## Summary of the Thesis

The thesis describes certain facilities which have been provided to offer computer users new ways of developing and executing programs. They have been implemented in the EGDON 3 operating system for the KDF 9 computer which is based on a segmented supervisor and incorporates a file storage system. An on-line system, created by allowing certain of the supervisor segments to be activated from the KDF 9 console, was extended to form the COTAN multi-access system by connecting the KDF 9 to a PDP 8 computer which acts as a multiplexer for a number of teletype terminals and a graphical display.

In the multi-access system there were facilities for inputing data and manipulating and editing files. Although these facilities could be used to construct the text of a program, it could only be compiled and executed by entering a job in the batch processing stream of the background system. Now the user, from an on-line teletype, is able to insert an entry into a job queue which is ordered so that the shortest jobs have the highest priority. The job scheduler of the background system was modified to alternately run one job from the batch input and one from the job queue. When required compilation may be semi-conversational with success or failure messages being returned

to the teletype. If the user wishes to interact with a program while it is being run, its execution in batch processing mode is suspended and it is written to disc in a standard manner. At any time following this action it can be reactivated and its execution continued in a fully conversational mode.

These innovations are facilitated by two supervisor segmen which perform the functions of remote job entry and on-line reactivation of a suspended program.

# Remote Job Entry and On-Line Reactivation of a Suspended Program

by

Donald R. Innes

Submitted for the Degree of Master of Science

at

The University of Glasgow.

Acknowledgements.

The facilities described in this thesis were implemented within the framework of the multi-access on-line system COTAN which was developed by Dr. Peter C. Poole at the Culham Laboratory of the United Kingdom Atomic Energy Authority. Much detailed discussion was thus necessary with Dr. Poole and I would like to acknowledge the help he gave me.

Professor D.C. Gilles has kindly acted as my supervisor throughout the period of study and I would like to thank him for his support and encouragement.

## Contents

## 1. Introduction

Now that the feasibility of multi-access systems has been established every effort is being made to develop techniques which will increase their utility and effectiveness. The facilities which are described in this thesis are adaptable tools with which many powerful new techniques may be fashioned.

The facilities were implemented in the new EGDON 3 operating system which has been developed for the English Electric KDF 9 computer. This system is based on a segmented supervisor and incorporates a file storage system which allows user jobs to make effective use of the disc. An on-line system was created by allowing certain of the supervisor segments, which operate on the common data base, to be activated from the KDF 9 monitor console. This system was extended to form the COTAN multi-access system by connecting the KDF 9 to a PDP 8 computer which acts as a multiplexor for a number of teletype terminals and a graphical display.

In the multi-access system there were facilities for inputing data and manipulating and editing files at a teletype. Although these facilities could be used to construct the text of a program it could only be compiled and executed by entering a job in the batch processing stream of the background system. Now the user is able to enter the job at an on-line console. If it is required compilation may be

semi-conversational with success or failure messages being returned to the teletype. Although execution of a program follows its compilation it may be suspended and continued sometime later in a fully conversational manner. The innovations are facilitated by the two supervisor segments, outlined in the following paragraphs, which perform the functions of remote job entry and on-line reactivation of a suspended program.

In order to harness the background system to the multi-access system a remote job entry facility was included in COTAN. To compile and run a program at a console the user can insert, into a queue file, an entry containing the name of the file on disc in which the program is stored, a time limit for its execution, and the number of the console from which it was entered. The queue is ordered on the time limits so that the shortest jobs have the highest priority. The job scheduler of the background system was modified to alternately run one job from the batch input and one from the job queue. So that this queue could be reserved for short jobs, particularly programs in development, a second ordered queue was provided for long production jobs. This queue is serviced as though it was concatenated on to the end of the short job queue. Since the number of the console at which the job was entered is available at compile time, system messages indicating success or failure can be output on it if the user wishes.

The remote job entry facility logically precedes and introduces

the concept of on-line reactivation of a suspended program. If the user wishes to interact with a program while it is being run its execution is suspended and it is written to disc in a standard manner to separate its execution from its compilation. This separation is necessary partly because the actual time of compilation cannot be determined precisely and partly because of the very different system facilities required by a program at compilation and execution times. A high degree of interaction is not feasible unless there is some form of time sharing. The user may be considered to be a very slow input device and it would indeed be a very inefficient system which waited for such a lethargic peripheral. Thus the facility was implemented in the multi-access system in such a way that processor time which is not being used by the interactive task is being absorbed by the job running in the background system. This involves restricting the programmer to only interacting at predetermined points in the program. At these points additional data is requested from the user and execution of the program is suspended. After the user has supplied all the required data the program is reactivated and execution continues from the point at which it was suspended. The user is allowed an unlimited number of bursts of processing and there is

no control over the time between suspension and reactivation. However, there is a limit on the time between reactivation and suspension, when the task is being executed, and failure occurs if it is exceeded. By restricting the amount of computation possible during any one burst of processing it is hoped to reserve the facility for applications in which interaction is necessary or highly desirable and to discourage its misuse. While the task is suspended execution of the job in the background system continues and only the time required to switch from one system to the other and back again is lost. Nevertheless this overhead may be quite high since, if the two programs cannot co-exist in core, part of the background job has to be saved on disc when the interactive task is loaded while the interactive task is always preserved on disc when the background job is being executed. It is therefore clear that a program to be run in this conversational mode requires to be treated in a rather special way and that its activity while being executed should be confined to a well defined precinct.

## 2. The Environment.

No attempt will be made in this thesis to give a complete description of the KDF 9 computer, its associated hardware or its operating system. Not only are they adequately specified elsewhere they are for the most part so to speak "over the horizon". In order to maintain a high degree of relevance particular features, either of hardware or of software, which are referred to in the text will be described as they occur. There are, however, three paramount landmarks whose major characteristics need to be summarised. These are the segmented supervisor, the on-line system, and the file system. Nevertheless no matter how large and general or small and particular these elements may be it must not be forgotten that they are all intrinsic components of an integrated system. As a result, between the recent innovations presented later and the principal elements summarised below, there is a clearly defined interface. Since the innovations take the form of command programs, as described below, and as there are many other command programs performing different functions but sharing the same interface, it is clear that these programs should in their operation conform to standard precepts. Thus command programs in addition to performing a particular activity must also carry out any housekeeping activities that

are necessary to maintain consistency throughout the system. Thus modularity is attained at the expense of repeating similar operations in different command programs. However to improve systems efficiency many of the operations have been absorbed by the on-line supervisor and are only activated from the command program. Yet it is still true to say that many command programs perform a large variety of operations which cannot be deduced from consideration of their function in isolation from the environment.

## 2.1 The Segmented Supervisor.

The original EGDON supervisor was a non-segmented program capable of multi-programming one user job and a number of system functions. It was clear that if the supervisor could be restructured so that those parts which were active for long periods of time but which could be interrupted occasionally for short intervals without decreasing the overall efficiency of the system, those parts which were active for short periods of time but which were only activated ~~activated~~ infrequently, and those parts which provided services for the background job where such services were mutually exclusive, could share core rather than co-exist then a considerable amount of memory would be released for other purposes. The resulting supervisor consisted of a global section and a number of segments. In addition to freeing core the exercise resulted in

the development of a powerful segmentation control package. Each segment is compiled independently to operate in a specified overlay area, of which there are three, and each is assigned a priority, the longest running segments having the lowest priority. Occupancy of an overlay area is determined on the basis of priority. A request for the activation of a segment enters a queue and if the priority of the segment at the head of the queue is higher than that of the segment in the overlay area, then the latter is suspended and the former is brought into the area. The suspended segment will be resumed automatically when the higher priority segment becomes inactive. If the request is for a segment of equal or lower priority then activation is held up until the segment recently in the area terminates its processing. However, a filter mechanism is provided to allow subsequent calls for higher priority segments to reach the head of the queue.

Thus the segment control package uses the priority to govern the allocation of both core and C.P.U. time between the segments assigned to a given area. Via this mechanism the multi-access on-line system COTAN has been developed.

the development of a powerful segmentation control package. Each segment is compiled independently to operate in a specified overlay area, of which there are three, and each is assigned a priority, the longest running segments having the lowest priority. Occupancy of an overlay area is determined on the basis of priority. A request for the activation of a segment enters a queue and if the priority of the segment at the head of the queue is higher than that of the segment in the overlay area, then the latter is suspended and the former is brought into the area. The suspended segment will be resumed automatically when the higher priority segment becomes inactive. If the request is for a segment of equal or lower priority then activation is held up until the segment recently in the area terminates its processing. However, a filter mechanism is provided to allow subsequent calls for higher priority segments to reach the head of the queue.

Thus the segment control package uses the priority to govern the allocation of both core and C.P.U. time between the segments assigned to a given area. Via this mechanism the multi-access on-line system COTAN has been developed.

## 2.2 The On-line System

The on-line system is composed of three main sections, the communication package, the on-line supervisor, and the command programs, all of which operate in supervisor mode.

The communication package is a small section in the global part of the supervisor which controls the flow of messages between the on-line system and various on-line "devices". The latter include a PDP 8 (which in turn acts as a multiplexor for 20 on-line consoles and a display), the monitor flexowriter and the background jobs.

The on-line supervisor consists of two top priority overlays operating in overlay areas 0 and 1. They are activated via calls to the segmentation control package.

The command programs operate in the third overlay area but are brought in under control of the on-line supervisor. They carry out the functions requested by the console users and return output via the communication package. A command program is in a sense an overlay but it is allowed to expand past the end of the supervisor into the user area if necessary. Thus multi-programming between the background job and the on-line system is only possible if there is sufficient space for the two to co-exist in memory. If such is not the case then the background system is held up while

some of its core is used for other purposes.

## 2.3 The File System

The file system is concerned with the management of information stored in the disc. The information is assigned a structure and methods are available for addressing and manipulating it. Precautions are taken to guard against the corruption of the information by machine or system malfunction and back up facilities are provided to ensure effective utilisation of the space available on the disc.

In EGDON 3 the disc is divided into a number of various size units called "logical discs". All permanent information is stored in one such unit called the file storage area. This information can be addressed by the logical disc number and a sector address which is relative to the start of the logical disc. Thus the information held in files can be changed dynamically at run time.

A file is a contiguous number of sectors in the logical disc used for file storage. It consists of two parts - a head and a body. The head contains descriptive and statistical information about the contents and use of the file while the actual information is held in the file body.

The file head is constructed when the file is created and is a fixed number of sectors in length. It contains such information as the name of the file and the file owner, when it was created, how big it is, what it contains, who may access it and in what manner. The permitted modes of access are read only, write only, append only or any combination thereof. The status of each access mode is either public or private. If public, then any authorised user of the system is allowed the corresponding mode of access; if private then only those users whose names appear in a list in the file head are permitted to access the file in this manner. Statistics about the frequency and last time of occurrence of each type of access are also kept in the file head. This ensures that the system always has available information to determine whether a file is being used frequently enough to warrant its retention on the disc. The file head also contains run time markers which govern multiple access to a file. For example any number of users may be simultaneously reading a file but only one user at a time is allowed to write into a file and then only if no one is currently reading it.

The file body is the section of the file actually accessed by

the user. Disc transfers to and from a file are monitored to ensure that file boundaries are not exceeded and that no attempt is made to read non-existent information.

For each authorised user of the system there is a file, in a standard format with the same name as the user, known as a file directory and owned by the privileged user SYS, which contains a list of the names and current whereabouts of all the files owned by the user. The directory file is also used to hold information about the user. This includes upper limits on the use of such resources as disc space and computer time together with statistics about current usage and a list of the user's job codes.

The user SYS also owns a similar directory called SYS which contains a list of all the system files and a special file, called USERNAME, which contains the names of all the authorised users of the system together with the sector addresses of their file directories. This is referred to as the USERNAME directory. Thus the system, in locating a given file, first searches the USERNAME directory for the name of the file owner; this then points to a file directory which in turn points to the file itself. Thus files having the same name but belonging to different users cannot be confused.

3. Remote Job Entry

Clearly in electronic data processing the most general application, and one that must be accomplished efficiently, is that of program development. Yet a major obstacle to the rapid development of programs is the inavailability of prompt machine time during the testing stage. The main reason for this is that most current computers have a single processing unit which services users one at a time and executes programs in succession with negligible interaction. Further, the use of input and output wells necessitates batch processing which, despite making probably the most efficient use of the computer, leads to rigid batching and indeterminate turn-around times that decrease the flexibility of the operating system. Thus the system is designed for a typical user who is tailored to its peculiarities.

Although compilation normally makes heavy demands on memory, when a fast compiler is available, program development does not require much machine time. It is therefore reasonable to suppose that programs in development should have some priority over long production jobs. The problem of system degradation which often results when scheduling is based primarily on pre-assigned priorities can be avoided in a multi-access system.

The term multi-access is reasonably accurate when applied to a software structure but is misleading when applied to the central processor since the essence of a useful multi-access system lies in the software, particularly the common data base, and not in the hardware. Although the solution lies in how processing time can be shared between jobs with different priorities the term time-sharing is somewhat inappropriate, particularly in the absence of multi-programming as it emphasises the ultimate goal of providing each user with a private 'virtual computer'. However, it is evident that a multi-access system's ability to provide, in certain circumstances, the equivalent accessibility of a private computer is a necessary characteristic.

While the great majority of the computer users are orientated towards the batch processing system for program compilation and execution it is inadvisable to produce an alternative system which is radically different if there is no intention to run it exclusively. Therefore, so that programmers will not be confused or feel that their skill and experience are being wasted if they adopt the new facilities two of the design criteria have been ease of use and compatibility with existing features.

The remote job entry facility consists of a command program which places a job ready for compilation and execution into one of two queues, to be run either at the end of the current background job or as a true background job at some later time. In either case the job is executed in the same manner as an ordinary background job. This facility is in fact the front end of the remote job facility of the on-line system. It accepts requests for the entries from on-line consoles, checks the validity of the data and adds the entries to the appropriate queue file. Although at run time all output is handled by the pseudo-off-line system, if the user activates the command JOIN, after entering the job, system messages will also be output on the console.

The queues are stored in two files called REMOTJOB and BACKGRND owned by the user SYS. They are referred to as the remote and background job queues. Statistics in their directory files determine the maximum run times which users can assign to jobs in either queue. Keeping the time limit for a job in the remote queue small and that for the background queue reasonably high ensures that production jobs are entered in the background queue reducing the waiting time for jobs in the remote queue. This last point is not immediately obvious unless it is understood that different users have different time limits.

Each entry in either queue consists of a job card and an auxiliary card. The job card is a modification of the control card which precedes all jobs in the background system. Among the changes the two most important are the introduction of the file name, specifying the file in which the program to be run is located, and the console number. Since this number is available at compile time, system messages describing the course of the compilation, which are output on the paper-tape punch, can also be directed back to that particular console. The auxiliary card contains the date and time when the entry was added to the queue. At present this information is redundant but in future it will be used by a more sophisticated schedular.

At present the schedular alternately runs one job from the background system and one job from the remote job queue. If the remote job queue is empty it alternates with the background job queue. It is, however, possible for the operator to suspend the servicing of either or both of these queues and of the background system. Thus although machine time is not being made available on demand the request is being noted and the response time adjusted according to the amount of time required. The flexibility of the operating system has been increased and it can now be biased in favour of batch processing or time sharing according to the

particular requirement of any installation.

The user inserts an entry in either queue by activating the JOB command program. Successful activation requires that the user should supply the command name and four items of data. Whenever the system expects the user to input a command name the console will print ?** on a new line. A control key on the console may be used at any time to ensure that a console is being serviced by the system. In this case ?** will also be printed if the console is ready to accept a command. Although the user normally supplies the command name JOB, for brevity, only the first letter need be typed. The system will accept the contracted name as an abbreviation and replace it with the full name from an internal list of valid commands. The system thus checks the user's response and rejects any invalid command names. If the command name is preceeded by a full stop the output of command program messages of a confirmatory nature is suppressed.

In normal practice a series of prompts are evoked to remind the user of the data required for the command. The next prompt is evoked by ending the current line on the console with a carriage return character. When the series of prompts is expended the system will request the next command.*

The first prompt following the JOB command is TYPE ?, asking

* for further details consult reference 6, 'A User's Guide to COTAN'

the user which queue is required. The standard replies are REMOTE and BACK referring to the remote and background job queues respectively. However, either data item may be contracted to its first letter.

The next prompt FILE? asks for the name and version number of the file in which the user's program is stored. A reply might take such a form as NEWPROGRAM/1234. A standard EGDON 3 file name of up to twelve alpha-numeric characters followed by a four digit version number separated by a slash from the name part. If the version number is not supplied then the first file with the same name part in the list of files stored in the user's directory file is taken. Only the programs, which are stored in those files of the user which can be legally accessed immediately, may be run.

The third prompt CODE? requests one of the user's job codes. The user has a set of job codes, each of up to six alpha-numeric characters, which may be assigned to different types of jobs. Only the four characters, of one of the eight basic codes stored in the user's directory file, need be specified.

The last prompt is TIME? to which a variety of answers may be given. The basic reply is a time limit for the execution of the job, given as a digital number. Alternately the user may

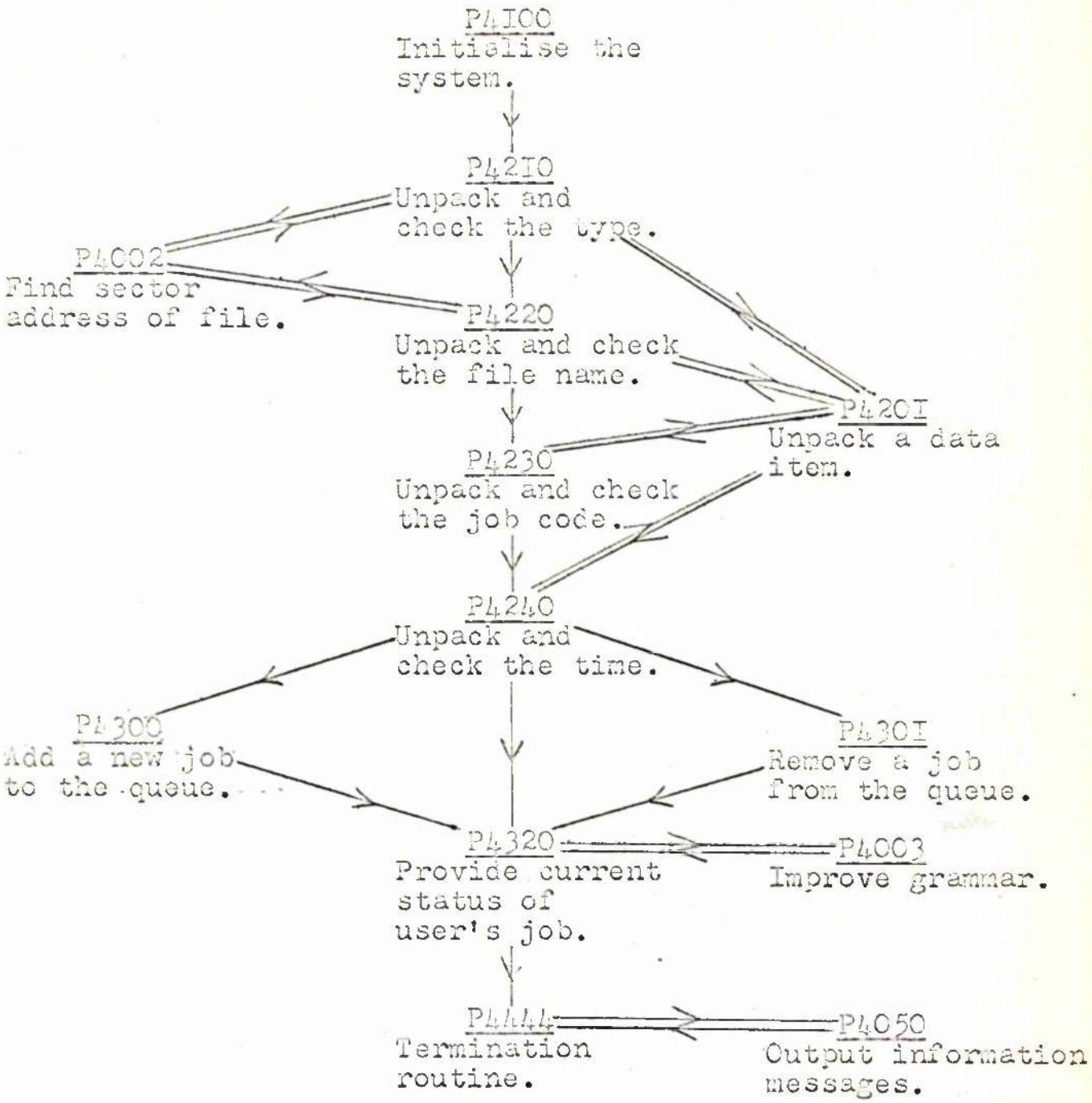
respond with MAXIMUM, REMOVE, STATUS or a contraction of one of these names. In the case of MAXIMUM the maximum time allowed for the execution of any of this user's jobs, stored in the user's directory file, is being requested. REMOVE requests that if the user's job, corresponding to the rest of the data, is in the queue, it is to be taken out. For STATUS, messages describing the current status in the queue of the user's job are output.

Except when they are supressed by the user, information messages describing the state of the system are output. If a fault is found in the data then failure messages are output. The structure of the command program is shown in figure 1 and its operation described in greater detail in the commentary on the major routines.

