A basic approach to remote access†

Michael Harrop

I. P. Sharp Associates; Toronto Dominion Centre, Toronto 1, Ontario

The central computer system at the Defence Research Telecommunications Establishment, Ottawa, is a Control Data 3200 with 32K of memory. In addition, several sub-sections within the Establishment have small computers which are used as part of the electronic instrumentation associated with their research projects. Problems encountered by some of the departments using these small machines, in particular the limited computation capability, led to the setting up of the Roll-In System, a limited remote access system which enables anyone of eight remote stations to gain access to the central computer on a priority interrupt basis.

(Received June 1969)

The Roll-In System was devised to meet the requirements of two groups within the Defence Research Telecommunications Establishment.

The first group is involved in the analysis of ionospheric soundings. Before the Roll-In System existed, analogue ionospheric data was converted to photographic records and manually scaled. Data corresponding to points on the ionograms were punched on to cards and the cards submitted to the central computer as input to an editing and computing program.

Under the Roll-In System the analogue data is converted to digital form and displayed on a CRT display unit through a Digital Equipment Corporation PDP9, which is online to the central computer. The operator decides which of the displayed data constitute valid ionogram traces and, using a light pen, selects points on the traces which are then transmitted directly to the central computer by the PDP9. The central computer performs essentially the same calculations as in the older method, but in addition sends back the results to the PDP9 for immediate display and verification. The operator thus has rapid feedback of results and is able to quickly correct errors in judgement.

The second group, involved in research and development into engineering design techniques, was assembling a computer graphics facility for circuit design work. This consisted of a PDP9 (with 8K of memory) and a CRT display with drum backing store. The PDP9 was to be used for man/machine interaction and the display functions, but for major computations it was necessary for the PDP9 to have direct access to a larger machine (the 3200).

The Roll-In System satisfies the immediate requirements of these users and provides for future expansion.

Hardware changes

The original configuration of the central computer is shown in **Fig. 1.** Channel 4 was available as a remote station channel. The hardware constructed to link the remote stations to the 3200 consisted of a transmit-receive unit (known as 'the link'), a channel 4 to 'link' interface, and remote station to 'link' interfaces.

The new channel 4 configuration is shown in **Fig. 2.** The link from the remote computers to the central computer is a bi-directional bus system with all remote stations connected on to the bus. A lock-out feature ensures that a remote user cannot access the bus if the bus is already servicing another remote request.

Each remote station is assigned one of the standard equipment interrupt lines which is used to identify the caller to the central computer. This effectively restricts the number of remote stations to eight (i.e. the number of equipment interrupt lines on the channel).

Software—the Roll-In System

The design of the software to handle remote access to the 3200 was done bearing in mind that the machine would probably be replaced in 12 to 18 months. This factor combined with the absence of a suitable memory protect feature and the fact that multiprogramming on the 3200 (using the Mass Storage Operating System—

† *The system described was set up at the Defence Research Telecommunications Establishment, Ottawa, whilst the author was under contract to the Establishment*

