater to the above, an option is being added whereby the operator can specify the type of job which is to be favored. EL2 will then be sorted (EL1 is considered not worth treating) on the basis of the strategy given, and jobs will then be run in the order they appear.

The BREAKIN option being developed will enable a high priority job to oust a longer job of lower priority from control of the computer. This should not be confused with the time sharing already in operation in the display subsystem DAVE. This BREAKIN option will facilitate economical use of the 19-in. graphical display unit by allowing main programs to be swapped, and will allow programmers editing and requesting execution of main jobs through the character displays, a faster turnround time.

Acknowledgments. It is obvious that a system such as DAD is very much the result of a team effort and that ideas have been contributed by a great many people. The system has been realized as a result of cooperation between the C.S.I.R.O. Computing Research Section and Control Data Australia. The authors would like to acknowledge in particular the work of I. Wadham and N. Heal (Control Data Australia).

Among those participating in the early discussions on the DAD system were people originally from a number of computing centers in the United Kingdom and the United States, including the Mathematical Laboratories of Cambridge and Manchester Universities, the Jet Propulsion Laboratory, and Stanford University. It is obvious that we have incorporated into the DAD monitor ideas which came from the operating systems of these places, but which we cannot acknowledge separately or adequately. However, particular mention should be made of the TITAN Temporary Supervisor, written by H. P. F. Swinnerton-Dyer of the University Mathematical Laboratory, Cambridge, England.

RECEIVED MARCH, 1967; REVISED MAY, 1967.

REFERENCES

1. PEARCEY, T., AND KERR, R. H. Adapting to the next phase of computer usage. Proc. Third ANCCAC Conf., Canberra, Australia, May 1966, pp. 275–281.
2. AUSTIN, B. J., AND HOLDEN, T. S. The development of a drum and display monitor. Proc. Third. ANCCAC Conf., Canberra, Australia, May 1966, pp. 286–289.
3. KERR, R. H., AND KAROLY, G. The utilization of keyboard display consoles in a conventional operating environment. Proc. Third ANCCAC Conf., Canberra, Australia, May 1966, pp. 389–392.
4. FROST, P. H., AND LANGRIDGE, D. J. A FORTRAN interpreter for use with on-line displays. Proc. Third ANCCAC Conf., Canberra, Australia, May 1966, pp. 345–347.
5. KILBURN, T., PAYNE, R. B., AND HOWARTH, D. J. The Atlas supervisor. Proc. AFIPS 1961 Eastern Joint Comput. Conf., 20, Dec. 1961, pp. 279–294.
6. DAD system programmers' manual. C.S.I.R.O., Canberra, Australia, July 1966.