Although command programs operate in supervisor mode it is possible to partially debug them in program mode using a supervisor simulator. Since both the interface and the environment are well defined a dummy environment is used which imitates many of the reactions of the real environment and checks the validity of the information which is passed across the interface. The checking is effectively syntax orientated so that final testing must be carried out in supervisor mode in the real environment. However, with a combination of simulator checks and standard program mode debugging aids, development of this command program was greatly speeded.

Figure I

JOB COMMAND - BLOCK DIAGRAM




Auxiliary routines

P4001
Alpha/binary conversions.

P4050
Store message parameters.

## 3.1 Commentary on the Major Routines

### 3.1.1 Initialise the System.

A communications area in the segment is loaded with the addresses of variables and jumps to subroutines in the resident part of the supervisor to allow the command program to communicate with its environment. The address to which control is to be returned when the command program terminates, is preserved. The address of the area, in which the on-line supervisor can preserve machine registers, whenever the command program calls an external subroutine, is set.

If the console type number is not that of an on-line console, indicating that the command program is operating on behalf of a background job, either after a direct call or after it has been called into execution via a macro call, normal exit occurs and the message

THIS COMMAND CAN ONLY BE CALLED FROM A CONSOLE.

is output to the line printer. Normal exit will cause the system to obey the next command in the sequence or, if the current command is the last command, to return control to the calling sequence.

If no one is logged in pause exit occurs and the message

NO ONE IS LOGGED IN

is output to the console. Pause exit causes processing of the command file to be suspended and any subsequent commands not yet obeyed are preserved. The console then returns to the command state to allow the user to log in and call the command again.

If the QUIET command is in operation, or if the command has been called with a full stop preceeding the command name, then a quiet marker is set to indicate that the output of messages of a confirmatory nature is to be suppressed.

The current console number is inserted in the job card so that at run time any output requested by the user is returned to the correct console.

The date and time are inserted in the auxiliary card. Since at the present time there is no way of determining the possible run times of jobs in the background system queue the development of a sophisticated job schedular appears to be impracticable. However, when the run times of these jobs are stored in a file as they are loaded into the input well a schedular will be developed which will make use of the information stored in the auxiliary card.

3.1.2 Unpack and Check the Type

Entry is made to the data unpacking subroutine to unpack the type. If this is the only item of data in the input buffer interactive failure exit occurs and the message

DATA INCOMPLETE

is output to the console.* Interactive failure exit causes processing of the command file to be suspended and any subsequent commands not yet obeyed are preserved. When the console returns to the input state the first prompt for the command is output. This form of exit is used whenever a command fails and the amount of data that has been supplied with the command is relatively small. The command interacts with the user who is expected to retype all the data associated with the command.

If the type is neither REMOTE nor BACKGROUND then interactive failure exit occurs and the message

TYPE NOT RECOGNISED

is output to the console, otherwise the type marker is set accordingly.

It is possible, through operator intervention, to stop the servicing of a particular queue. If the requested queue is not being serviced the parameters of the information message

THIS QUEUE IS NOT BEING SERVICED AT THE MOMENT

* See note on page 32.

are added to a list. In order to reduce the number of disc transfers information messages are not output as they occur. Instead their parameters are added to a list and before the command program terminates, if the quiet marker is not set, the messages are read into one buffer and output as one compound message.

It is also possible for the operator to tell the system to service none of the queues when the current job has terminated, even though the queues may be switched on. This is referred to as pausing the system. When the current job terminates the system will go into a pause state, allowing the operator to issue instructions which cannot be given while a job is running. Servicing of the queue may only be continued after the system has been unpaused. If the queue is being serviced but the system is paused the parameters of the information message

THIS QUEUE IS BEING SERVICED BUT THE BACKGROUND SYSTEM IS PAUSED

are added to the list.

The name of the queue file and the logical disc number and sector address of the owner's directory file are set to obtain the sector address of the queue file via the appropriate subroutine.

For each queue there is a queue control word through which the command program can pass information about the queue file to the supervisor card reading package. The control words contain three items of information: the sector address of the queue file, a busy marker which is set when the JOB command program is in operation, and a count of the number of jobs in the queue which are still to be executed. The sector address part is set and the busy marker switched on.

The first sector of the file head of the queue file is read into the core and the number of the occupied and reserved sectors in the file body are compared. If the two are not the same an allowance of one sector is made for a possible increase in the length of the queue.*

The number of words in the file head and occupied sectors of the file body, which have to be read from the disc, is calculated. This plus any allowance made for an increase in the size of the queue is the number of words of external work space required from the on-line supervisor. External work space is taken from the area outside the supervisor which is normally reserved for background jobs. If the background job requires all of this area it is interrupted and part of it is dumped on disc. However, since with this command very little space is required, and the background job rarely requires all of

* See page 28 line 4 where number of occupied sectors is recalculated.

its area, in practice the job is not interrupted. The file head and occupied sectors of the file body are read into the external work space.

Finally if any of the jobs in the queue have terminated the remainder of the queue is shifted forward overwriting this garbage.

3.1.3 Unpack and Check the File Name

The file name marker is set and entry is made to the data unpacking subroutine to unpack the name and version number of the user's job deck file. If the file name is the last item of data, interactive failure exit occurs and the message

DATA INCOMPLETE

is output to the console. The file name and version number are set in the job card in the standard form. If the version number is not specified it is assumed to be zero.

Finally entry is made to the subroutine which finds the sector address of a file to check the status of the user's file. If the job deck file has a legal status, control is returned and is passed to the routine which unpacks and checks the job code.

3.1.4 Unpack and Check the Job Code.

Entry is made to the data unpacking subroutine to unpack the job code. If the job code is the last item of data, interactive

failure exit occurs and the message

DATA INCOMPLETE

is output to the console.

The second sector of the user's directory file is read to obtain the maximum elapse time, in the minutes, allowed for the user's job in the requested queue. This is compared with the value 999 the maximum time that can be put on a job card, and the minimum value retained. The original maximum time from the directory is inserted in the position XXXXXX in the information message

YOUR MAXIMUM TIME FOR THIS QUEUE IS XXXXXX MINUTES

the parameters of which are added to the list.

The job code is inserted in the job card. If there is not a corresponding job code in the user's file directory, interactive failure exit occurs and the message

ILLEGAL JOB CODE

is output to the console.

3.1.5 Unpack and Check the Time

The time is obtained via the data unpacking routine. If it is not the last item of data, interactive failure exit occurs and the message

TOO MANY DATA ITEMS

is output to the console.

If for the time the word REMOVE is given, control passes to the routine which will remove the specified job from the queue. If the word STATUS is supplied, control passes to the routine which outputs messages describing the current status, in the queue, of the job. If the reply is the word MAXIMUM, it is replaced by the maximum time, previously determined, which the user is allowed to give to a job in the queue.

If the time specified is not one of these, then, if it is zero, greater than the allowed maximum, or not all numeric, interactive failure exit occurs and the message

TIME SPECIFIED UNACCEPTABLE

is output on the console.

If a legal time is specified then it is inserted in the job card and control is passed to the routine which will insert the job entry in the queue.

### 3.1.6 Add a New Job to the Queue

If there is no room for another entry in the queue interactive failure exit occurs and the message

THE QUEUE IS FULL

is output to the console. Otherwise the run time of the new job is compared with the times of the jobs already in the queue, waiting to be executed. The new entry is inserted immediately before the first job in the queue which has a greater time limit. If no such entry is found the new entry is placed at the end of the queue.

3.1.7 Remove a Job from the Queue.

Starting at the beginning of the queue, each job, excluding the one which is currently running, is compared to the new job. If one is found with the same file name as the specified job it is removed from the queue.

3.1.8 Provide the Current Status of the User Job

A word containing ENDQUEUE is inserted in the queue immediately after the last entry. This marker is used by the supervisor card reading package. The rest of the card is set blank and the first word of the next card is loaded with *ENDBLOC. This marker is used by the other command programs which may operate on the file. Information in the file head indicates the number of sectors which contain information while the *ENDBLOC marker determines the end of information within the last occupied sector.

Since the queue files have no unique status and may be operated upon by a variety of system programs the statistics contained in their file heads have to be updated by the command program to conform to general standards. Since the length of the queue may have altered the number of occupied sectors is recalculated and the marker, indicating that the file body has been altered since the last prime, is set. The name of the last user to alter the file, in this case SYS the file owner, and the date and time of the last reading and overwriting are set. Since both the file head and the file body have been altered, their sum checks, which will be recalculated the next time the disc is primed, are set blank. Under the statistics corresponding to the file owner SYS, the number of times that SYS has read and overwritten the file body are both incremented and the date and time of the last reading and overwriting are set.

If the user's job is no longer in the queue, or if it is currently being run, then the parameters of the information message

YOUR JOB IS NO LONGER IN THE QUEUE

are added to the list. Otherwise the number of jobs before the current user's job in the queue, and their time limits, are toted up and inserted in the information message

XX JOBS IN THE QUEUE - - TOTAL TIME XXX MINUTES

Entry is then made to an auxiliary routine which suppresses irrelevant parts of this message and the parameters of the message are added to the list.

The head and body of the queue file are written back to disc. In the queue control word, the number of jobs in the queue is set and the file busy marker is cleared.

3.1.9 Termination.

Since all exit routes, including failure routes pass through this routine, the busy marker in the queue control word is set unbusy. If the quiet marker is not set then entry is made to the subroutine which outputs the information messages' buffer.

A number is set indicating whether a normal exit or a failure exit has occurred. The return address is fetched and control passes back to the on-line supervisor.

3.1.10 Output Information Messages in One Buffer

If there are any information message parameters in the list the messages are loaded into one forty word buffer and output to the console. They are loaded in the same order in which their parameters were added to the list and the necessary control characters are inserted.

3.1.11 Obtain the Sector Address of a File

The first sector of the user's directory file is read to determine the number of sectors containing entries describing the user's files. Thereafter each occupied sector of the directory file is read and the file name part of each entry matched against the name of the stated file.

If a match is not found, interactive failure exit occurs and the message

FILE NOT FOUND

is output to the console.

When a match is found the other parameters in the entry describing the file are examined and if an illegal state is found, interactive failure exit occurs and the message corresponding to the particular state is output to the console.

Possible messages and their meaning are

THIS FILE DOES NOT EXIST

there are no parameters for this file,

FILE DELETED

the file has been deleted or renamed,

FILE TO BE ARCHIVED

the file is to be removed from the disc at the next archive run,

FILE ARCHIVED

the file only exists on an archive tape,

FILE TO BE RESTORED

the file is to be restored to the disc from an archive tape.

In addition, if a fault has been found in the file during the last disc prime the warning message

** YOUR FILE COULD BE FAULTY

is output to the console.

If the file is not in an illegal state its sector address is fetched from the entry and supplied to the calling routine.

3.1.12 Unpack a Data Item

The data input at a console is presented to the command program in a forty word buffer. Between each item of data is a separator character and the data string ends with a termination character. Any character which is not a letter, a digit, or a control character is ignored. At each entry one item is taken from the input buffer and the residual data saved.

The file name is a special case, firstly because the name part may be more than one word long and secondly because the version number part is effectively a second item.

There are two exits back to the calling routine from this

subroutine, dependent upon whether, after extracting the current item, the input buffer is empty or not.

NOTE

The complete list of responses to the prompts is assembled in the P.D.P.8 and handed over to K.D.F.9 before the start of the execution of the command program.

## 4. Reactivation of a Suspended Program

It has been stated in the introduction that a program to be run in a conversational mode requires to be treated in a rather special way and that its activity, while being executed, should be confined to a well defined precinct. This will be explained in the first two sections of this chapter. The first describes the special treatment which is necessary to minimise the overheads incurred in program swapping and to separate execution from compilation. The second gives a detailed specification of the design of the precinct in which the program is active. This does not refer to the ALGOL command in COTAN which is based on the Whetstone translator

### 4.1 The Program Consolidation and Initial Suspension Mechanisms

When a program is run in the background system, independent of its actual size, it always occupies the whole of the program area allocated to it. The reason for this is that, while the coding occupies the lower end of the program area, the data is stored in the upper part. Thus there is a gap, between the coding and the data, which is used by neither of them. Since the interactive system requires that the user's program should frequently be swapped in and out of core it should be as small as possible to minimise the transfer time. The size of a

program is determined by its highest data address and this may be reduced using the program consolidation mechanism. Consolidation means moving the gap above the data area, thus bringing the coding and data together. This juxtapositioning is achieved by preceeding the first declaration of the program by the declaration of a large dummy array.* This array should be the same size or slightly smaller than the gap. The data which is declared first always occupies the area at the top end of core so that the program's data declarations will be allocated the space between it and the coding. However, since the system cannot recognise that part of the data area is redundant the user has to supply the SUSPND routine of the initial suspension mechanism with the number of words of the program which have to be written into the file on disc.

The interactive system operates in a restricted environment within which system and utility programs will not function. Since these programs include the assemblers and compilers it is necessary to precompile the user's program in the background system. After it has been compiled, at a predetermined point in its execution, it writes a copy of its core state, which includes a dump of the program accessible machine registers, into a file on disc. The

* In the case of a Fortran program this array must be named by the first identifier in the common list.

dumping and writing to disc is performed by the routine SUSPND which is resident in a file and made available to the user's program by the inclusion of a macro call which names this file. Thus the routine is compiled with the user's program. Since the routine can be called into action at any point in the execution of the program, data areas may be initialised and basic calculations completed before it is suspended. The points at which interaction is requested should, however, occur after the point of suspension. Therefore the program may be divided logically into two parts. The first part can fully utilise the existing facilities of the background system while the second can be designed principally to take advantage of the interactive facilities.

Although it is referred to as a suspension mechanism the operation of the routine is that of taking a snap-shot of a particular machine state. On entry to the routine all program accessible machine registers are saved in reserved core locations. A control card is then read identifying the user, the file into which the program is to be written, and the number of words of the program which have to be dumped.* The user's name and the file name are used to obtain the actual address of the file on disc. The specified part of the program, calculated from the

* Layout of card.
Word 1: User's on-line identifier
Words 2 & 3: File name and version number.
Word 4: Number of words to be dumped.

base address, is written into the file. Finally the machine registers are reset and control is returned to the point in the program from which the suspension routine was called. This final action is the first action carried out by the program when it is reactivated for the first time because the original initial jump which normally takes control to the first instruction of the program is replaced by a jump to this point in the routine. After control has been returned, the program continues to run to completion as though suspension had never occurred. Since the action of the suspension mechanism has no effect on its state, the program can be completely tested in the background system provided that dummy responses are made to requests for interaction.

## 4.2 The Interactive System

On-line reactivation of a suspended program, like the remote job entry facility, consists of a command program. However, when active, this command program usurps completely the power of the resident supervisor. In order that it may wield this power responsibly it is well endowed with facilities normally denied to command programs. By this it is not meant that the total structure of the supervisor has been duplicated. This implies,

since there is a correlation between structure and function, that many operations are still carried out in the supervisor. Nevertheless, even in such circumstances the command program exercises supreme authority.

The prerogative is assigned to the command program because it requires to construct a precinct which is inconsistent with the existing environment. This allows the same requests of the user program being executed in this interactive system to be interpreted differently from how they are when it is run in the background system. However, since the interactive system, just like the resident supervisor, must service other parts of the system, the body of the supervisor is preserved while the head is replaced and a new limb grown.

A contradiction arises when the use of peripherals by interactive programs is considered. It has been assumed that these programs may be tested in the background system before being run conversationally. However, not only must a background job not access a peripheral directly, but it never has access to a console. It is obvious that pseudo-off-line use of peripherals is incompatible with interaction. The solution adopted was to treat the {monitor / teletype} keyboard either as a paper-tape reader or a card reader and the {monitor / teletype} printer either as a paper-tape punch

or a line printer. In the interactive system all data transfer requests are trapped and they are assumed to refer to the user's on-line console.* However, an exception is made for messages to the graphical display which may be used on-line in the background system. A mechanism has been implemented whereby both systems may share the display using it as an output device.

Specifying the action of this command program is rather like describing the colour of a chameleon. Its operation is related to the actions of the user, the user's program, and the resident supervisor. Therefore these relationships will first of all be discussed individually.

### 4.2.1 Relationships with the User

Essentially the user can only control the initiation and termination of the command program. Depressing the 'stop execution' control key on the console causes the on-line supervisor to set a marker which the command program recognises as a request to terminate its operation. Initiation also occurs via the on-line supervisor but, depending on the particular circumstances, may take one of the following three forms.

* Any attempt by an interactive-mode program to use magnetic tapes or the disc leads to failure. This implies that there can be no CHAIN changing within the interactive system.

In the case of the reactivation of a program, which has been written to disc by the initial suspension mechanism, a copy of the file containing the program is transferred into a work area of disc known as the file nesting store. One of these areas is associated with each on-line console and operates as a push down store, the top cell of which can be accessed by a command program. By issuing the command EXECUTE the user calls the command program which reads a copy of the contents of the top cell into core and, obeying the modified initial jump instruction, reactivates the program from the point at which it was suspended.

If at any point in its execution the user program requests data the command program outputs a prompt to the console, retains the initial state of the program in the second top cell of the file nesting store, writes the current state into the top cell and then terminates. When data is supplied the command program is recalled and entered interactively. It returns a copy of the current state of the user program to core, transfers the data to it and resumes its execution. Input from remote consoles is effectively batched and the user may supply more data than is requested up to the limit of the size of the on-line system input buffers. Whether all of the data will be transfered to the user program and utilised, depends upon the size of its buffers and

how they are employed.

If the user does not wish to supply data after a prompt the current state of the program, held in the top cell of the file nesting store, may be saved in a named file. At some later time, to continue the conversation, a copy of this file is transferred back into the file nesting store and the command RESUME issued. This calls the command program to reactivate the user program by reading a copy of it into core and repeating the last prompt output at the previous session. The user then continues by supplying the appropriate data as above.

### 4.2.2 Relationships with the User Program

The command program calculates the in-core size of the user program and requests this amount of external work area from the on-line supervisor. The base address and number of locations, of the part of this area which is to be occupied by the coding and data of the program, are set in the memory protection register. The user program is then read into core and execution recommences as though it was a standard binary program.

While the user program is being executed it may make a request to the supervisor, which takes the form of a machine interrupt. It is the duty of the command program to trap, decode, and service such interrupts. Since in many cases there is very little correlation between the requests of the user program and the services performed by the command program their connexion will be discussed. It should, however, be noted that, if the command program accepts a request, the user program will be given the impression that it was serviced normally.

The simplest requests are those for the current date and time, and the user's job code. The command program obtains this information from the on-line supervisor and passes it over, in the required format, to the user program.

The only peripheral transfer request which is serviced in the normal way is the one to output a message to the graphical display. The command program adds the message to the display message queue which is normally only accessed by a background job. If the message contains a standard display file then the picture will be refreshed automatically.

Messages to the line printer and paper-tape punch are output on the teletype printer. Since this is a relatively slow device,

long line printer messages are truncated. Output to the card punch is ignored since to provide facilities to handle these messages would have involved unnecessary duplication.

Requests for input from the card reader and paper-tape reader are referred to the on-line console. Data input at the teletype keyboard is converted to the format expected from the requested device and passed over to the user program.

A request for termination from the user program causes the command program to terminate itself. However, when a failure condition is found by the user program it may request a core dump before the failure termination. Such a request is interpreted as a request to restart the user program from the point at which it last read data. Detection of a failure condition by the machine hardware results in direct termination.

A time limit of twenty seconds is imposed on the user program. This is determined to be the maximum time between reactivation and suspension, as defined in the previous section.

4.2.3 Relationships with the Resident Supervisor.

Since the resident supervisor contains the bulk of the

information about the state of the system, during its operation the command program is obliged to access this information via the communications area. In addition the supervisor contains the message queues and their servicing routines. The command program, therefore, only inserts and removes messages while the actual transfers are performed by the supervisor. However, since control resides in the command program, it is necessary that it should regularly be passed to the supervisor so that the queues may be serviced.