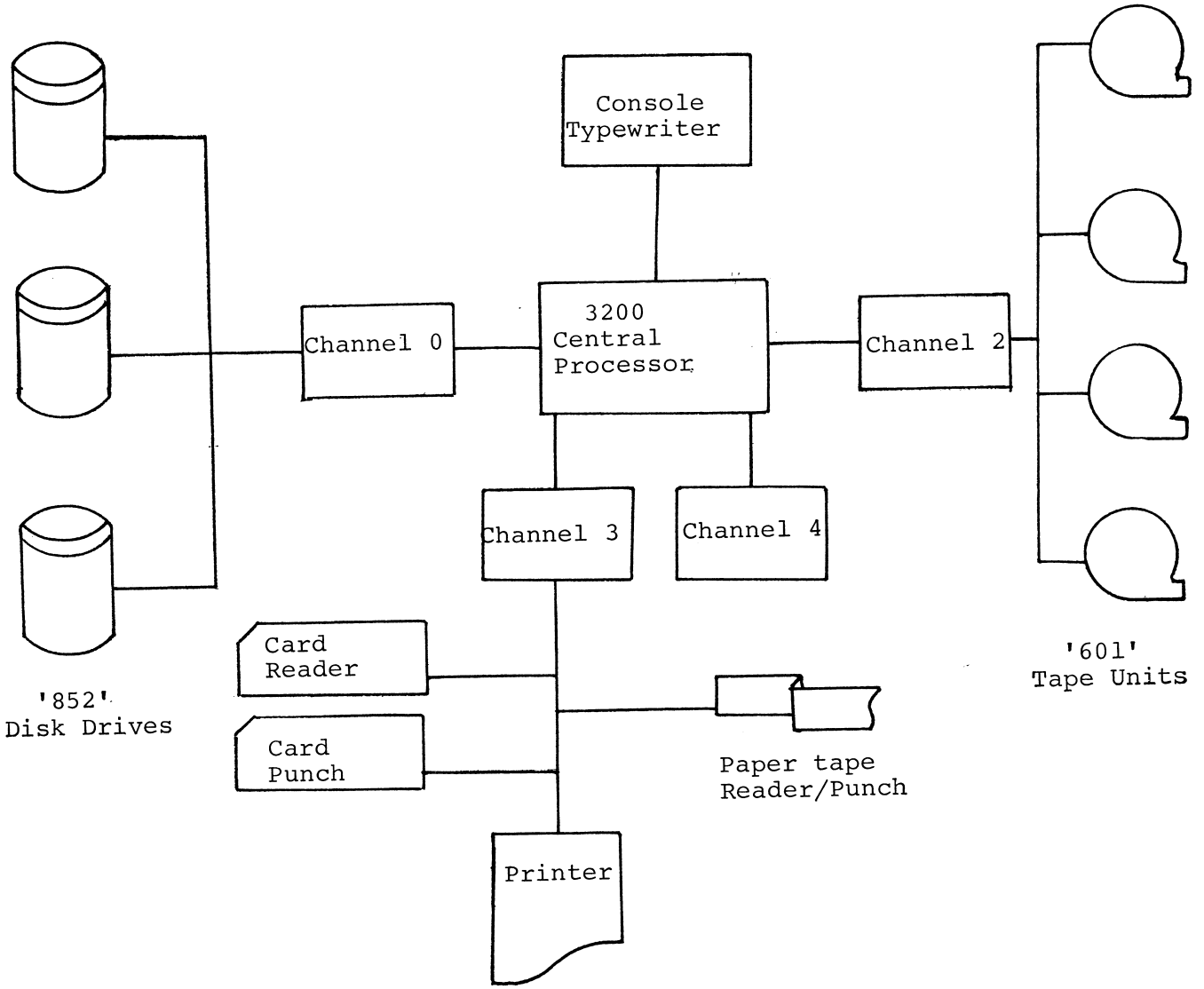


Fig. 1. The original configuration

MSOS) is limited to a simple form of dual program time sharing, led to the decision that the remote access software should be based on a core swap.

The Roll-In System meets the following requirements:

(i) The system permits any one of up to eight remote stations to gain access to the central computer. When the request by the remote station for access has been granted, the central computer remains dedicated to that remote station pending completion of a program associated with the particular remote user.

(ii) A request by a remote station for the 3200 is made via a channel 4 equipment interrupt which is delivered to the 3200 by a special purpose interface attached to channel 4.

(iii) There exist on channel 4, eight equipment interrupt lines. Each remote station of the Roll-In System is assigned one of these interrupt lines. For each equipment interrupt line assigned to a remote station, there is an associated program which is loaded and executed in response to that particular interrupt.

(iv) On acceptance of a request for the 3200 by a remote station, normal processing is suspended, the entire contents of core dumped on to a disc file, and the user program corresponding to the interrupt is loaded and executed. Following the execution of the user program, the original contents of core are restored and regular processing is allowed to continue from the point at which it was interrupted, without the person who submitted the interrupted program being aware that anything irregular has happened during his run.

(v) Input and output for the remote user programs is restricted to operations on the Roll-In section of disc, using a special mass-storage processor, and to operations conducted on channel 4 via the special interface. (The Roll-In System has been assigned exclusive use of a section of one of the on-line disc packs. See below.)