## 4.3 Structure of the Command Program

Because of the variety of the relationships already described, the command program performs a large number of apparently independent functions. However, where two or more of these functions are similar they are performed in a similar manner. This leads to the overlaying of one function by another to produce an agglomerate with no obvious structure. To simplify the description of the action of the command program, it will be considered that a particular function is performed by a section whereas a group of similar or closely connected functions are performed by a phase. From this point of view the structure of the command program consists of connected phases (Figure 2). Since a number of programmes

The Interactive Command Program

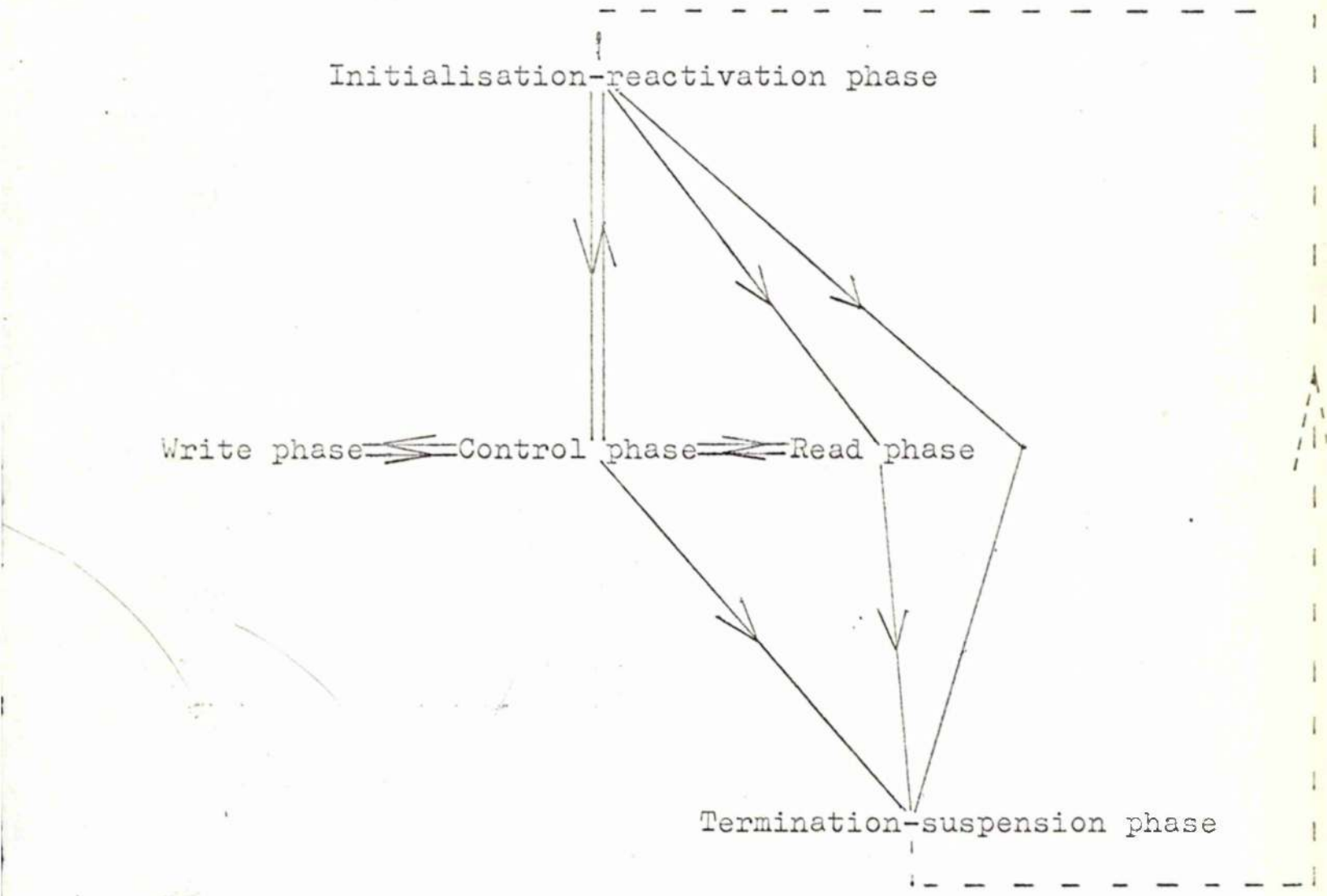



Figure 2

might be using the interactive system at the same time, the command program has been made re-entrant so that a new copy is not required whenever the active user changes.

### 4.3.1 Initialisation-Reactivation Phase (Figure 3)

This phase controls the initialisation of the command program and the reactivation of the user program. Reactivation by the user always involves initialisation of the command program. However, reactivation from within the command program by the control phase, operates directly.

#### 4.3.1.1 Commentary on the Routines.

##### 4.3.1.1.1 Initialise

This routine starts by performing the basic actions of the command program : initialising the communication area, providing the on-line supervisor with a dump area, and preserving the entry number and return address.

The interactive marker and the address of the display queue parameters are obtained. A check is made that there is a file in

# The Initialisation-Reactivation Phase

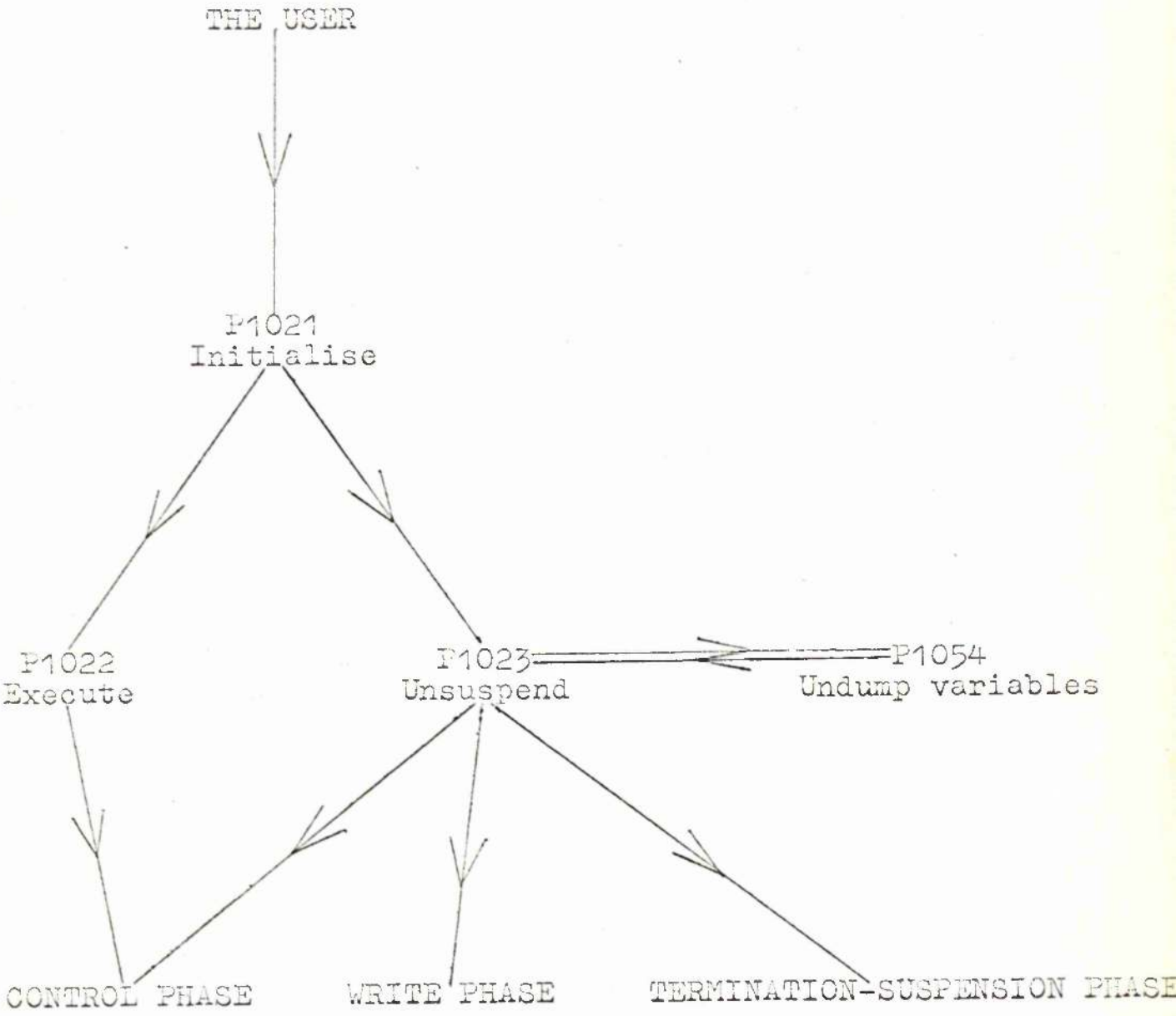



Figure3

the file nesting store and, if there is, that it is not larger than the external work area available to it. If the entry is not interactive a system message is output to the user's console indicating that the command program has been initiated.

The layout of the user program in core differs from the machine state preserved by the initial suspension mechanism. In order that the command program's buffers and variables can be preserved during any future suspension, an additional data area is attached to the front of the user program. This new program state and the position of various command program pointers is shown in Figure 4.

Sufficient external work space for this layout is requested, the pointers are set, and the machine 'base address/number of locations' register is loaded. Finally the machine 'overflow' and 'test' registers are cleared and the run time limit of the user program is set. The entry number indicates whether control is to be passed to the execute routine or the unsuspend routine.

4.3.1.1.2 Execute

The class of the file in the top cell of the file nesting store is checked to make certain that it contains a binary program

Layout of User's Program in Core

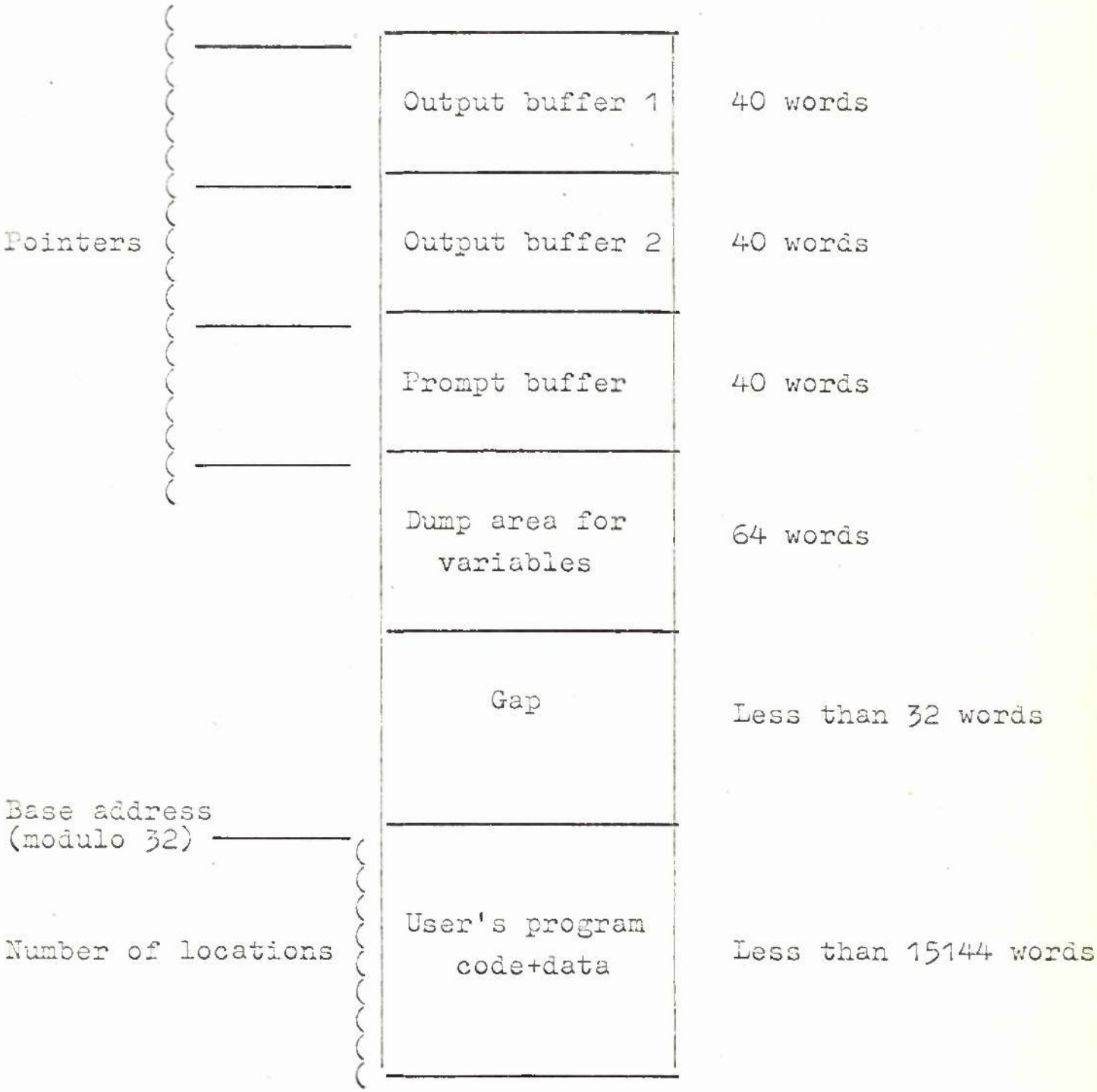



Figure 4

A copy of this program is read into the space in the external work area reserved for it and the address of the point at which execution is to commence is set. The count of the number of words in the prompt buffer is zeroed and control is passed to the control phase.

4.3.1.1.3 Unsuspend.

The type and class of the file in the top cell of the file nesting store is checked to make certain that it contains a binary program which has been suspended by this command program. A copy of the user program is then read into the external work area and the undump routine entered to reset command program variables. If entry was interactive, control is passed to the read phase otherwise it is passed to the termination phase.

4.3.1.1.4 Undump

Command program variables, which were not held in machine registers but which were saved with the user program during the previous suspension, are reset. These include the contents of the machine registers which were preserved when control was returned to the command program from the user program, following a request for additional data, and the count of the number of words in the prompt buffer.

4.3.1.2 Execute Section (Figure 5)

By issuing the command EXECUTE the user calls the command program to reactivate a program produced by the initial suspension mechanism. When the user program is read into core the additional data area, initially empty, which in future will be considered an integral part of the program state, although not directly accessible from within the program, is attached to it. Control is then passed to the control phase which initiates and supervises the execution of the user program.

4.3.1.3 Resume Section (Figure 6)

The user issues the command RESUME which calls the command program to reactivate a program which it has suspended. Instead of supplying data for the prompt output at the time of suspension the user has transferred from data mode to command mode. Any sequence of events may have been performed since then as long as the suspended program is now in the top cell of the file nesting store. The user program and the command program are returned to the same state as they were in prior to the previous suspension. This involves reading the user program into the core and resetting the

The Execute Section

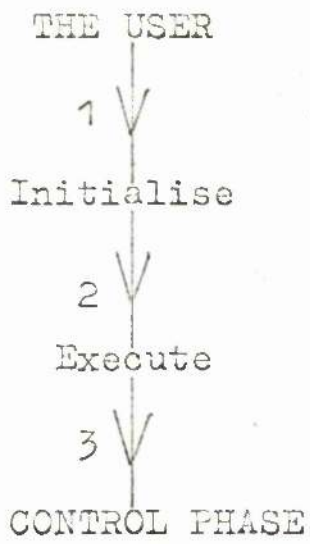



Figure 5

The Resume Section

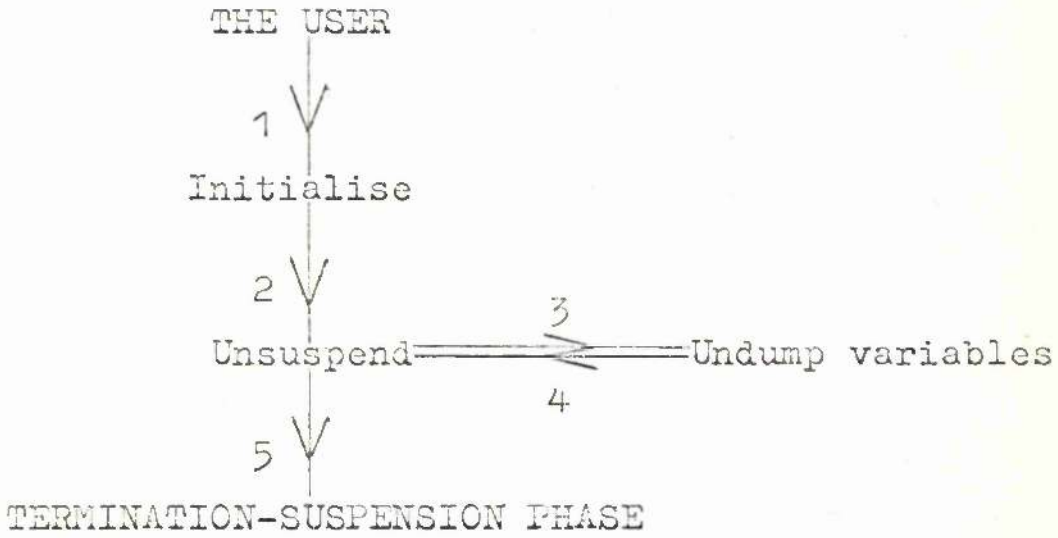



Figure 6

command program variables which have been dumped in the additional data area. Control is passed to the termination-suspension phase to repeat the previous prompt and await the required data.

4.3.1.4 Interact Section (Figure 7)

This section reactivates a program, suspended by the command program, after the user has recalled the command program by providing the data requested by the previous prompt. The user program is read into the core and the command program variables reset. To transfer the data from the buffer of the on-line supervisor's communication package into the buffer of the user program, control is passed to the read phase.

4.3.1.5 Restart Section (Figure 8)

Previously if, in response to a prompt, the user input data which was illegal in the context of this request, the user program output a message indicating why the data was unacceptable. It then output the contents of a section of core to the line printer and terminated, causing the command program to return control to the on-line supervisor. The user had to start with the original

The Interact Section

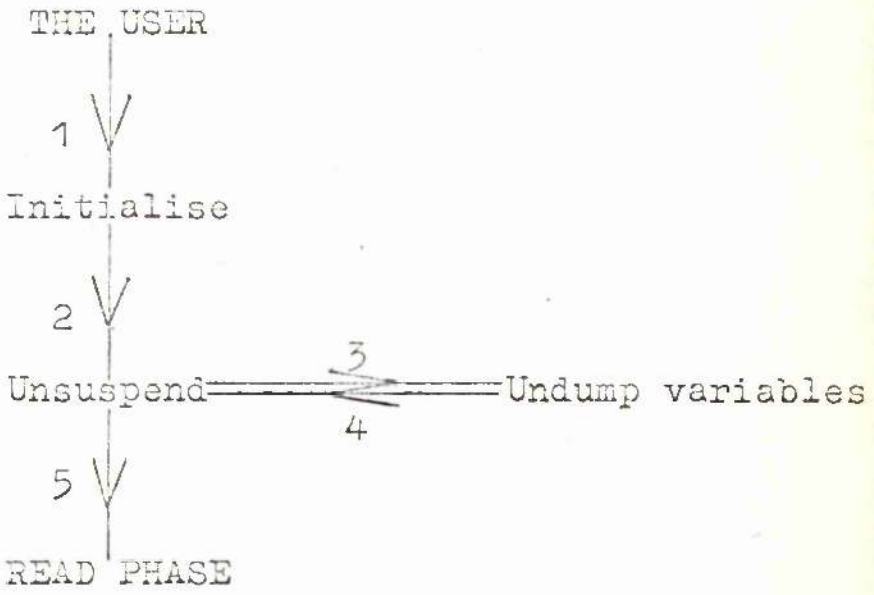



Figure 7

The Restart Section

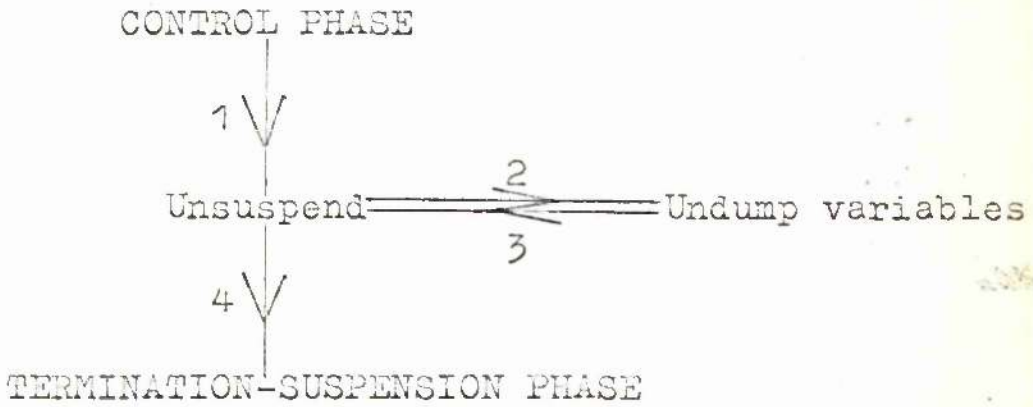



Figure 8

version of the program, issue the command EXECUTE, repeat the complete interactive sequence up to the last prompt, and then input the correct data.

However, the system was reorganised so that, during the interactive sequence, the original version of the user program was retained in the second top cell of the file nesting store while at each subsequent reactivation the current state was preserved in the top cell. This allowed the user, after a failure, to reactivate the program from its previous state by issuing the command RESUME. The last prompt was repeated and the correct data provided.

When this was done it became evident that the previous prompt could be repeated without intervention by the user. This is accomplished by allowing the failure message to be output but interpreting the request for a core dump as if it is the command RESUME issued by the user program. Since the command program is already initialised, the resume section route is intercepted at the unsuspend stage and thereafter this section acts as though the user had issued the command RESUME.

4.3.2 Control Phase (Figure 9)

This phase initiates the running of a suspended program and

The Control Phase

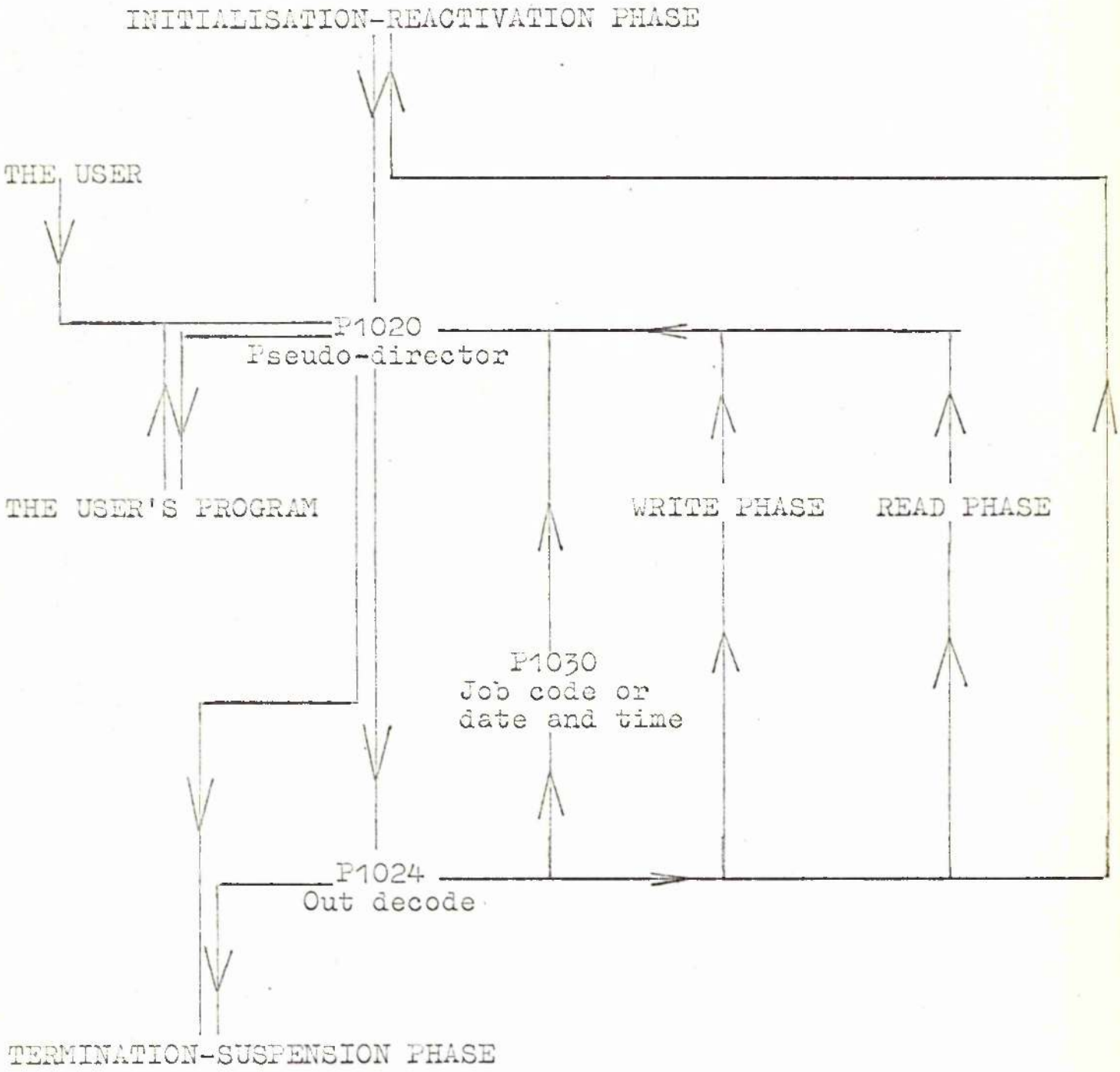



Figure 9

controls its execution by trapping and decoding interrupts. Although it services certain interrupts, in most cases control has to be passed over to another phase or to the body of the resident supervisor. However, control must be returned to this phase if it is to be given back to the user program.

4.3.2.1 Commentary on the Routines

4.3.2.1.1 Pseudo-director

Since this routine contains the general mechanism for handling interrupts which is already well documented, as it is found in standard supervisors for the machine, only the most significant points will be described.

When an interrupt occurs, control is transferred to a fixed location. In the standard system, by obeying a jump instruction, control is then passed to a routine in the resident supervisor which handles the interrupt. On entry to the pseudo-director routine a copy of this jump instruction is preserved and the original is overwritten by another jump instruction which will cause control to pass to this routine when any further interrupts occur. In this way the command program gains overall control of the operating system.

The first concern of this routine is to update the clock and terminate the user program if it should exceed its time limit. Termination may occur upon request by the user or the user program, or if an error is detected by the hardware. In fact if a program request is found, control is passed to the out decode routine while termination is handled by a separate phase. However, with end of data transfer interrupts, control is passed to the queue servicing routines of the resident supervisor.

4.3.2.1.2 Out Decode

The out decode routine translates program requests and interprets them with regard to the available facilities. Control is then passed to the section, normally in another phase, that will service the particular request.

4.3.2.1.3 Job Code and Date and Time

Entry is made to this routine either to obtain the job code of the user or the current date and time. In both cases the information is obtained from the resident supervisor, by way of the communications area, and converted to the required format.

It is made available when control is returned to the user program via the pseudo-director routine.

4.3.2.2 Execute Section (Figure 10)

This is an extension of the execute section in the initialisation-reactivation phase. The jump instruction is planted which will cause control to be transferred to the pseudo-director routine after any subsequent interrupts. The user program in core is restarted from the point at which it was suspended by causing control to obey its initial jump instruction and continue its execution.

4.3.2.3 Input Section (Figure 11)

Requests for data to be input from the paper-tape reader or card reader cause control to be passed to the read phase.

4.3.2.4 Output Section (Figure 12)

If the user program requests that a message should be output to the paper-tape punch, card punch, line printer, or graphical display, control is given to the write phase.

The Execute Section

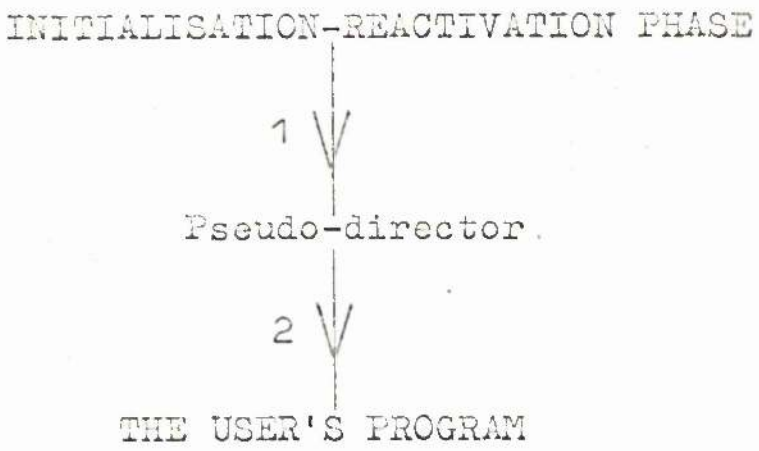



Figure 10

The Input Section

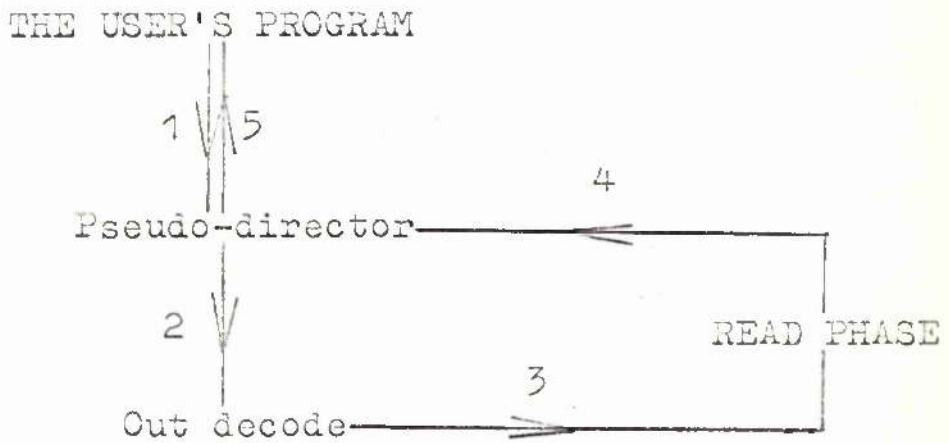



Figure 11

The Output Section

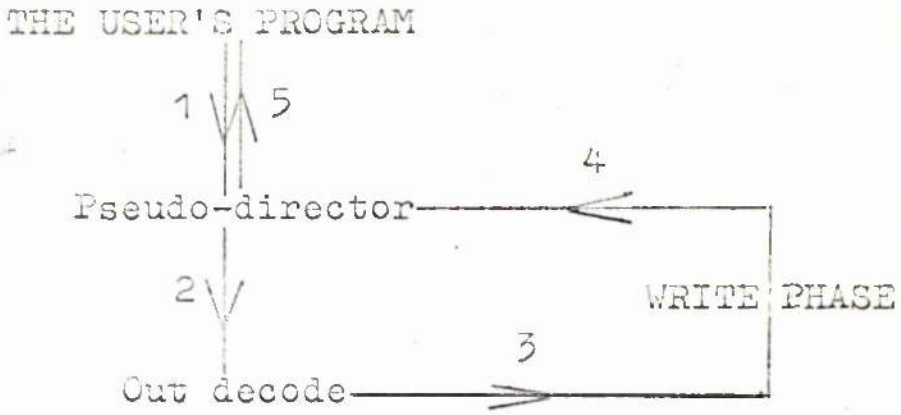



Figure 12

The Restart Section

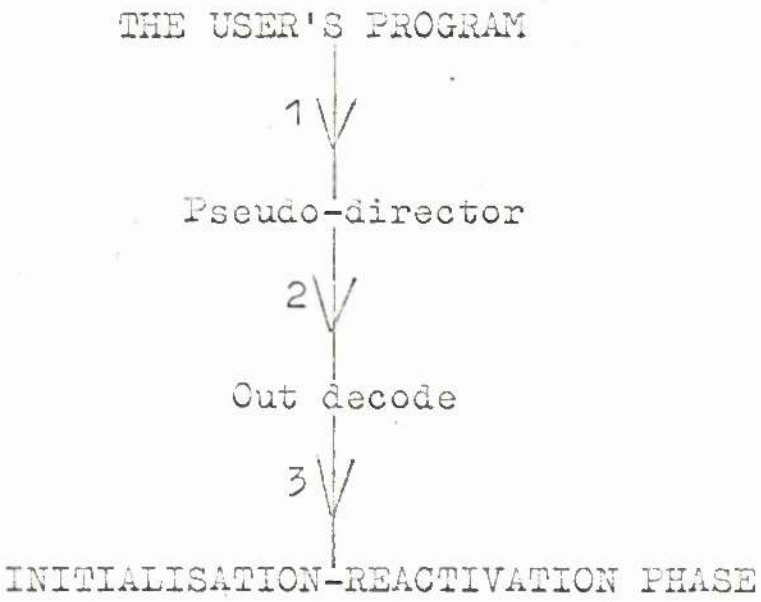



Figure 13

4.3.2.5 Restart Section (Figure 13)

A request from the user program to carry out a core dump is interpreted as a request to restart the program from the previous interactive exit. Control is passed to the restart section of the initialisation-reactivation phase to carry out the actions already described.

4.3.2.6 Utility Section (Figure 14)

Since servicing a request for the user's job code or the current date and time can be easily implemented, it is handled completely within the control phase.

4.3.2.7 Failure, Terminate, and Discontinue Sections(Figures 15,16 and 17)

These sections are different paths leading to the same conclusion. Termination follows a request to finish made by the user program, discontinue is a consequence of the user depressing the stop execution control key on the console, and failure results when the machine hardware detects an error condition.

4.3.3 Write Phase (Figure 18)

Although this is referred to as a write phase the command

The Utility Section

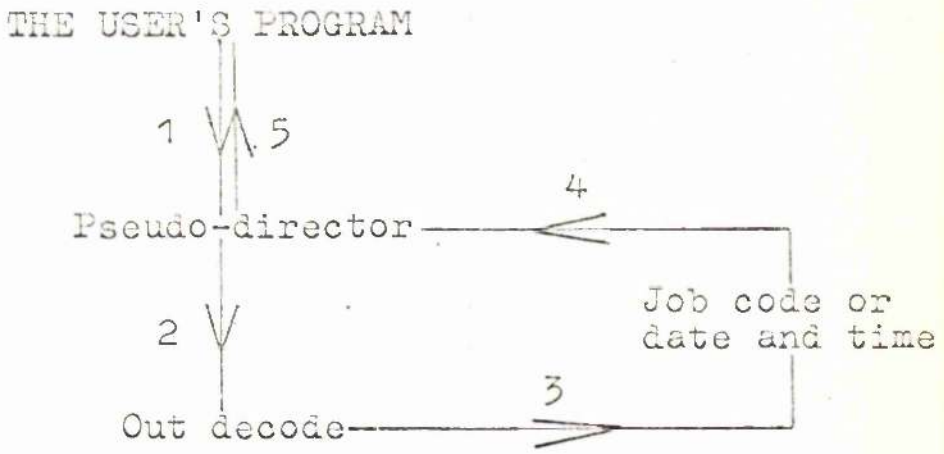



Figure 14

The Failure Section

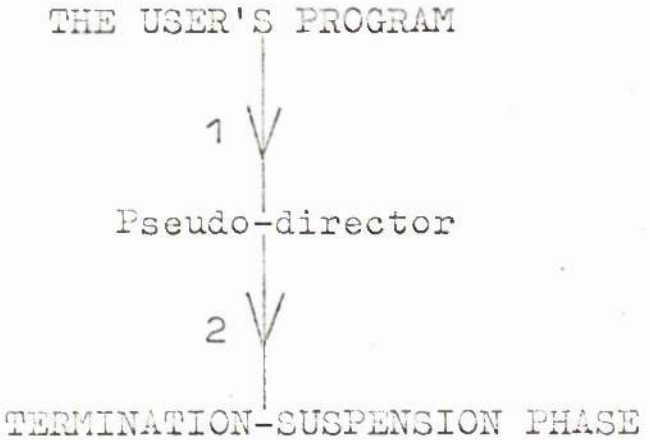



Figure 15

The Terminate Section

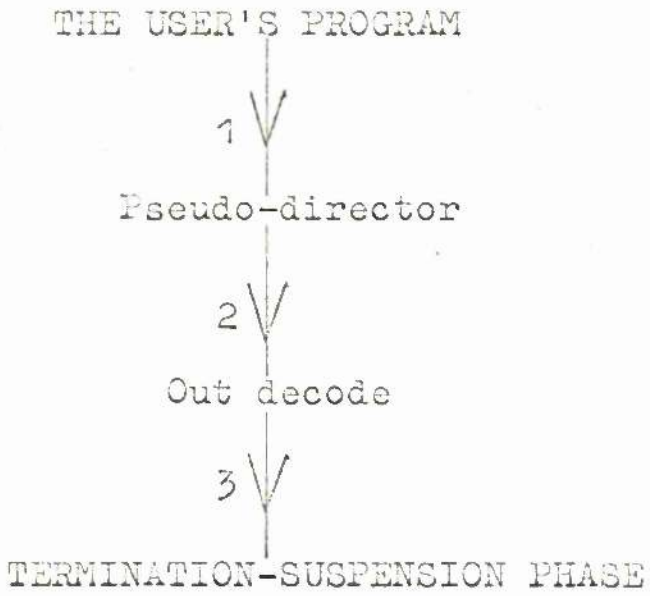



Figure 16

The Discontinue Section

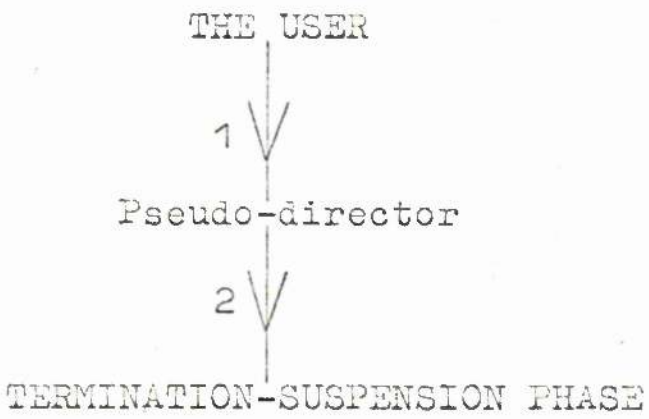



Figure 17

The Write Phase

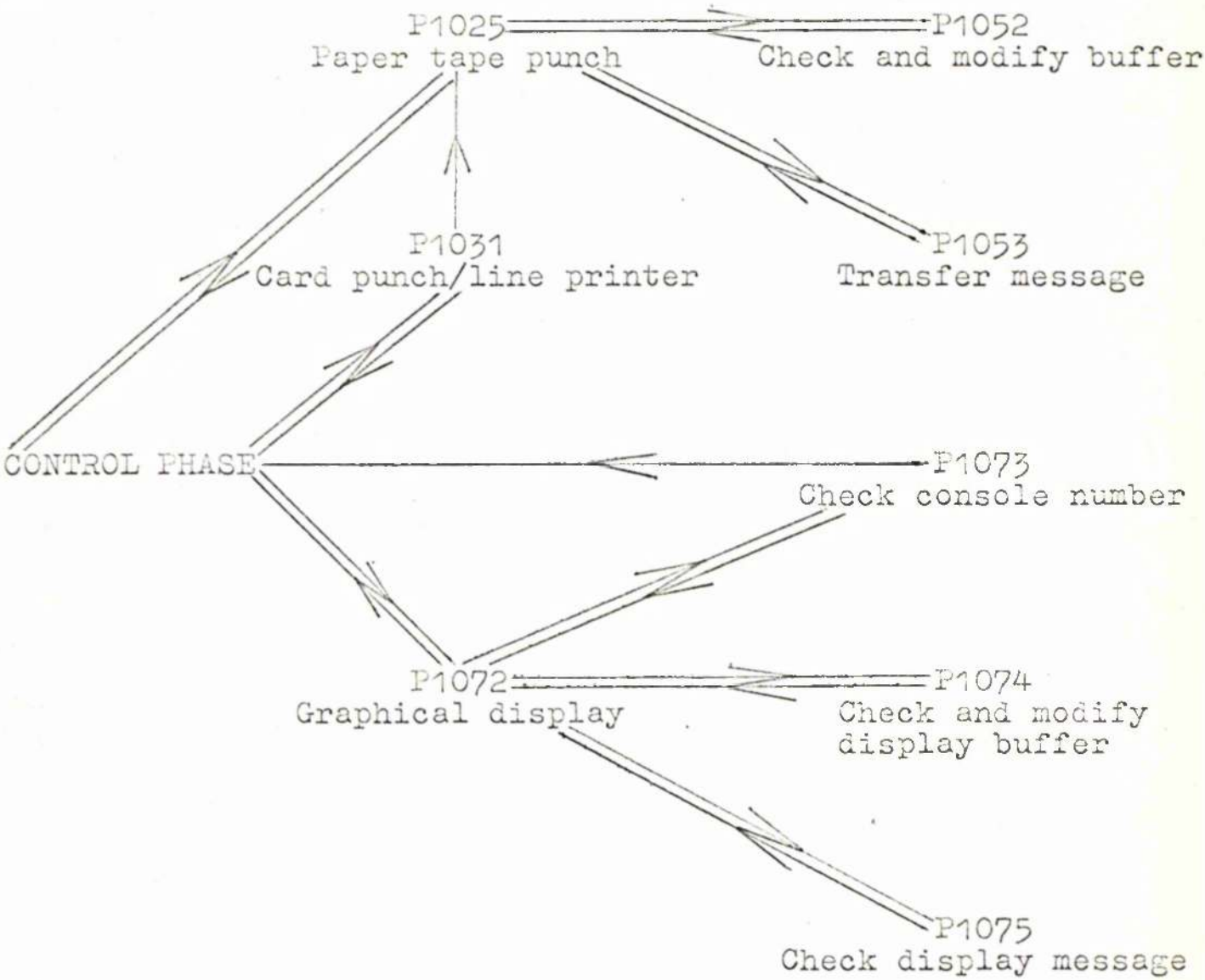



Figure 18

program does not itself transfer messages either to or from the peripheral devices. The action of this phase is to guarantee that the messages are in a legal format, insert them in the correct queue, permit the queue to be serviced, and finally ascertain that they have been dispatched.

4.3.3.1 Console Output Sections

Since output to the on-line console has been subdivided into partially overlapping sections, some general points concerned with it shall be discussed separately.

On-line system characteristics dictate that any message output to the console should be in paper-tape code and should not exceed forty words in length. The additional data area associated with the user program contains the buffers, each forty words in size. The first two are assigned as output buffers to allow double buffering of normal messages. However, there is another type of message known as a prompt.

When additional data is required the command program outputs a standard prompt to the console. Whereas the user program may contain more than one input statement, this prompt only indicates that data is required. It is therefore necessary for the user program to identify individual requests. This is accomplished by

preceeding each input statement by another statement which will output a tag message designated as a prompt. This message is held in the third buffer, the prompt buffer, overwriting any previous unissued prompt from the user program. Although normal messages are output immediately, a prompt message is held until data is requested. If and when this occurs it is output in preference to the command program prompt.

4.3.3.2 Paper-tape Punch Section (Figure 19)

The facility to handle messages intended for the paper-tape punch was implemented as this is the device normally output to by assembly language programs. Code conversion is unnecessary and a machine register is used to mark prompt messages. Since the user has full control of the output parameters, stricter checks are made for illegal conditions than are made for messages output by a high level language program which uses a standard package.