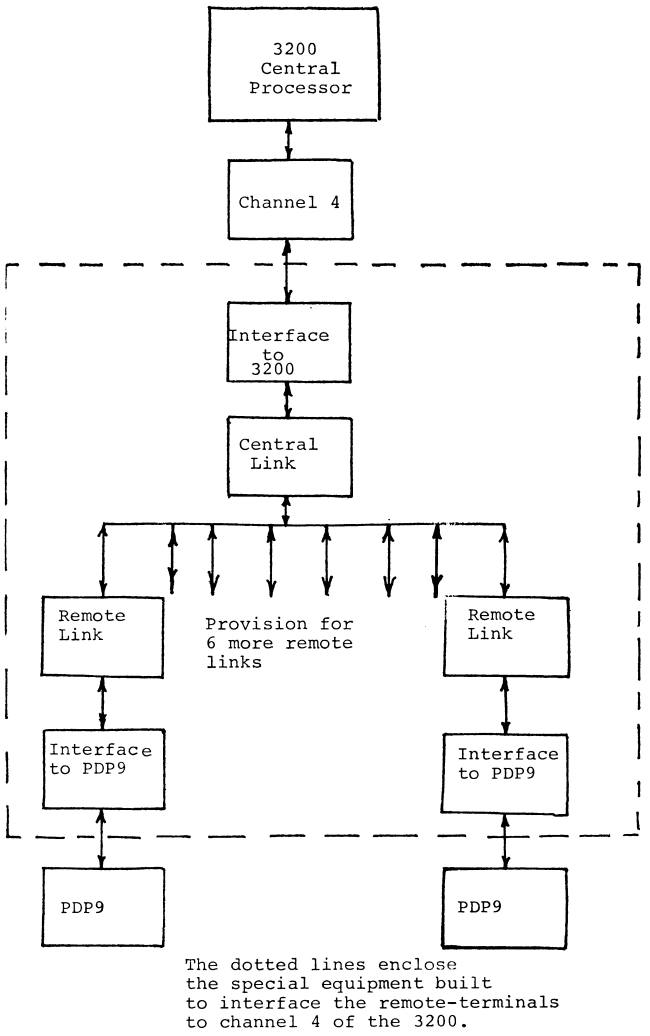


Fig. 2. The new Channel 4 configuration

System subdivisions

The Roll-In System consists basically of three sections:

(i) *Library generation*

This is the section in which user programs are put into a form in which the Roll-In Operating System may retrieve and execute them.

(ii) *The operating system*

This is the monitor program which takes control of the 3200 whilst the remote user program is being executed.

(iii) *The resident section*

This is a program which is resident in the Mass Storage Operating System (MSOS) and which performs part of the core swap. It also provides linkage between MSOS and the Roll-In Operating System.

For programming convenience two other sections were added. These are:

(iv) *Mass Storage communication programs*

These are used by the library generator, the Roll-In Operating System, and by user programs, for performing input/output operations on the Roll-In section of disc.

(v) *Associated utility programs*

These are such things as a directory dumping program, an accounts processor and a re-load facility: programs associated with, but independent of the Roll-In System.

Implementation

(i) *Mass Storage organisation*

A section of one of the disc packs is permanently reserved for Roll-In use. This has been done by deleting the section from the MSIO directories (i.e. the directories used by the MSOS mass storage I/O processor) thus ensuring that the section is not available to MSOS. This section of disc contains:

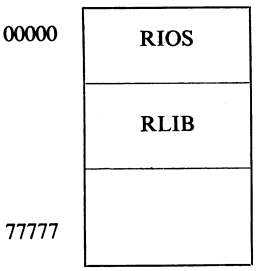
(*a*) Storage space for the entire contents of the 3200 memory. It is on to this section of the disc that the memory is dumped when a Roll-In interrupt is accepted.
(*b*) The Roll-In Operating System.
(*c*) The user programs which are to be loaded in response to the interrupts.
(*d*) Data files—storage space required by user programs.

For reasons of speed and disc space economy it was decided to perform all disc I/O in Track Mode (745 words per track and a minimum record size of one track) rather than Sector Mode (500 words per track with a minimum record size of 25 words) which is used by MSOS. Using Track Mode, the entire memory may be written into 44 tracks in about 1·5 seconds. In Sector Mode the same operation would take about 2·3 seconds and occupy 66 tracks.

The Roll-In section of disc may be accessed only by using the special mass storage communication programs provided. These, by the use of a file directory, enable the user to reserve areas of disc within the Roll-In section of disc. The user may then perform read/write head positioning and input/output operations on that section of disc by referring to a pre-defined file ordinal, rather than by using hardware track addresses. These routines may be run either under MSOS or under the Roll-In Operating System. A user may, therefore, write data on to the Roll-In 'disc' under one operating system and read it back under the other operating system. This overcomes the restriction of not being able to use other I/O devices under the Roll-In System.