4.3.3.2.0 Commentary on the Routines

4.3.3.2.1 Paper-tape Punch

First entry is made to the subroutine which checks and modifies the program's buffers and then to the message transfer subroutine to

The Paper-Tape Punch Section

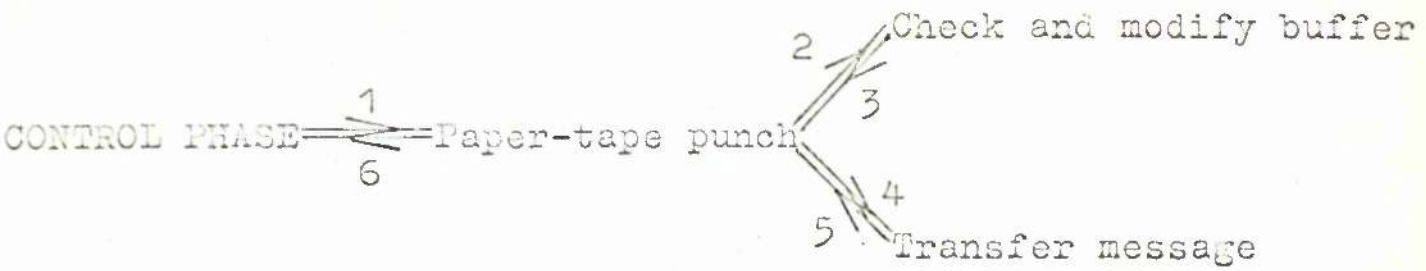



Figure 19

The Card Punch/Line Printer Section

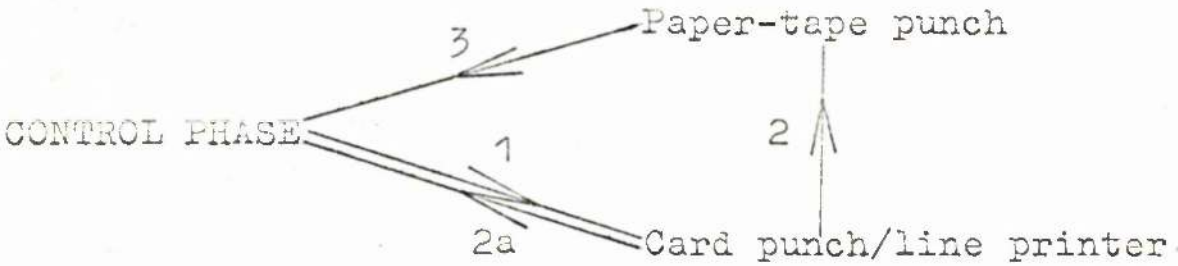



Figure 20

The Graphical Display Section

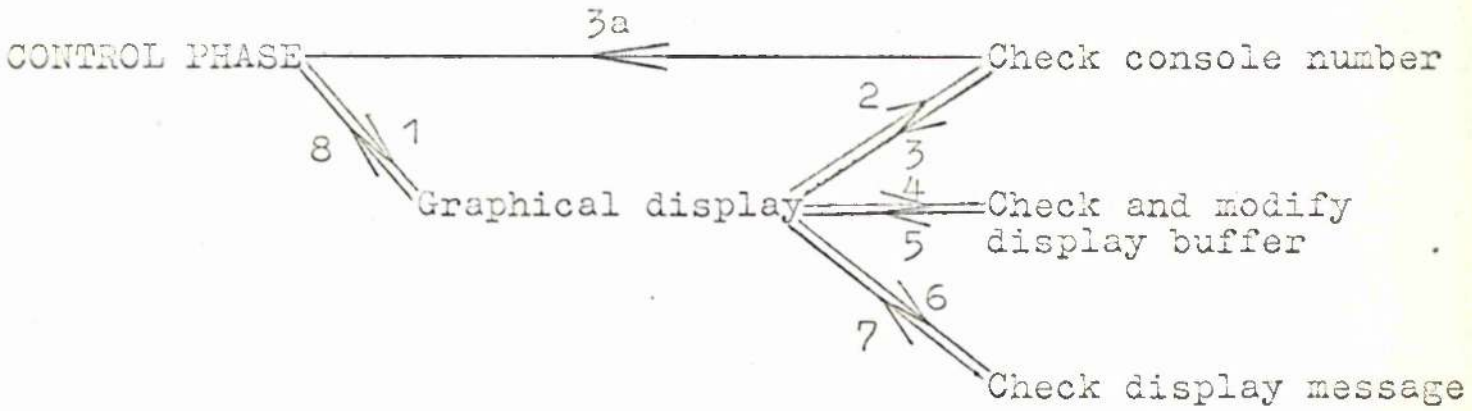



Figure 21

load a prompt into the prompt buffer or a normal message into the currently available output buffer. With normal messages control is passed to the on-line supervisor to initiate the transfer and the buffers are switched.

4.3.3.2.2 Check and Modify a Buffer

The size of the buffer and its upper and lower addresses are checked. These addresses, which are relative to the base address of the user program, are transformed into absolute machine addresses.

4.3.3.2.3 Transfer a Message

Given the parameters of two forty word buffers A and B, this routine transfers the contents of buffer A into buffer B until either a termination character is found in the information or buffer B is full.

4.3.3.3 Card Punch/Line Printer Section (Figure 20)

Facilities to handle messages intended for the card punch or the line printer were implemented as these are the devices normally output to by a high level language program. However, messages are

not normally output on the card punch and since computer readable output is not being produced, to avoid unnecessary duplication, punch output is ignored without penalising the user program. A message is designated a prompt by preceeding the printable string with a control character. Since the user has little control over the size of the message buffer, excessive output is truncated rather than treating it as an error condition. Truncation may also occur as the result of the necessary code conversion which tends to increase the length of line printer messages.

4.3.3.3.0 Commentary on the Routines

4.3.3.3.1 Card Punch/Line Printer

In the case of punch output control is returned directly to the control phase. If the user program issues a message for the line printer the buffer addresses are modified and the information, with code conversion and any necessary truncation, is transferred into either the prompt buffer or the currently available output buffer as specified. Control is returned to the control phase via the paper-tape punch section which will output normal messages and switch buffers.

4.3.3.4 Graphical Display Section (Figure 21)

Display messages produced by the user program are transferred to a message queue in the resident supervisor which previously had not been accessible by a command program. Since there is only one graphical display, to avoid confusion, display messages are only accepted from two consoles which are in the proximity of the display unit. Although in most cases display files will be generated automatically by the GHOST graphical output system, since assembly language programs may also use this facility, checks have to be made on both the message parameters and certain of the contents of the message which act as parameters at a later stage.

4.3.3.4.0 Commentary on the Routines

4.3.3.4.1 Graphical Display

Entry is made to the subroutines which check the console number, check and modify the message buffer, and check the parameters in the message. The new entry is not inserted in the queue until any previous entry has been transmitted. When the entry is inserted lockouts are set on its buffer. Control is not returned to the control phase until the message has been sent.

4.3.3.4.2 Check the Console Number

If the display message originates from a program reactivated from a remote terminal (not the KDF 9 monitor flexowriter or the PDP 8 monitor teletype) it is ignored without penalty and a warning message output to the console. Indication is also given if the message cannot be output because the display unit is not available.

4.3.3.4.3 Check and Modify the Display Buffer

The size of the buffer and its upper and lower addresses are checked. These addresses which are relative to the base address of the user program are transformed to absolute machine addresses. If this buffer has been used for a previous transfer and the lockouts are still set control will not proceed until they have been cleared.

4.3.3.4.4 Check Display Message

A display message consists of a display file preceeded by file parameters. This routine checks that the values of the parameters agree with the graphical system specification of them.

4.3.4 Read Phase (Figure 22)

If interaction is to be considered from the point of view of efficient use being made of the computer then this is the most important phase. Since it is assumed that the user's reaction to a request for data will be slow, suspension and interactive exit from the command program take place automatically. When all the required data has been input the command program is recalled and the user program reactivated. Thus only the time taken to dump and undump the user program and part of the background job is lost. This is in effect although time slicing, often under a different name, and in many systems. is not uncommon, either the impression is given that two or more programs are being processed simultaneously, which is blatently false, or the rather dubious assertion is made that the switching time is negligible. What can be said, however, is that when the user reduces the frequency of the interactions the effect of the interactive system on the background system is less conspicuous.

4.3.4.1 Commentary on the Routines

4.3.4.1.1 Paper-tape Read

Although entry from the control phase causes interactive

The Read Phase

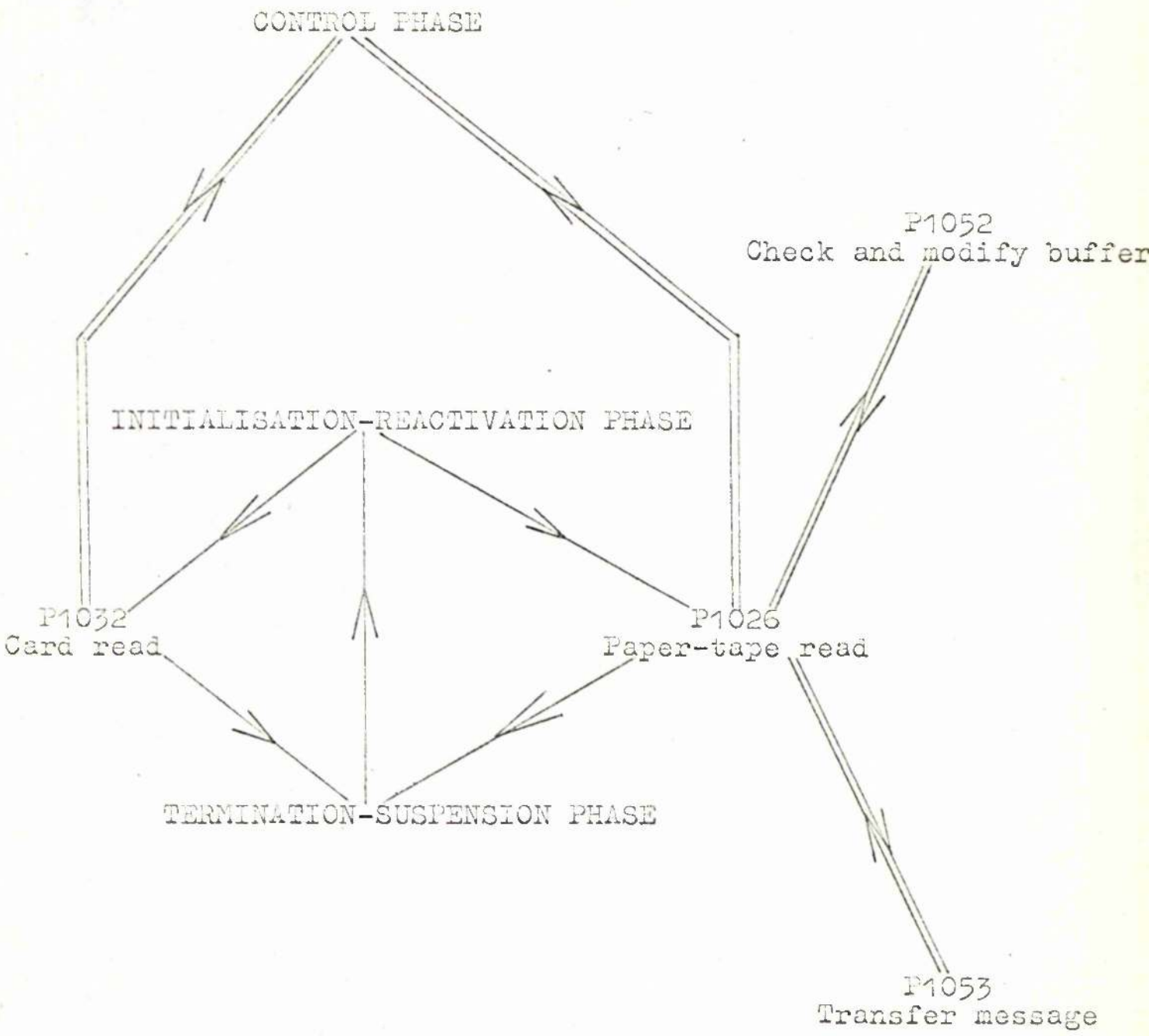



Figure 22

exit via the termination-suspension phase when the required data has been input control is returned by the initialisation-reactivation phase. The buffer of the user program is checked and modified and the data transferred into it. Before returning to the control phase the prompt buffer is cleared.

4.3.4.1.2 Card Read

The only significant difference between this routine and the paper-tape read routine is the necessity of code conversion. If the case of a character is considered an integral part of that character then conversion is straight forward as there is a one-to-one relationship between the different codes. Although there is direct conversion from paper-tape code to alpha-numeric code, since the binary code is rarely used, it is sufficient to provide for alpha-numeric to binary conversion.

4.3.4.2 Paper-tape Read Section (Figure 23)

Like the paper-tape punch section of the write phase, with which it shares the check and modify buffer and transfer message subroutines, this section provides a facility intended primarily for assembly language programs.

The Paper-Tape Read Section

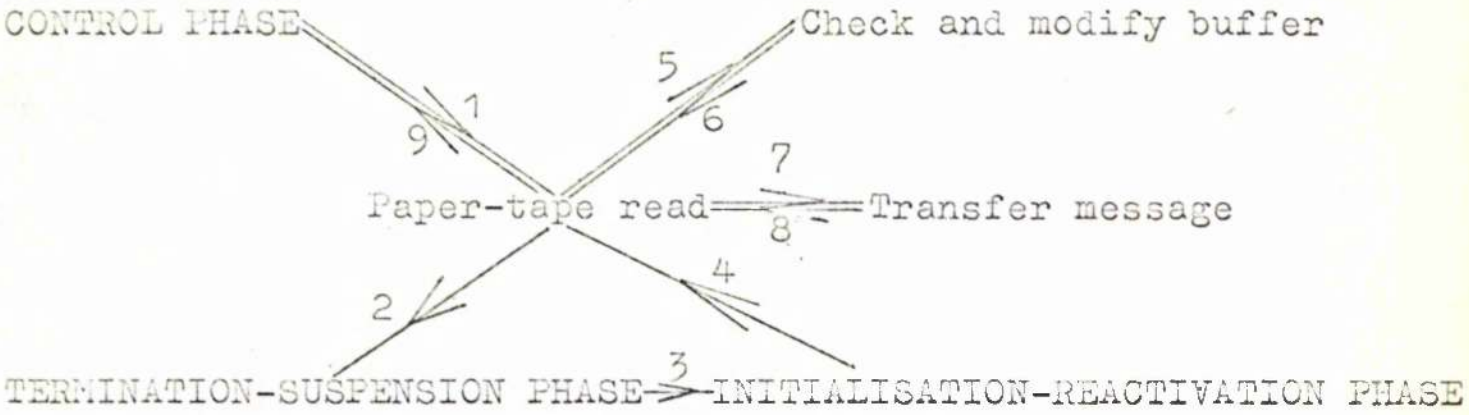



Figure 23

The Card Read Section

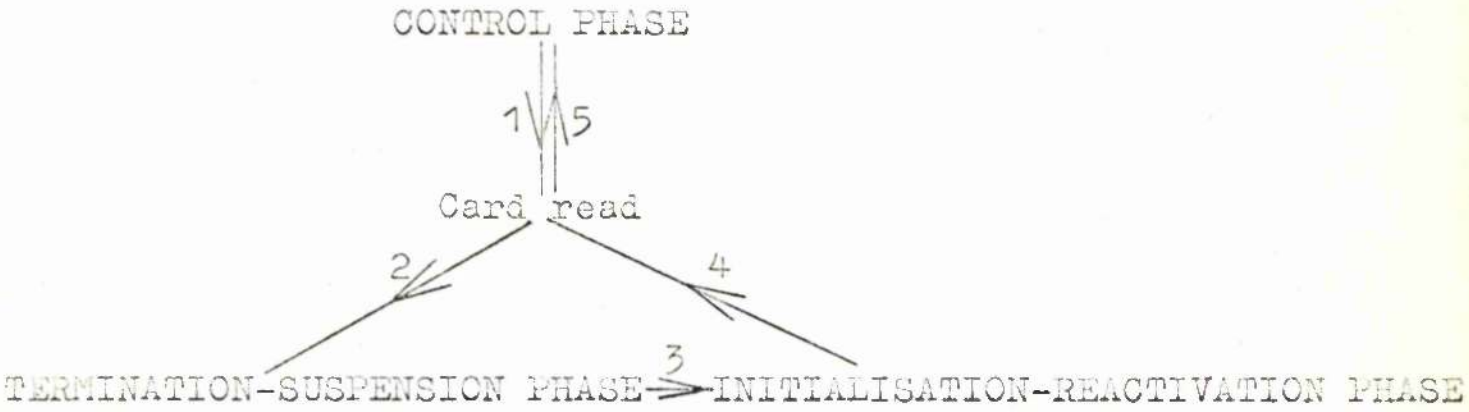



Figure 24

4.3.4.3 Card Read Section (Figure 24)

This section, providing an input facility for high level language programs, corresponds to the card punch/line printer section of the write phase. The user program may read cards as though the data was in fixed or free format and in alpha-numeric or binary code. In fact the preferred types are free format, which is the convention of the on-line system, and alpha-numeric code, which is the only code of the write phase.

4.3.5 Termination-Suspension Phase (Figure 25)

The title of this phase reflects its duality of function. The command program may be terminated upon detection of an illegal condition or upon the request of the user or the user program. A request by the user program causes immediate exit, the current version of the user program is not saved, and the system reverts to the command state. Alternatively, when a request is received which implies that additional data is required, the current version of the user program is written to disc while, on exit from the command program, the system remains in the data state.

4.3.5.1 Commentary on the Routines

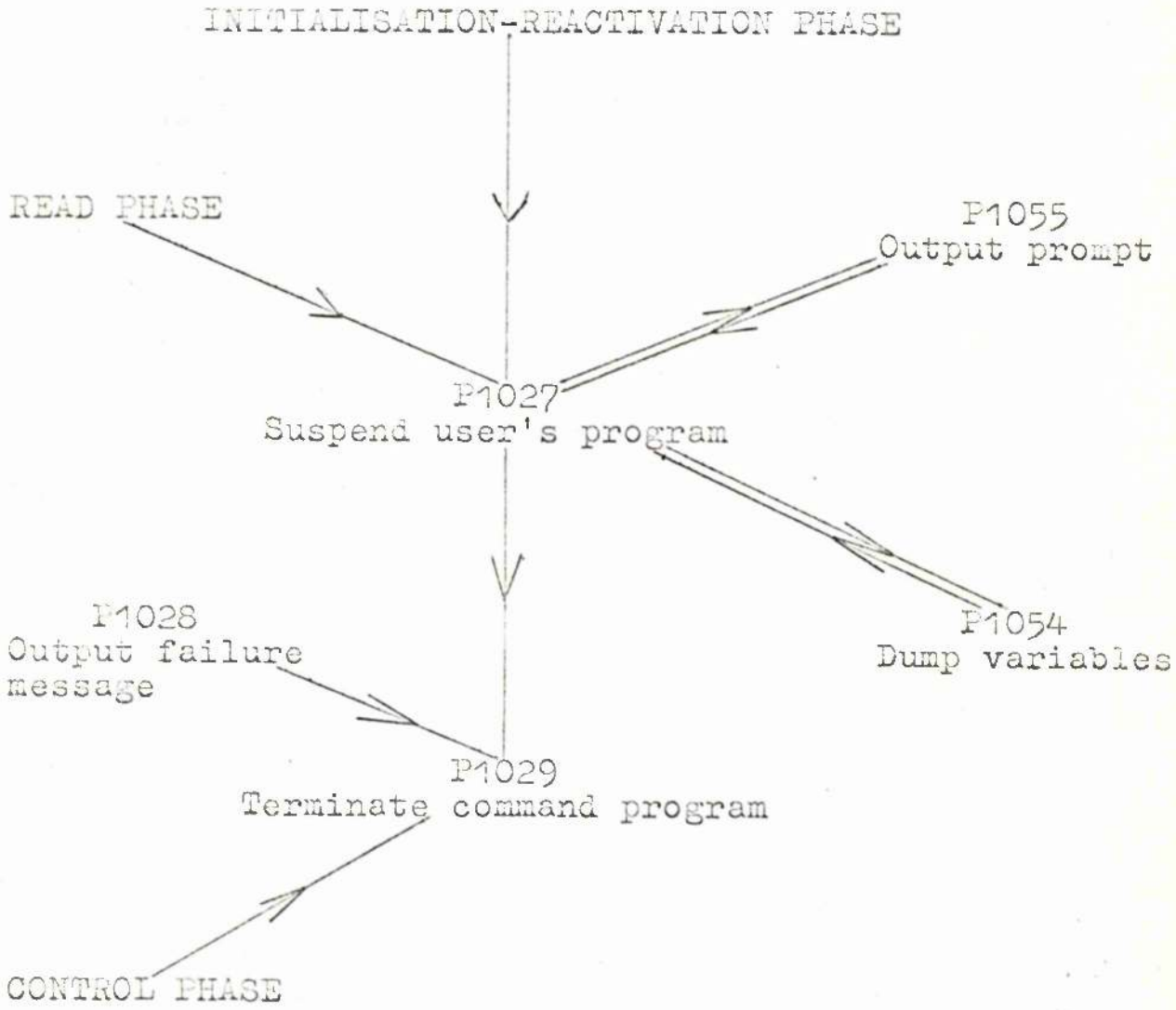

INITIALISATION-REACTIVATION PHASE
READ PHASE
P1055
Output prompt
P1027
Suspend user's program
P1028
Output failure
message
P1054
Dump variables
P1029
Terminate command program
CONTROL PHASE

Figure 25

4.3.5.1.1 Suspend User Program

Entry is first made to the subroutines which output the prompt and dump the variables. The previous current version of the user program, if any, is removed from the file nesting store and the new current version written into the top cell. The original version is always preserved in the second top cell.