(ii) *Library generation*

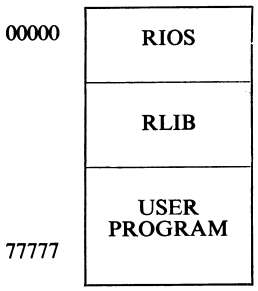
User routines which are to be loaded into core in response to an interrupt are put into the most suitable form for achieving this by the Roll-In Library Generator (RLIB). The method adopted is to load the Roll-In Operating System into low core (into the position it will occupy when rolled-in). Just above the Operating System is loaded the Library Generator which is a modified version of the MSOS LOADER. (See **Fig. 3.**) Each user program is then loaded in turn into high core (also into its rolled-in position). When loading of each user program is complete, that section of core containing the user program and its DATA area (but not its COMMON area) is dumped on to the Roll-In section of disc. (See **Fig. 4.**) When all the user programs have been loaded and written in absolute binary on to the disc, the



The Operating System is loaded into low core, occupying its 'Rolled-In' position.

The library generator lies above the Operating System. This area is released to the user as COMMON at execution time.

Fig. 3



User programs are loaded one at a time into high core and then written out on to the disc in absolute binary.

Fig. 4

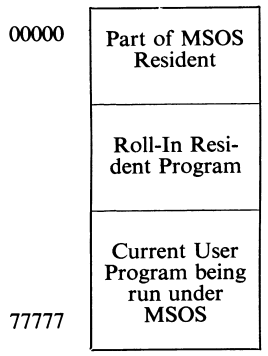
Roll-In Operating System is itself written out on to the disc.

All threading between subroutines, subprograms and the operating system is therefore accomplished at Library generation time and the only loading necessary during the core swap is to read the user program and Operating System directly from the disc into core. (The area used by the RLIB program during the library generation is released to the user as COMMON at execution time.)

(iii) *The core swap*

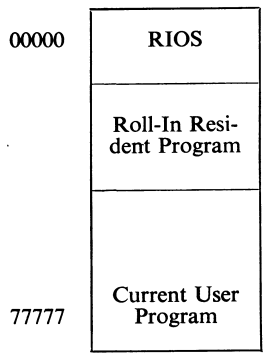
Upon acceptance of a channel 4 equipment interrupt, control is transferred to the Central Interrupt Processor within MSOS which decides whether the interrupt was initiated by a remote station. If so, control transfers to the Roll-In Resident Program (part of MSOS Resident) which decides whether or not the Roll-In can proceed. If the Roll-In facility has not been disabled by a manual interrupt message from the operator, a message is printed on the console typewriter indicating that the Roll-In is underway.

The unit status table (UST) is checked for incomplete input or output operations and the completion of any outstanding I/O is awaited. (The one exception here is that tape re-winds need not be completed prior to the core swap.) The contents of the 3200 internal clock registers are stored for later accounting use. Stage one of the core swap is then initiated. This involves blocking out, on to the core dump file of the Roll-In disc, the lower 2980 words of core (i.e. 4 tracks on the disc). By blocking out this amount of core, enough space is freed for the Roll-In Operating System to be loaded (see Fig. 5). The Roll-In Operating System (RIOS) is then loaded into the vacated section of memory (see Fig. 6) and control passes to the RIOS which completes the blocking out of memory (see Fig. 7) and loads in the user program corresponding to the interrupt received (see Fig. 8). The unused portion of memory is set to an abnormal exit call



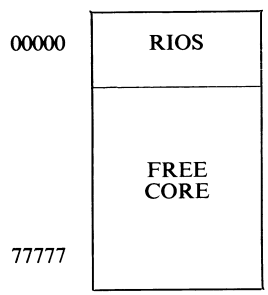
This section of core is blocked out by the Roll-In resident program.

Fig. 5



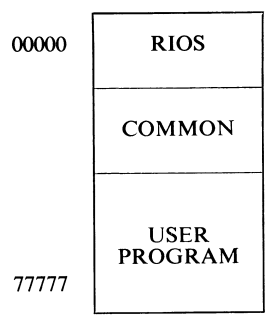
After bringing in RIOS.