4.3.5.1.2 Output Prompt

The last prompt given by the user program is output to the console. When such a prompt is not available the standard prompt of the command program is used.

4.3.5.1.3 Dump Variables.

Certain user program variables, normally held in machine registers but preserved in core locations by the command program after an interrupt, are stored in the additional data area.

4.3.5.1.4 Output Failure Message

The link address of the user program (WWWWW/S), the number

of the command program routine (RRRR) in which the error test was made, and the number of the test (TTT) within this routine are inserted in the failure message

PROGRAM FAILURE RRRR/TTT

LINK - WWWW/S

which is output to the console.

4.3.5.1.5 Terminate Command Program

The exit number, which may indicate normal or interactive exit, is set. The return address is fetched and control returned to the on-line supervisor.

4.3.5.2 Interact and Suspend Sections (Figures 26 and 27)

These sections have been separated, not because of any difference in their action, but because of differences in the reasons for entering them. If the command RESUME was used to call the command program or automatic restart is being attempted then the suspend section is entered. However, if suspension is only a direct result of a request for additional data by the user

The Interact Section

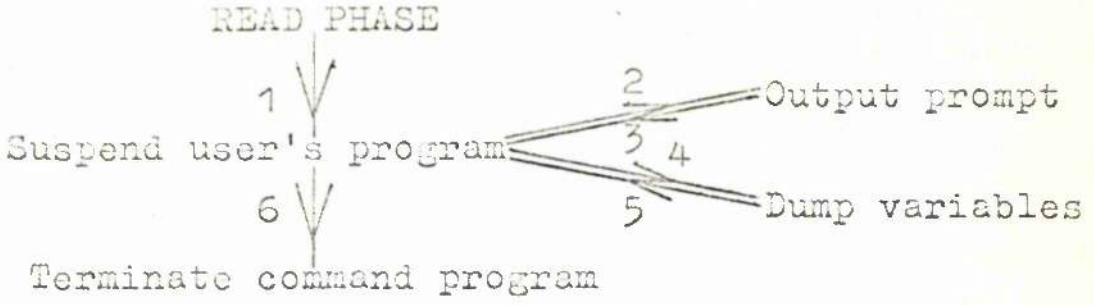



Figure 26

The Suspend Section

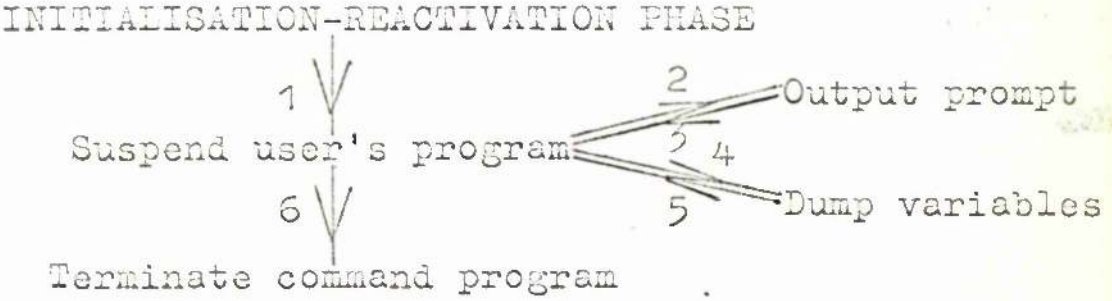



Figure 27

The Failure Section


Figure 28

program then it is referred to as the interact section.

In either case the action of the section is to output the previous prompt, write the user program to disc and cause interactive exit from the command program allowing the user to input the required data.

4.3.5.3 Failure Section (Figure 28)

This section can be entered from any phase without any restriction. Failure messages output by the command program are intended to pinpoint the position at which an illegal condition was detected rather than to explain the cause of failure. The information output consists of the link address of the user program, the number of the command program routine in which the error test was made, and the number of the test within this routine. To obtain the cause of failure the user may utilise the routine and test numbers to access a corresponding explanatory message from a list held in a public file. The principal reason for excluding explanatory messages from the command program was to limit its storage requirements. Such messages would not only require storage space but also additional coding to handle them. Once the command program exceeds a certain size it competes with

the user program for core. Thus increasing the size of the command program decreases the area available for the user program. Since a large number of illegal conditions are tested for this policy has resulted in a substantial saving in space.

Another reason for such a policy is that in most circumstances there is no immediate correction which the user can make in order to continue execution. Therefore the time required to access the explanatory messages is not very important and in no case critical. Similar logic determines that normal exit to the online-supervisor should occur after failure since everything that can be has been stacked in the file nesting store and is available if required.

## 4.4 Development and Testing

The interactive system command program unlike the JOB command program could not be developed or tested with the aid of the supervisor simulator. Since many parts are only activated after an interrupt is trapped and instructions are used which are only legal in supervisor mode the command program could not be run in program mode. Instead the complete machine had to be reserved for development and testing runs. In the event of failure the most reliable of diagnostics was the information from machine state

indicators and core dumps. In consequence a larger amount of real and machine time were consumed than were required for a standard command program.

## 5. Conclusions

The power of a multi-access system should lie in the comprehensiveness of its data handling facilities where both 'data' and 'handling' are given their widest interpretation. It should be comprehensive in the sense that it embraces a great variety of facilities which are brought together to form an integrated unit. It follows that each facility gains strength and meaning from its association with the others. However, when the facilities are associated by chaining them so that the results of one is the data of another it should be remembered that a chain is only as strong as its weakest link.

The remote job entry facility forms a link in a chain for program development and must be evaluated as part of this chain as well as upon its own merits. It is preceded by a formidable array of character string and file manipulation facilities which are directed to the task of composing the textual form of the program and followed by the facility which allows diagnostic messages, relating to the compilation and execution of the program, to be returned to the user. Within this context its function, to introduce programs into the job execution stream, is clearly defined and has been fully implemented. In addition it has been expanded

and generalised as far as would appear to be purposful and profitable.

Implementation of the interactive system can be justified if the system provides facilities, not otherwise available, which, when used to solve a problem, consumes less processor time than would have been expended had the problem been solved by more orthodox methods. Consider the class of problems in which a mathematical model is constructed in order to simulate a physical system. At some point in the construction of the model it may be necessary to make adjustments using speculative or empirical procedures. Where such procedures require a large number of short bursts of processor time, because the interactive system is so responsive to the user, more efficient use should be made of the processor. This gain in efficiency results from the avoidance of unnecessary repetition. When processor time is available in a small number of long bursts the strategy which covers the greatest number of possibilities is the most promising. As the number of bursts increases and their length decreases it becomes more favourable to select the most probable continuation thus avoiding many of the redundances of the previous strategy. The display may prove an invaluable aid in this selection since it is often easier to make an estimation or decision from a graphical rather than a numerical representation of results. It is thus apparent that the

interactive system may prove a worthwhile experimental tool for the research worker.

## REFERENCES


1. KDF 9 Programming Manual (English Electric Computers)

2. KDF 9 Non-Time-Sharing Director Support Documentation (English Electric Computers)

3. KDF 9 EGDON3 Reference Manual Volume 1 (English Electric Computers)

4. Poole, P.C. and Lang, T. The Development of On-Line Computing Facilities for the KDF 9. Part 1 : COSEC - A Single On-Line Console. (The Computer Journal Vol.11, pp. 5-11)

5. Poole, P.C. Some Aspects of the EGDON3 Operating System for the KDF 9 (IFIP Congress 68 pp. c43-c47)

6. A User's Guide to COTAN. (U.K.A.E.A. Culham Laboratory).

7. Larkin, F.M. A Graphical Output Language and its Implementation. (U.K.A.E.A. Culham Laboratory).

8. Larkin, F.M. User's Guide to the Culham Graphical Output System (U.K.A.E.A. Culham Laboratory).

```
*CHAIN25
*IDENTIFIERP1CH25
*USERCODE
 ROUTINE
 P1,V3900*
 END,
*IDENTIFIERP0CH25
*USERCODE
 ROUTINE
 P0,V1220*
       V0=00/AP4000/AV0P0,(LIMIT OF SEGMENT),
       V1=11,(COMMAND PROGRAM NUMBER),
       V2=00/1280/21,(0/EXTERNAL WORK SPACE/POSITION IN INDEX),
       V3=B52 57 42,(JOB),
       V4=02/0/0,(PROMPTS ON DISC),
       V5=01/AV7/AV15,
       V6=0,(TERMINATOR),
       V7=4,(NO OF PROMPTS),
       V8=1,
       V9=B0206 6471 6045 0734,(PROMPT--TYPE),
       V10=1,
       V11=B0206 4651 5445 0734,(FILE),
       V12=1,
       V13=B0206 5257 4277 0734,(JOB CODE),
       V14=1,
       V15=B0206 6451 5545 0734,(TIME),
       SETAV0,JP8102,
 END,
*SUBSTITUTE FILL01
*SUBSTITUTE NFILL1
*SUBSTITUTE FILL00
*SUBSTITUTE NFILL0
*IDENTIFIERP4000
*USERCODE
 ROUTINE
 P4000*
```

```
        (FIRST ROUTINE OF THE JOB SEGMENT),
        JP4100, (INITIALISE THE SEGMENT),
 END,
*SUBSTITUTECORSAV
*SUBSTITUTECOMMUN
*USERCODE
 ROUTINE
 L4000,V44*
        V 0=0, (EXIT NUMBER),
        V 1=0, (RETURN ADDRESS),
        V 2=B 37 37 37 37 00 00 00 00, (....     ),
        V 3=B 17 17 17 17 00 00 00 00, (////     ),
        V 4=0, (0-REMOTE/1-BACKGROUND),
        V 5=0, (TIME LIMIT IN BINARY),
        V 6=0, (FIRST WORD OF DATA),
        V 7=0, (SECOND WORD OF FILE NAME),
        V 8=0, (FILE VERSION NUMBER),
        V 9=0, (UNPACK FILENAME MARKER),
        V10=B 45 56 44 61 65 45 65 45, (ENDQUEUE),
        V11=B 15 45 56 44 42 54 57 43, (*ENDBLOC),
        V12=B 62 45 55 57 64 52 57 42, (REMOTJOB),
        V13=B 42 41 43 53 47 62 56 44, (BACKGRND),
        V14=B 02 75 77 77 77 77 77 77,
        V15=0, (L/S OF THE QUEUE FILE),
        V16=B 00 00 17 00 00 17 00 00, (  /  /  ),
        V17=0, (L/S OF USERS DIRECTORY FILE),
        V18=B 00 41 62 43 50 51 66 45, ( ARCHIVE),
        V19=B 00 62 45 63 64 57 62 45, ( RESTORE),
        V20=0, (QUIET MARKER),
        V21=B 15 52 57 42 00 00 00 00, (*JOB     ),
        V22=B 43 57 56 62 45 55 00 17, (CONREM /),
        V23=B 00 00 00 00 00 00 17 17, (JOBNUM//),
        V24=0, (ONLINE NAME),
        V25=B 17 00 00 00 00 00 00 15, (TIME LIMIT),
        V26=0, (CONSOLE NUMBER),
        V27=0,
        V28=0, (FIRST WORD OF FILE NAME),
        V29=0, (SECOND WORD OF FILE NAME),
```

```
        V30=0,
        V31=0, (FIRST WORD OF AUXILIARY CARD LEFT BLANK),
        V32=0,
        V33=0,
        V34=0,
        V35=0,
        V36=0,
        V37=0, (DATE),
        V38=0,
        V39=8 20 20 37 20 20 37 20 20, (TIME),
        V40=0,
        V41=0, (NUMBER OF WORDS OF EXTERNAL WORK SPACE),
        V42=0, (BA OF EXTERNAL WORK SPACE),
        V43=0, (0/LA/UA OF EXTERNAL WORK SPACE),
        V44=0, (NUMBER OF ENTRIES/20/BA OF FILE BODY),
 END,
*USERCODE
 ROUTINE
 P4100*
        (INITIALISE THE SYSTEM),
        JSP3611, (INITIALISE THE COMMUNICATION AREA),
        ERASE, (ENTRY NUMBER),
        =V1L4000, (RETURN ADDRESS),
        V3L3611, =M5, SETAV0P0, =M0M5, (DUMP AREA FOR THE SUPERVISOR),
        V0L3611, =M5, E15M5, SHL32, J50NEZ, (NOT FROM A CONSOLE),
        M0M5, (ONLINE NAME),
        DUP, =V24L4000, J51=Z, (NO ONE LOGGED IN),
        F3M5, SHL46, =V20L4000, (SET QUIET MARKER),
        V4L3611, =M5, M0M5, JS1P4001, (CONSOLE NUMBER),
        SHL36, SHC18, =V26L4000, (TO JOB CARD),
        JS1P3600, (TIME IN SECONDS TO 24 INTEGRAL PLACES),
        SHL-24, SET60, /I, JS1P4001, (SECONDS),
        REV, SET60, /J, JS1P4001, (MINUTES),
        SHL18, REV, JS1P4001, SHL36, (HOURS),
        OR, OR, V39L4000, OR, =V39L4000, (TO AUXILIARY CARD),
        V0L3612, =M5, M0M5, =V37L4000, (DATE TO AUXILIARY CARD),
        JP4210, (UNPACK AND CHECK THE TYPE),
 50, SETAV74L4444, SET6, JSP5050, (NOT FROM A CONSOLE),
```

```
        ERASE, J3P4444, (EXIT 0),
    51, SETAV0L4444, SET3, DUP, =V0L4000, (EXIT 3),
        JSP5050, ERASE, J2P4444, (NO ONE IS LOGGED IN),
  END,
 *USERCODE
  ROUTINE
  P4210,R50*
        (UNPACK AND CHECK THE TYPE),
        JSP4201, (UNPACK),
        J50, (TOO LITTLE DATA),
        V6L4000, DUP, J51=Z, (TYPE BLANK),
        ZERO,
     1, SHLD6, DUP, J1=Z, (FIND FIRST NON-ZERO CHARACTER),
        REV, ERASE, SETB62, J2=, (REMOTE),
        SETB42, J51NE, (NOT BACK),
        SET1, =V4L4000, (BACKGROUND),
     2, ERASE, V1L3612, =M5, (AV0L4989),
        E7M5, =M5, (AV0L4987),
        E2M5, SHC16, =05, (M5-AV0L1022),
        C5, =M6, (AV0L6902),
        V4L4000, =+M5, (ALLOW FOR QUEUE TYPE),
        E34M5, J3NEZ, (THE QUEUE IS BEING SERVICED),
        SETAV6L4444, SET7, JSP4050, ZERO, J5,
     3, E6M6, J4NEZ, (BACKGROUND SYSTEM NOT PAUSED),
        SETAV13L4444, SET9, JSP4050,
     4, V5L3612, =M5, E7M5, E2M5, -, DUP, J5LEZ,
        SHA-24, SET60, /I, ERASE, DUP, JS2P4001,
        REV, SET160, -, J5GTZ, (JOB TERMINATED),
        SHL30, V61L4444, SHC-12, SHL-18, SHLD18, SHC12, =V61L4444,
     5, ERASE, SETAV56L4444, SET7, JSP4050,
        ZERO, SETAV12L4000, =M5, V4L4000, =M6, M5M6, (QUEUE FILE NAME),
        V1L3612, =M5, E8M5, (L/S OF SYS DIRECTORY FILE),
        JSP4002, (FIND SECTOR ADDRESS OF QUEUE FILE),
        SET109, SHL24, 0, DUP, =V15L4000, (SAVE),
        V1L3612, SET3, +, V4L4000, +, =M5, M0M5, (QUEUE CONTROL WORD),
        =07, ZERO, NOT, =I7, (SET FILE BUSY),
        DUP, SET4, +, =M7, (SET SECTOR ADDRESS OF QUEUE FILE BODY),
        07, =M0M5, (SET CONTROL WORD),
```

```
      V0L4888, JSP5035, (READ 1ST SECTOR OF FILE HEAD),
      V5L4888, JSP4001, DUP, (NO OF SECTORS OCCUPIED BY BODY),
      V6L4888, JSP4001, NEV, (NO OF SECTORS RESERVED FOR BODY),
    DUP, J6=Z, (ALL SECTORS OF BODY OCCUPIED),
      ERASE, SET40, (ALLOW FOR INCREASE OF OCCUPIED SECTORS),
  6, DUP, CAB, SET4, +, (FILE HEAD + OCCUPIED SECTORS),
      DUP, SHL2, +, SHL3, DUP, (NO OF WORDS TO BE READ DOWN),
      CAB, +, DUP, =V41L4000, (NO OF WORDS OF EXTERNAL SPACE),
    DUP, SET20, DUP, =I6, /I, ERASE, SET8, -, =C6,
      JSP5029, DUP, JS2LTZ, (CLAIM WORK SPACE AND CHECK),
      DUP, =V42L4000, (BA OF EXTERNAL WORK SPACE),
      =R15, M+I5, M5T006, NE6, NOT, =+M5, (Q5=0/LA),
      V15L4000, Q5, JSP5035, (READ FILE HEAD+OCCUPIED SECTORS),
    DC5, NC5, Q5, +, =V43L4000, (1/LA/UA OF EXT WORK AREA),
      SET160, =+M6, Q6, =V44L4000, (MAX NO OF JOBS IN BODY/20/BA),
      V41L4000, =RC5, M6TOQ5,
   7, M0M6, V2L4000, NEV, J8NEZ, (NOT A .... CARD),
      M0M60, ERASE, J7C6NZ, (TEST NEXT JOB CARD),
   8, C6, DUP, SHL2, +, SHL2, =C6, I6=+1, (NO/1/),
   9, J10C6Z, M0M60, =M0M50, J9, (OVERWRITE GARBAGE),
  10, J11C5Z, ZERO, =M0M50, J10, (ZERO END OF FILE),
  11, JP4220, (UNPACK AND CHECK THE FILE NAME),
  50, SETAV29L4444, SET3, JP4444, (DATA INCOMPLETE),
  51, ERASE, SETAV3L4444, SET3, JP4444, (TYPE NOT RECOGNISED),
 52, ERASE, ERASE, ERASE,
      SETB17202S, J1P4444, (NOT ENOUGH EXTERNAL WORK SPACE),
 END,
*USERCODE
 ROUTINE
 P4220*
      (UNPACK AND CHECK THE FILE NAME),
      ZERO, NOT, =V9L4000, (SET THE UNPACK FILE NAME MARKER),
      JS1P4201, (UNPACK),
      JS0P4210, (TOO LITTLE DATA),
      ZERO, V7L4000, J1NEZ, (NAME GREATER THAN 8 CHARACTERS),
      V6L4000, JS3, (LEFT JUSTIFY FIRST WORD OF FILE NAME),
      =V4L4000, J2, (SAVE FIRST WORD),
   1, V7L4000, JS3, (LEFT JUSTIFY SECOND WORD OF FILE NAME),
```

```
      =V7L4000, (SAVE SECOND WORD),
    2, V8L4000, SHL24, SHC24, (FILE VERSION NUMBER),
      V7L4000, OR, DUP, =V29L4000, (FILE NAME),
      V6L4000, DUP, =V28L4000, (TO JOB CARD),
      V0L3611, =M5, E4M5, DUP, =V17L4000, (L/S OF USRES DIRECTORY FILE)
      SET2, +, JSP4002, (FIND L/S OF USERS FILE),
      ERASE, JP4230, (UNPACK AND CHECK THE JOB CODE),
    3, SHLD-6, DUP, J3NEZ, (LEFT JUSTIFY),
      ERASE, EXIT 1,
 END,
*USERCODE
 ROUTINE
 P4230*
      (UNPACK AND CHECK THE JOB NUMBER),
      JS1P4201, (UNPACK),
      J50P4210, (TOO LITTLE DATA),
      V17L4000, NOT, NEG, V0L4888, JSP5035, (READ THE SECOND SECTOR),
      V4L4000, DUP, SHL2, +, SHL1, (0-REMOTE/10-BACKGROUND),
      SETAV3L4888, +, =M5, M0M5, (MAXIMUM ELAPSE TIME),
      JSP4001, DUP, SET999, (MAXIMUM TIME ON JOB CARD),
      MAX, ERASE, =V5L4000, (SAVE MINIMUM),
      JS2P4001, SHL6, ZERO, SHLD12,
      V26L4444, OR, =V26L4444, V27L4444, OR, =V27L4444,
      SETAV22L4444, SET7, JSP4050, (MAXIMUM TIME MESSAGE),
      V6L4000, DUP, SHL12, V23L4000, OR, =V23L4000, (TO CARD),
      ZERO, REV,
    1, SHLD-6, DUP, J1NEZ, (LEFT JUSTIFY),
      ERASE, SHC24, SHL24, (FIRST FOUR CHARACTERS),
      V44L4888, =05, (8/1/8A OF JOB CODE CARD),
  *,2, M0M50, J3=, *, J2C5NZS, (COMPARE JOB CODES),
      ERASE, SETAV39L4444, SET3, JP4444, (JOB NUMBER NOT FOUND),
    3, ERASE, JP4240, (UNPACK AND CHECK THE TIME),
 END,
*USERCODE
 ROUTINE
 P4240*
      (UNPACK AND CHECK THE TIME),
      JS1P4201, (UNPACK),
```

```
      J1,
      J50, (TOO MUCH DATA),
    1, V6L4000, DUP, SHL18, V25L4000, (REQUESTED TIME),
      SHC6, SHL36, SHC6, OR, =V25L4000, (TO JOB CARD),
      DUP, J52=Z, (BLANK TIME),
      DUP, ZERO,
    2, ERASE, ZERO, SHLD6, DUP, J2=Z, (1ST NON-ZERO CHARACTER),
      SETB55, J3NE, (NOT M-MAXIMUM),
      ERASE, ERASE, ERASE, (START AGAIN),
      V5L4000, JS1P4001, =V6L4000, J1, (WITH MAXIMUM TIME),
    3, REV, ERASE, SETB62, J5=, (R-REMOVE),
      SETB63, NEV, J6=Z, (S-STATUS),
      DUP, ZERO, SET8, =C5, (CHECK ALL CHARACTERS),
    4, ERASE, ZERO, SHLD6, DC5, DUP, J4=Z, (BLANK CHARACTER),
      DUP, SETB31, -, J51GTZ, (NOT A NUMERAL),
      DUP, SETB20, -, J51LTZ, (NOT A NUMERAL),
      J4C5NZ, (MORE CHARACTERS TO CHECK),
      ERASE, ERASE, JSP4001, (REQUESTED TIME TO BINARY),
      ZERO, J52=, (ZERO TIME REQUESTED),
      V5L4000, -, J53GTZ, (REQUEST GT LIMIT),
      JP4300, (ADD A NEW JOB TO THE QUEUE),
    5, ERASE, ERASE, JP4310, (REMOVE A JOB FROM THE QUEUE),
    6, ERASE, JP4320, (PROVIDE CURRENT STATUS OF USERS JOB),
   50, SETAV36L4000, SET3, JP4444, (TOO MUCH DATA),
   51, ERASE, ERASE, (NOT A NUMERAL),
   52, ERASE, (ZERO, TIME REQUESTED),
   53, SETAV32L4444, SET4, JP4444, (REQUEST GT LIMIT),
  END,
*USERCODE
 ROUTINE
 P4201,R1*
      (UNPACK DATA),
      SET41, =RC15, V1L3611, =M15, (BA OF DATA BUFFER),
      ZERO, =RC13, (NO OF CHARS LEFT FROM LAST FETCH),
    1, ZERO, =V6L4000, ZERO, =V7L4000, ZERO, =V8L4000,
      SET8, =RC14, SETAV6L4000, =M14, (STORE CONTROL),
      ZERO, =M13, J2C13Z, (NO RESIDUAL DATA),
      Q12, J3, (SET RESIDUAL DATA),
```

```
   2, SET8, =C13, MØM15Ø, J5ØC15Z, (BUFFER EMPTY),
   3, ZERO, SHLD6, DC13, (FETCH CHARACTER),
      DUP, SETB72, -, J8GTZ, (GT LETTER),
      DUP, SETB4Ø, -, J4GTZ, (LETTER),
      DUP, SETB31, -, J8GTZ, (GT NUMBER),
      DUP, SETB2Ø, -, J7LTZ, (LT NUMBER),
   4, M13M14, SHL6, OR, =M13M14, (STORE CHARACTER),
      DC14, J6C14NZ, (STORE WORD NOT FULL),
      V9L4ØØØ, J6=Z, (FILE NAME MARKER NOT SET),
      ZERO, =V9L4ØØØ, SET1, (UNPACK WORD 2 OF FILE NAME),
   5, SET4, =C14, =M13, (UNPACK UP TO 4 MORE CHARACTERS),
   6, J3C13NZ, (DATA WORD NOT EMPTY),
      ERASE, J2, (FETCH NEXT DATA WORD),
   7, SETB17, J8NE, (NOT VERSION NUMBER SEPARATOR),
      ERASE, SET2, J5, (UNPACK THE VERSION NUMBER),
   8, SETBØ1, J9=, (COMMAND MARKER),
      SETB75, J9=, (END OF DATA MARKER),
      SET2, NEV, J6NEZ, (NOT THE END OF THE ITEM),
      =Q12, EXIT 2, (SAVE RESIDUAL DATA , MORE DATA),
   9, ERASE, ERASE, EXIT 1, (ERASE RESIDUAL , NO MORE DATA),
  5Ø, ERASE, SETB172Ø21, J1P4444,
END,
*USERCODE
ROUTINE
P43ØØ*
      (ADD A NEW JOB TO THE QUEUE),
      V25L4ØØØ, SHL6, SHL-24, JSP4ØØ1, =RM5, (TIME TO BINARY),
      V43L4ØØØ, =Q6, M6TOQ14, (QUEUE PARAMETERS),
      SET-39, =+M6, MØM6, V21L4ØØØ, J5Ø=, (QUEUE FULL),
      V44L4ØØØ, =Q6, (QUEUE PARAMETERS),
   1, ERASE, MØM6Q, V3L4ØØØ, J1=, (////),
      V21L4ØØØ, NEV, J2NEZ, (NOT *JOB),
      M-I6, E4M6, SHL6, SHL-24, JSP4ØØ1, (ENTRY TIME TO BINARY),
      M5, -, J2GTZ, (ENTRY JOB TIME GT NEW JOB TIME),
      M+I6, ZERO, J1, (NEXT ENTRY),
   2, C6, NOT, NEG, DUP, SHL2, +, SHL2, =C14, C14TOQ15,
      I14=-1, I14TOQ15, M14TOQ15,
      SET-2Ø, DUP, =+C15, =+M14, J4C15Z, (INSERT AT END OF QUEUE),
```

```
*,3, M0M140, =M0M150, *, J3C15NZS, (SPACE FOR NEW ENTRY),
   4, SETAV21L4000, =RM15, I14=+1, M+I14,
 *,5, M0M150, =M0M140, *, J5C14NZS, (INSERT NEW JOB),
      JP4320, (PROVIDE CURRENT STATUS OF USERS JOB),
  50, ERASE, SETAV53L4444, SET3, JP4444,
 END,
*USERCODE
 ROUTINE
 P4310*
      (REMOVE A JOB FROM THE QUEUE),
      V44L4000, =Q5, (PARAMETERS TO FETCH WORDS 1),
 *,1, M0M5Q, V21L4000, NEV, *, J2=Z, J1C5NZS, J5, (*JOB),
   2, M5, =RM6, SET-13, =+M6, (PARAMS TO FETCH FILE NAME),
      M0M6Q, V28L4000, NEV, J1NEZ, (FIRST WORDS NOT SAME),
      M0M6, V29L4000, NEV, J1NEZ, (SECOND WORDS NOT SAME),
      C5, SHL2, DUP, SHL1, DUP, +, +, =C5, I5=+1, (TIMES 20),
      C5TOQ6, SET20, =+C6, SET-8, =+M6,
 *,3, M0M5Q, =M0M6Q, *, J3C5NZS, (OVERWRITE ENTRY),
 *,4, ZERO, =M0M6Q, *, J4C6NZS, (ZERO END OF QUEUE),
   5, JP4320, (PROVIDE CURRENT STATUS OF USERS JOB),
 END,
*USERCODE
 ROUTINE
 P4320*
      (PROVIDE CURRENT STATUS OF THE USERS JOB),
      V44L4000, =Q5, Q5TOQ6, J2, (WORD 1 OF ENTRY 1),
   1, ERASE, (//// OR *JOB),
   2, M0M5Q, V3L4000, J1=, (////),
      V21L4000, J1=, (*JOB),
      ERASE, SET10, =RC7, M-I5, M5TOQ7,
      V10L4000, =M0M7Q, (INSERT ENDQUEUE),
 *,3, ZERO, =M0M7Q, *, J3C7NZS, (ZERO REST OF CARD),
      V11L4000, =M0M7, (INSERT *ENDBLOC),
      C6, C5, -, DUP, =C8, (NUMBER OF ENTRIES OCCUPIED+1),
      SET2, /I, +, JS1P4001, (NUMBER OF SECTORS OCCUPIED),
      SHL12, SHL-12, V42L4000, =M5, =E4M5, (TO FILE HEAD),
      E8M5, SHC-6, SETB42, (FILE BODY ALTERED SINCE LAST PRIME),
      OR, SHC6, =E8M5, (TO CARD 1 OF FILE HEAD),
```

```
 E50M5, =E10M5, (NAME OF LAST USER TO CARD 2 OF FILE HEAD),
ZERO, =E20M5, ZERO, =E21M5, (CLEAR FILE HEAD SUMCHECK),
ZERO, =E22M5, ZERO, =E23M5, (CLEAR FILE BODY SUMCHECK),
 ZERO, V39L4000, SHL-18, SHLD-12, (TIME FROM AUXILIARY CARD),
 SHL-6, SHLD-12, ERASE, DUP, (SELECT HOURS AND MINUTES),
 E57M5, SHL-30, SHLD24, SHL6, (TIME OF READING),
 DUP, =E17M5, =E57M5, ERASE, (TO CARDS 2 AND 6 OF FILE HEAD),
 E51M5, SHL-30, SHLD24, SHL6, (TIME OF OVERWRITING),
 DUP, =E11M5, =E51M5, (TO CARDS 2 AND 6 OF FILE HEAD),
 ERASE, V37L4000, (DATE OF LAST OVERWRITING AND READING),
 DUP, =E12M5, DUP, =E18M5, (TO CARD 2 OF FILE HEAD),
 DUP, =E52M5, =E58M5, (TO CARD 6 OF FILE HEAD),
 E13M5, SHL-6, JSP4001, NOT, NEG, (UPDATE NUMBER OF OVERWRITES),
JS1P4001, SHL18, SHL-12, =E13M5, (TO CARD 2 OF FILE HEAD),
E19M5, SHL-6, JSP4001, NOT, NEG, (UPDATE NUMBER OF READS),
 JS1P4001, SHL18, SHL-12, =E19M5, (TO CARD 2 OF FILE HEAD),
E53M5, SHL-6, JSP4001, NOT, NEG, (UPDATE NUMBER OF OVERWRITES),
JS1P4001, SHL18, SHL-12, =E53M5, (TO CARD 6 OF FILE HEAD),
 E59M5, SHL-6, JSP4001, NOT, NEG, (UPDATE NUMBER OF READS),
 JS1P4001, SHL18, SHL-12, =E59M5, (TO CARD 6 OF FILE HEAD),
 ZERO, =RM7, J5, (FIND POSITION IN QUEUE),
4, SET-4, =+M5, M0M5, SHL6, (TOTAL UP NUMBER OF),
 SHL-24, JSP4001, =+C7, M+I7, (JOBS AND TIME LIMITS),
5, M0M60, V3L4000, NEV, J5=Z, (////),
 M-I6, M0M6, M+I6, V21L4000, NEV, J6NEZ, (*JOB),
 M6T005, SET-13, =+M5, M0M5, (FETCH FILE NAME),
 V28L4000, NEV, SET1, =+M5, J4NEZ, (WORD1 FILENAME NOT SAME),
 M0M5, V29L4000, NEV, J4NEZ, (WORD2 FILENAME NOT SAME),
 M7, JS2P4001, SHL36, SHL-12, (NUMBER OF JOBS),
 V63L4444, OR, =V63L4444, (TO MESSAGE),
 C7, JS2P4001, SHL30, SHL-18, (TOTAL TIME),
 V67L4444, OR, =V67L4444, (TO MESSAGE),
 JSP4003, (IMPROVE GRAMMAR),
 (XX JOBS IN QUEUE--TOTAL TIME XXX MINUTES.),
 SETAV63L4444, SET6, JSP4050, J7, (OUTPUT MESSAGE),
```

```
      SETAV63L4444,.SET6, JSP4050, J7, (OUTPUT MESSAGE),
      (YOUR JOB IS NO LONGER IN THE QUEUE.),
   6, SETAV69L4444, SET5, JSP4050, (OUTPUT MESSAGE),
   7,  V15L4000, V43L4000, JSP5035, (QUEUE FILE BACK TO DISC),
      V4L4000, =M5, V1L3612, SET3, +, =M6, (QUEUE CONTROL WORD),
      M5M6, =Q7, DC8, C8T007, (NUMBER OF JOBS IN QUEUE),
      I0T007, Q7, =M5M6, (CLEAR BUSY MARKER),
      J2P4444, (TERMINATION ROUTINE),
 END,
*USERCODE
 ROUTINE
 P4001,V2,R1,R2*
      V0=B 12 12 12 12 12 12 12 12,
      V1=B 17 17 17 17 17 17 17 17,
      V2=B 20 20 20 20 20 20 20 20,
      (ALPHA TO BINARY),
      V0, REV, V1, AND, TOB, EXIT 1,
      (BINARY TO ALPHA,REPLACE LEADING ZEROS BY SPACES),
   1, V0, REV, FRB, DUP, SET8, =C1,
 *,10,ZERO, SHLD6, DC1, *, J11NEZ, J10C1NZS,
  11, ERASE, C1, DUP, SHL1, +, SHL1, SET42, -, =C1,
      V2, SHLC1, OR, EXIT 1,
      (BINARY TO ALPHA,REPLACE LEADING ZEROS BY DUMMYS),
   2, V0, REV, FRB, DUP, SET8, =C1,
 *,20,ZERO, SHLD6, DC1, *, J21NEZ, J20C1NZS,
  21, ERASE, C1, NOT, NEG, DUP, SHL1, +, SHL1, =C1,
      ZERO, NOT, SHLC1, OR, V2, OR, EXIT 1,
 END,
*USERCODE
 ROUTINE
 P4002*
      (FIND SECTOR ADDRESS OF FILE),
      (ENTRY,N1-L/S OF USERS DIRECTORY FILE),
      (      N2-FIRST WORD OF FILE NAME),
      (      N3-SECOND WORD OF FILE NAME),
      (EXIT,N1-SECTOR ADDRESS OF FILE BODY),
      DUP, =Q8, (SAVE L/S OF USERS DIRECTORY FILE),
      V0L4888, JSP5035, (READ FIRST SECTOR),
```

```
      V11L4888, JSP4001, (NUMBER OF ENTRIES IN DIRECTORY),
      SHL-3, NOT, NEG, =C7, (NUMBER OF SECTORS CONTAINING ENTRIES),
    1, J50C7Z, (NO MORE SECTORS , FILENAME NOT IN DIRECTORY),
      Q8, DUP, NOT, NEG, =Q8, (INCREMENT SECTOR ADDRESS),
      V0L4888, JSP5035, (READ CURRENT SECTOR),
      DC7, (DECREASE SECTOR COUNT),
      V43L4888, =05, (CONTROL TO FETCH FIRST WORD OF FILE NAMES),
  *,2, M0M50, J3= , *, J2C5NZS, (COMPARE FIRST WORDS),
      J1, (NO MATCH FOUND ON 1ST WORDS,TRY NEXT SECTOR),
    3, REV, DUP, SHL24, SHC24, (VERSION NUMBER),
      M5, SET4, -, =RM6, M0M60, (2ND WORD OF FILE NAME),
      REV, J4NEZ, (VERSION NUMBER NOT BLANK),
      SHL-24, SHL24, (DISREGARD VERSION NUMBER),
    4, J5=, (2ND WORDS OF FILE NAME MATCH),
      REV, J2C5NZ, (MORE ENTRIES IN CURRENT SECTOR),
      J1, (NO MATCH FOUND ON 2ND WORDS , TRY NEXT SECTOR),
    5, ERASE, ERASE, M0M60, (WORD 2 OF ENTRY),
      DUP, J51=Z, (FILE DOES NOT EXIST),
      DUP, SHL12, SHC6, SHL12, SHC6,
      SHL12, V16L4000, NEV, J53=Z, (FILE ARCHIVED),
      DUP, SHL-42, SETB46, J6NE, (NO FAULT IN FILE),
      SETAV94L4444, SET4, JSP5050, ERASE, J7, (FILE FAULTY),
    6, SETB44, J52=, (FILE DELETED),
      SETB41, J7NE, (SECTOR ADDRESS FOUND),
      M0M6, V19L4000, J54=, (FILE TO BE RESTORED),
      V18L4000, NEV, J55=Z, (FILE TO BE ARCHIVED),
    7, ERASE, SHL6, SHL-6, JSP4001, (SECTOR ADDRESS OF FILE HEAD),
      EXIT 1,
   50, ERASE, ERASE,
      SETAV42L4444, SET3, JP4444, (FILE NOT FOUND),
   51, ERASE,
      SETAV90L4444, SET4, JP4444, (FILE DOES NOT EXIST),
   52, ERASE, ERASE,
      SETAV45L4444, SET2, JP4444, (FILE DELETED),
   53, ERASE, ERASE,
      SETAV47L4444, SET3, JP4444, (FILE ARCHIVED),
   54, ERASE, ERASE, ERASE,
      SETAV50L4444, SET3, JP4444, (FILE TO BE RESTORED),
```

```
   55, ERASE, ERASE,
       SETAV101L4444, SET3, JP4444, (FILE TO BE ARCHIVED),
  END,
 *USERCODE
  ROUTINE
  P4003*
       (IMPROVE GRAMMAR),
       M7, J1NEZ,
       V65L4444, SETB7777, OR, =V65L4444,
       ZERO, NOT, DUP, =V66L4444, DUP, =V67L4444,
       SHL12, V68L4444, OR, =V68L4444,
       EXIT 1,
    1, M7, NEG, NOT, J2NEZ,
       SETB77, SHC-6, V64L4444, OR, =V64L4444,
       ZERO, NOT, SHL12, V66L4444, OR, =V66L4444,
       C7, NEG, NOT, J2NEZ,
       SETB77, SHL12, V68L4444, OR, =V68L4444,
    2, EXIT 1,
  END,
 *USERCODE
  ROUTINE
  P4444,R1,R2,R3*
       JSP5050, ERASE, SET1, =V0L4000, J2,
    1, SHL24, V100L4444, OR, =V100L4444, (SYSTEM FAILURE /XX/.),
       SETAV98L4444, SET3, JSP5050, (OUTPUT MESSAGE),
       SET1, =V0L4000, (INTERACTIVE FAILURE EXIT),
    2, V1L3612, SET3, +, V4L4000, +, =M5, (QUEUE CONTROL WORD),
       M0M5, =Q6, I0TOQ6, Q6, =M0M5, (CLEAR BUSY MARKER),
       V20L4000, J3NEZ, (QUIET MARKER SET),
       ZERO, NOT, =V20L4000, JS10P4050, (OUTPUT MESSAGES),
    3, J4EJ, LINK, ERASE, J3, (CLEAR SJNS),
    4, J5EN, ERASE, J4, (CLEAR NEST),
    5, J3NEJ, J4NEN, (CHECK SJNS AND NEST CLEARED),
       V1L4000, =LINK, V0L4000, EXIT,
  END,
 *USERCODE
  ROUTINE
  P4050,R10*
```

```
      (R0 - SAVE MESSAGE PARAMETERS),
      (R10 - LOAD MESSAGES INTO BUFFER AND OUTPUT),
      SET5, =C5, I5=+2, SETAV80L4444, =M5,
 *,1, M0M5Q, J2=Z, *, J1C5NZS, J50,
   2, M-I5, I5=+1, =M0M5Q, =M0M5, EXIT 1,
  10, V45L4888, =Q5,
      SET10, =RC6, SETAV80L4444, =M6,
  11, M0M6Q, =RC7, J13C7Z, (NO MESSAGES),
      M0M6Q, DUP, =M7, J51LEZ, (NUMBER OF WORDS LE ZERO),
 *,12,M0M7Q, =M0M5Q, *, J52C5Z, J12C7NZS, J11, (FILL BUFFER),
  13, V14L4000, =M0M5Q, C5, SET38, J14=, (INSERT END MESSAGE),
      SETAV1L4888, REV, SET39, REV, -, JSP5050, (OUTPUT BUFFER),
  14, ERASE, EXIT 1,
  50, SETB172520, J1P4444, (TOO MANY MESSAGES),
  51, SETB172521, J1P4444, (ZERO WORDS IN MESSAGE),
  52, SETB172522, J1P4444, (OUTPUT BUFFER FULL),
 END,
*USERCODE
 ROUTINE
 L4444,V103*
      (NO ONE IS LOGGED IN.),
      V0=B 02 07 56 57 00 57 56 45,
      V1=B 00 51 63 00 54 57 47 47,
      V2=B 45 44 00 51 56 37 02 75,
      (TYPE NOT RECOGNISED.),
      V3=B 02 07 64 71 60 45 00 56,
      V4=B 57 64 00 62 45 43 57 47,
      V5=B 56 51 63 45 44 37 02 75,
      (THIS QUEUE IS NOT BEING SERVICED AT THE MOMENT),
      V6=B 02 07 64 50 51 63 00 61,
      V7=B 65 45 65 45 00 51 63 00,
      V8=B 56 57 64 00 42 45 51 56,
```

```
V9=B 47 00 63 45 62 66 51 43,
V10=B 45 44 00 41 64 00 64 50,
V11=B 45 00 55 57 55 45 56 64,
V12=B 37 02 77 77 77 77 77 77,
(THIS QUEUE IS BEING SERVICED BUT THE),
(BACKGROUND SYSTEM IS PAUSED.),
V13=B 02 07 64 50 51 63 00 61,
V14=B 65 45 65 45 00 51 63 00,
V15=B 42 45 51 56 47 00 63 45,
V16=B 62 66 51 43 45 44 00 42,
V17=B 65 64 00 64 50 45 00 42,
V18=B 41 43 53 47 62 57 65 56,
V19=B 44 00 63 71 63 64 45 55,
V20=B 00 51 63 00 60 41 65 63,
V21=B 45 44 37 02 77 77 77 77,
(YOUR MAXIMUM TIME FOR THIS QUEUE IS XXXXXXX MINUTES),
V22=B 02 07 71 57 65 62 00 55,
V23=B 41 70 51 55 65 55 00 64,
V24=B 51 55 45 00 46 57 62 00,
V25=B 64 50 51 63 00 61 65 45,
V26=B 65 45 00 51 63 00 00 00,
V27=B 00 00 00 00 00 00 55 51,
V28=B 56 65 64 45 63 37 02 77,
(DATA INCOMPLETE.),
V29=B 02 07 44 41 64 41 00 51,
V30=B 56 43 57 55 60 54 45 64,
V31=B 45 37 02 75 77 77 77 77,
(TIME SPECIFIED UNACCEPTABLE),
V32=B 02 07 64 51 55 45 00 63,
V33=B 60 45 43 51 46 51 45 44,
V34=B 00 65 56 41 43 43 45 60,
V35=B 64 41 42 54 45 37 02 75,
(TOO MANY DATA ITEMS.),
V36=B 02 07 64 57 57 00 55 41,
V37=B 56 71 00 44 41 64 41 00,
V38=B 51 64 45 55 63 37 02 75,
(ILLEGAL JOB CODE.),
V39=B 02 07 51 54 54 45 47 41,
```

```
V40=B 54 00 52 57 42 00 43 57,
V41=B 44 45 37 02 75 77 77 77,
(FILE NOT FOUND),
V42=B 02 07 46 51 54 45 00 56,
V43=B 57 64 00 46 57 65 56 44,
V44=B 37 02 75 77 77 77 77 77,
(FILE DELETED),
V45=B 02 07 46 51 54 45 00 44,
V46=B 45 54 45 64 45 44 02 75,
(FILE ARCHIVED),
V47=B 02 07 46 51 54 45 00 41,
V48=B 62 43 50 51 66 45 44 37,
V49=B 02 75 77 77 77 77 77 77,
(FILE TO BE RESTORED),
V50=B 02 07 46 51 54 45 00 64,
V51=B 57 00 42 45 00 62 45 63,
V52=B 64 57 62 45 44 37 02 75,
(THE QUEUE IS FULL),
V53=B 02 07 64 50 45 00 61 65,
V54=B 45 65 45 00 51 63 00 46,
V55=B 65 54 54 37 02 75 77 77,
(REMAINING RUNNING TIME ON BACKGROUND JOB XXX MINUTES),
V56=B 02 07 62 45 55 41 51 56,
V57=B 51 56 47 00 62 65 56 56,
V58=B 51 56 47 00 64 51 55 45,
V59=B 00 57 56 00 42 41 43 53,
V60=B 47 62 57 65 56 44 00 52,
V61=B 57 42 00 77 77 20 00 55,
V62=B 51 56 65 64 45 63 37 77,
(XX JOBS IN THE QUEUE--TOTAL TIME XXX MINUTES.),
V63=B 02 07 00 00 00 52 57 42,
V64=B 63 00 51 56 00 64 50 45,
V65=B 00 61 65 45 65 45 36 36,
V66=B 64 57 64 41 54 00 64 51,
V67=B 55 45 00 00 00 00 00 55,
V68=B 51 56 65 64 45 63 37 02,
(YOUR JOB IS NO LONGER IN THE QUEUE),
V69=B 02 07 71 57 65 62 00 52,
```

```
V70=B 57 42 00 51 63 00 56 57,
V71=B 00 54 57 56 47 45 62 00,
V72=B 51 56 00 64 50 45 00 61,
V73=B 65 45 65 45 37 02 77 77,
(THIS COMMAND CAN ONLY BE CALLED FROM A CONSOLE.),
V74=B 02 64 50 51 63 00 43 57,
V75=B 55 55 41 56 44 00 43 41,
V76=B 56 00 57 56 54 71 00 42,
V77=B 45 00 43 41 54 54 45 44,
V78=B 00 46 62 57 55 00 41 00,
V79=B 43 57 56 63 57 54 45 75,
((QUEUE FOR MESSAGE PARAMETERS)),
V80=0,
V81=0,
V82=0,
V83=0,
V84=0,
V85=0,
V86=0,
V87=0,
V88=0,
V89=0,
(THIS FILE DOES NOT EXIST.),
V90=B 02 07 64 50 51 63 00 46,
V91=B 51 54 45 00 44 57 45 63,
V92=B 00 56 57 64 00 45 70 51,
V93=B 63 64 37 02 75 77 77 77,
(** YOUR FILE COULD BE FAULTY),
V94=B 02 06 36 36 07 00 71 57,
V95=B 65 62 00 46 51 54 45 00,
V96=B 43 57 65 54 44 00 42 45,
V97=B 00 46 41 65 54 64 71 37,
(SYSTEM FAILURE /XX/.),
V98=B 02 07 63 71 63 64 45 55,
V99=B 00 46 41 51 54 65 62 45,
V100=B 00 00 00 00 17 37 02 75,
(FILE TO BE ARCHIVED.),
V101=B 02 07 46 51 53 45 00 64,
```

```
        V102=B 57 00 42 45 00 41 62 43,
        V103=B 50 51 66 45 44 37 02 75,
  END,
*USERCODE
  ROUTINE
  L4888,V45*
        (BUFFER USED TO READ/WRITE SECTORS),
        V0=Q 0/AV1/AV40,
        V41=Q 1/AV1/AV40,
        V42=Q 20/2/AV1,
        V43=Q 8/5/AV1,
        V44=Q 8/1/AV31,
        V45=Q 39/1/AV1,
  END,
```