Fig. 6



After blocking out the remaining part of core.

Fig. 7



After loading user program.

Core is now in the same state as after loading the user program during library generation (Figs. 3 and 4) except that the COMMON block displaces the RLIB.

Fig. 8

and control is then given to the user program.

When the user program execution is complete it returns control to the RIOS which reloads that section of memory previously blocked out by the RIOS, and returns control to the Roll-In Resident Program (which it has just blocked in with the rest of high core).

The Roll-In Resident Program completes the blocking in, performs some basic accounting operations (which include making sure the MSOS user is not charged for

the time taken by the Roll-In user), indicates to the operator that the Roll-In is now complete, and then gives control back to MSOS which continues processing the interrupted program at the point at which it left off. Should the user program fail to return control to the RIOS, an emergency auto-load feature in the form of a special auto-load card, exists whereby the operator may reload memory and continue the interrupted program as if a normal return from Roll-In had occurred.

System usage

It was anticipated when the system was designed that system usage would vary considerably according to the requirements of the individual groups using it. This has been found true in practice and it is unusual to find two remote stations requesting the machine at the same time. When this does happen, however, the remote station operating on the interrupt line having higher priority is allowed to access the central computer first.

The actual time involved in a complete Roll-In operation depends, of course, on the user program. Usually the total time involved has been of the order of 10 seconds of which 6 seconds is overhead time: core swapping, loading, checking and accounting.

In the case of the group doing analysis of ionospheric soundings, Roll-In requests have usually been made over a period of an hour or two at intervals of about one minute. It has been found that this does not seriously disrupt normal batch processing operations.

Conclusions

The Roll-In System illustrates what can be done in the field of remote access with a machine not originally designed to provide this facility. No figures are available to indicate the cost of the hardware changes, as much of the equipment was built within the Establishment. The design and implementation of the software, however, took about three man months.

The system obviously has its limitations but, as a short term solution, is very effective.

Book review

Automatic Programming, Vol. 6, part 2, by P. J. Brown and G. F. Colouris, 1969; 67 pages. (*Pergamon Press Ltd.*, £1.75)

For Volume 6 of the Annual Review in Automatic Programming, the second to appear since its renaissance under a new Editorial Board, the publishers have adopted a new format, and the volume appears in four separate paperback parts (though a hard cover edition will be available later). This practice will commend itself to those who might balk at paying £7 for a complete volume when only one of the papers is of interest to them.

Most of this part-volume is devoted to 'A Survey of Macro Processors' by P. J. Brown. This is a masterly survey, comprehensive and clearly written. The author first defines a macro processor in a general way as 'a piece of software designed to allow the user to add new facilities of his own design to an existing piece of software', and discusses the main application areas of such systems. He then considers in detail the problems that face the designer of a macro processor, highlighting the difficulties and giving illustrations of the solutions that have been produced by various designers. He succeeds in fitting a number of diverse systems into a general framework that permits meaningful comparisons and gives a clear account of two problems which many designers of macro systems do not even consider, the question of 'call-by-name' versus 'call-by-value' and the question of multi-level calls, (that is, can a macro call be built up of separate pieces of text?). Lest the reader should think that this latter question is merely an academic fancy, the author shows how it has important applications in code optimisation. The problems of implementation are discussed, and the article concludes by returning to the application areas outlined at the start, and drawing conclusions as to the applicability of macro processors in each area. It is rounded off with a comprehensive and cross-indexed set of references.

The remainder of the volume is taken up with a short (15 page) article by G. F. Colouris, entitled 'A Machine Independent Assembly Language For Systems Programs'. The article gives a short account of SICTRAN, a language in which 'in order to achieve machine independence and yet retain the control of program and data organisation required, parts of the run-time structure of the program and its data are left undefined by the translator'. The description is brief, and illustrated by few examples. Possibly the examples are not typical, but they give the impression that one could achieve the same effect by writing a few macro definitions for one of Dr. Brown's macro processors.

The meat of this volume is undoubtably the survey of macro processors. This should be compulsory reading for anyone concerned with the design of such systems, and can be read with profit by anyone interested in computer software. I will certainly make sure that all my students master its contents.

D. W. BARRON (Southampton)

Vol. 6 part 3 appears on page 170