Appendix B

The Initial Suspension Routine

```
*USERCODE
  ENTRYNAMES SUSPND*
  ROUTINE P1111,V48*
      J2EN, =V1, Q1, =V34, SETAV1, =RM1, V1,
    1, =MOM10, J1NEN, J3,
    2, Q1, =V34, COTOQ1,
    3, C1, NEG, =V0, SETAV18, =RM1, J5EJ,
    4, LINK, =MOM10, J4NEJ,
    5, C1, NEG, =V17,
       Q2, =V35, Q3, =V36, Q4, =V37, Q5, =V38, Q6, =V39,
       Q7, =V40, Q8, =V41, Q9, =V42, Q10, =V43, Q11,=V44,
       Q12, =V45, Q13, =V46, Q14, =V47, Q15, =V48,
       SETAR100, =M1, MOM1, =E0,
       SETAZ170, =R11, SET2, =C1, Q1, SET101, OUT,
       SETAZ1, SETAZ160, SHL16, OR, Z169, Z168, SET130, OUT,
       ERASE, Z167, Z166, Z165, AND, TOB,
       SET1, SHL32, OR, SET106, OUT,
   50, V35, =Q2, V36, =Q3, V37, =Q4, V38, =Q5, V39, =Q6,
       V40, =Q7, V41, =Q8, V42, =Q9, V43, =Q10, V44, =Q11,
       V45, =Q12, V46, =Q13, V47, =Q14, V48, =Q15,
       V17, DUP, =C1, I1=-1, SETAV17, +, =M1, JS2C1Z,
   51, MOM10, =LINK, J51C1NZ,
   52, V0, DUP, =C1, SETAV0, +, =M1, JS4C1Z,
   53, MOM10, JS3C1NZ,
       ERASE, V34, =Q1, V1, EXIT 1,
    54, V34, =Q1, EXIT 1,
  *;100, J50,
  END,
```

```
*SUBSTITUTE COTMIN
*CHAIN3
*USERCODE
 ROUTINE
 P1,V3690*
 END,
*USERCODE
 ROUTINE
 P0,V2000*
       V0=0 0/AP1021/AV0P0,
       V1=0, (RELATIVE COMMAND NUMBER),
       V2=0 0/8800/3,
       V3=B 72 45 70 45 43 65 64 45, (EXECUTE),
       V4=0 2/0/0,
       V5=0 0/AV11/AV13,
       V6=0 0/8800/4,
       V7=B 72 62 45 63 65 55 45, (RESUME),
       V8=0 2/0/1,
       V9=0 0/AV11/AV13,
       V10=0,
       V11=1,
       V12=1,
       V13=B 02 06 21 61 22 07 34,
       SETAV0P0, JP2,
 END,
*SUBSTITUTE FILL00
*SUBSTITUTE FILL01
*USERCODE
 ROUTINE
 P1021*
       (MAIN ROUTINE),
       JSP3611, (INITIALISE COMMUNICATOR AREA),
       V1L3612, =M1, E7M1, =M1, E1M1, (0/AP4987/AV0L133),
       SHL32, SHC16, =V1L1024, (AV0L133 DISPLAY QUEUE PARAMETERS),
       =V3L1021, (ENTRY NUMBER,0-EXECUTE/1-RESUME),
       =V0L1029, (RETURN ADDRESS),
       V4L3611, =M1, E10M1, =V4L1021, (INTERACTIVE MARKER),
       V3L3611, =RM1, SETAV0L1026, =M0M1, (DUMP AREA FOR SUP),
```

```
        M-I1, M0M1, =V11L1020, (DUMP AREA FOR BACKGROUND),
        V2L3611, =M1, M0M1, J50=Z, (NO FILE OPEN),
        E1M1, DUP, SET1130, -, J51GTZ, (FILE TOO BIG),
        SHL1, DUP, SHL2, +, (SIZE OF FILE IN WORDS),
        DUP, SET104, -, =V16L1020, (SIZE OF PROGRAM IF NOT EXECUTE),
        SET112, +, (SPACE FOR OUTPUT BUFFERS AND ROUND UP),
        V3L1021, V4L1021, OR, J1NEZ, (RESUME OR INTERACT),
        DUP, SET112, -, =V16L1020, (SIZE OF PROGRAM IF EXECUTE),
        SET104, +, (SPACE FOR PROMPT BUFFER AND DUMP AREA),
     1, JSP5029, (OBTAIN EXTERNAL WORK SPACE),
        DUP, J52LTZ, (NOT ENOUGH EXTERNAL WORK SPACE),
        V4L1021, J2NEZ, (INTERACTIVE RETURN),
        SETAV0L1021, SET3, JSP5050, ERASE, (OUTPUT MESSAGE),
     2, DUP, =V0L1025, (OUTPUT BUFFER 1),
        SET40, +, DUP, =V1L1025, (OUTPUT BUFFER 2),
        SET144, +, SHL-5, NOT, NEG, (BA-BASE ADDRESS),
        DUP, SHL5, SET64, -, DUP, =V12L1020, (BA OF DUMP AREA),
        SET40, -, =V3L1025, (BA OF PROMPT BUFFER),
        V16L1020, SHL-5, NEG, NOT, (NOL-NUMBER OF LOCATIONS),
        SHL14, OR, =V15L1020, (SET UP BANOL),
        SET20, SHL24, =V2L1020, (SET RUN TIME LIMIT),
        ZERO, =V8L1020, (CLEAR OVERFLOW AND TEST REGISTERS),
        V3L1021, V4L1021, OR, J1P1023NEZ, (RESUME/INTERACT),
        JP1022, (EXECUTE),
    50, SET1, J100, (NO FILE OPEN),
    51, ERASE, SET2, J100, (FILE TOO BIG),
    52, ERASE, SET3, (NOT ENOUGH EXTERNAL WORK SPACE),
   100, SET1021, JP1028, (FAILURE ROUTINE),
   END,
  *USERCODE
   ROUTINE
   L1021,V4*
        (F-F-FROZEN FFORTRAN.),
        V0=B 07 02 46 36 46 36 46 62, (NCF-F-FR),
        V1=B 57 72 45 56 00 46 46 57, (OZEN FRO),
        V2=B 62 64 62 41 56 37 02 75, (RTRAN.CE),
        V3=0, (ENTRY NUMBER,0-EXECUTE/1-RESUME),
        V4=0, (INTERACTIVE MARKER),
```

```
 END,
*IDENTIFIERP3611
*USERCODE
 ROUTINE
 P3611,V0,R1*
       (LOAD COMMUNICATION AREAS IN COMMAND PROGRAM),
       (ENTERED WITH ADDRESS OF ADDRESSES OF AREAS IN SUPERVISOR),
       REV,=V0,
  1,   V0,
       =M1,M0M1,=Q1,
       I1,=RM2,
       SET10,(SIZE OF JUMP TABLE),=C2,
       SETAP3600,=RM3,
 *,3,  M0M2Q,*,=M0M3Q,J3C2NZS,
       SET10,(SIZE OF BA LIST FOR SUPERVISOR V STORES),=RC2,
       ZERO,=RM3,
 *,2,  M1M2Q,SHL+16,SHL-32,*,=V0L3611M3Q,J2C2NZS,
       (NOW LOAD BA OF GLOBAL V STORES),
       E14M1,SHL32,SHL-32,=V0L3612,(ADDRESS OF DATE),
       E12M1,DUP,SHL32,SHL-32,=V1L3612,(ADDRESS OF PERM INFO),
       (NOW LOAD SECONDARY JUMP TABLE),
       =M2,E7M2,=M2,E1M2,SHL-16,=RM2,
       SET2, =C2, (SIZE OF SECONDARY JUMP TABLE),
       SETAP5051,=RM3,
  *,4,   M0M2Q,*,=M0M3Q,J4C2NZS,
       E11M1,SHL32,SHL-32,=V2L3612,(ADDRESS OF CONSOLE STATUS),
       E7M1,SHL32,SHL-32,=V3L3612,(AV0L4991),
       E2M1,SHL32,SHL-32,=V4L3612,(AV0L3835),
       E21M1,SHL32,SHL-32,=V5L3612,(AV0L300),
       E1M1,SHL32,SHL-32,=V6L3612,(AV0L0),
       E16M1,SHL32,SHL-32,=V7L3612,(AV0L4988),
       EXIT1,
       END,
*IDENTIFIERP3600
*USERCODE
 ROUTINE
 P3600,R1,R2,R3,R4,R5,R6,R7,R8,R9
       ,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20
```

```
      *
 1,    JE0,(UPDATE CLOCK),
 2,    JE0,(JP5030),
 3,    JE0,(JP5024),
 4,    JE0,(JP5004),
 5,    JE0,(JP5007),
 6,    JE0,(JP5027),
 7,    JE0,(CALLS ROUTINE TO SET UP OUTPUT),
 8,    JE0,(CALLS P300),
 9,    JE0,(DUMP Q STORES ETC P5002),
10,    JE0,(UNDUMP  P5003),
11,JE0,(J3P8191),
12,JE0,(JP8189),
13,JE0,(J1P10),
14,JE0,(JP5015),
 15,J15,(JP5050),
 16,J16,(JP5035),
 17,J17,(JP5036),
 18,J18,(JP5037),
 19,J19,(JP5031),
 20,J20,(SPARE),
     END,
*IDENTIFIER
*IDENTIFIERL3611
*USERCODE
 ROUTINE
 L3611,V9*
      (ADDRESSES OF V STORES IN SUPERVISOR),
      V0=0,(ADDRESS OF USER PARAMS),
      V1=0,(ADDRESS OF CURRENT DATA),
      V2=0,(ADDRESS OF PARAMS DEFINING FILE IN TOP CELL OF FNS),
     V3=0,(AV1L4996  ADDRESS OF LOCATION FOR ADDRESS OF DUMP AREA),
      V4=0,(AV0L5001  ADDRESS OF TEMP VARIABLES IN SUPERVISOR),
      V5=0,(AV0L3984  ADDRESS OF COMMAND INDEX),
      V6=0,(AV0L5029),
      V7=0,(AV0L3611),
      V8=0,(AV0L5030),
      V9=0,(AV0L5002),
```

```
        END,
*IDENTIFIER P5051
*USERCODE
  ROUTINE
 P5051,R2,R3,R4,R5*
        (SECONDARY JUMP TABLE BYPASSING ONLINE SUPERVISOR),
        2,  JE0,(SEARCH LIST OF ATTACHED FILES),
        3,  JE0,(SEARCH INCORE SYSTEM DIRECTORY),
    4, JE0, (JP3201 PLACE ENTRY IN PDP8 QUEUE),
    5, JE0, (SPARE),
        END,
*IDENTIFIERL3612
*USERCODE
 ROUTINE
 L3612,V7*
        (ADDRESSES OF V STORES IN GLOBAL),
        V0=0,(AV0L320  ADDRESS OF DATE),
        V1=0,(AV0L4989  BA OF PERMANENT GLOBAL INFO),
        V2=0,(AV0L5000  BA OF CONSOLE STATUS),
        V3=0,(AV0L4991),
        V4=0,(AV0L3835),
        V5=0,(AV0L300),
        V6=0,(AV0L0),
        V7=0,(AV0L4988),
        END,
*IDENTIFIER L5050
*USERCODE
 ROUTINE
 L5050,V6*
        V0=0, (LINK),
        V1=2, (PRIORITY),
        V2=0, (NUMBER OF WORDS IN BUFFER),
        V3=0, (BA OF BUFFER),
        V4=B 02 57 65 64 60 65 64 00, (OUTPUT),
        V5=B 42 65 46 46 45 62 00 46, (BUFFER),
        V6=B 65 54 54 02 37 75 77 77, (FULL.),
 END,
*IDENTIFIERD5050
```

```
*USERCODE
 ROUTINE
 P5050*
      (WRITE OUTPUT BUFFER FOR COMMAND PROGRAM OR INTERNAL SUBROUTINE),
     (N1=NO OF WORDS IN BUFFER),
      (N2=BA OF BUFFER),
      =V2L5050, =V3L5050,
      V0L3611, =M1, E1M1, SHC-1, J2GEZ,
      (OUTPUT IS SUPPRESSED), ZERO, NOT, EXIT 1,
   2, V2L5050, J1NEZ,
      ZERO, EXIT 1,
   1, LINK, =V0L5050,
   4, V1L5050, JS4P3600, (WAIT FOR DISC),
      V3L5050, V2L5050, V4L3611, =M1, M0M1, JS7P3600, (JSP5005),
     =V2L5050, (OUTPUT EXCESS MARKER),
      ZERO, (MARKER TO CAUSE OVERLAPPED TRANSFER),
      JS5P3600, (DISC ROUTINE),
      V2L5050, J3=Z, (MORE OUTPUT SPACE AVAILABLE),
      V0L3611, =M1, E1M1, SHC-1, J3LTZ, (OUTPUT SUPPRESSED),
      E1M1, NOT, NEG, =E1M1,
      SET3, =V2L5050, SETAV4L5050, =V3L5050,
      J4, (OUTPUT FINAL MESSAGE),
   3, V2L5050, V0L5050, =LINK, EXIT 1,
      END,
*SUBSTITUTE DISCTR
*SUBSTITUTE FNSCON
*SUBSTITUTE SAVCOR
*USERCODE
 ROUTINE
 P2*
      Q15, SETAR1, =M15,
      M0M15, =V10L1020, (TRAP JUMP),
      =015, JP8102,
 *,1, JP1020,
 END,
*USERCODE
 ROUTINE
 P1020,R5,R6*
```

```
     (PSEUDO DIRECTOR),
     (BANOL-BASE ADDRESS,NUMBER OF LOCATIONS),
     (RFI-REASON FOR INTERRUPT),
     (CPDAR-CURRENT PERIPHERAL DEVICE ALLOCATION REGISTER),
     (EDT-END OF DATA TRANSFER),
     (CX-THE CONTROL AND X KEYS ON THE CONSOLE),
     LINK, =V0L1020,
     JS8P3600, (CALL P300 CLOCK ROUTINE),
     J1, (LONG PATH),
     (SHORT PATH),
     Q1, V5L3612, =M1, (BA OF L300),
     M0M1, (CLOCK OVERFLOW),
     V1L1020, (PROGRAM TIME),
     +, V13L1020, (SHORT PATH TIME),
  10, -, DUP, =V1L1020, (UPDATE RUN TIME),
     V2L1020, (TIME LIMIT),
     -, J7GEZ, (TIME LIMIT FAIL),
     =Q1, V0L1020, =LINK, (RESTORE),
     V15L1020, *, =K1, ERASE, (RESET BANOL),
     EXIT D, (RETURN TO MAIN PROGRAM),
   1, (LONG PATH N1=RFI),
     Q1, =V3L1020, (SAVE),
     V9L1020, =E2, (REPLACE ORIGINAL JUMP),
     STR, =V7L1020, (OUT MARKER),
     V5L3612, =M1, (AV0L300),
     M0M1, V1L1020, +, E1M1, +, (ADD REST OF CLOCK),
     DUP, =V1L1020, (UPDATE RUN TIME),
     V2L1020, (RUN TIME LIMIT),
     SIGN, SHA-1, OR, (D47=1 IF TIME GT LIMIT , ELSE 0),
   2, ZERO, NOT, =K2, (ALL ONES TO CPDAR),
     SHL-46, J3NTR, (TEST REGISTER NOT SET),
     SHC-2, (SAVE STATE OF TEST REGISTER),
   3, J4NV, (OVERFLOW REGISTER NOT SET),
     SHC-1, (SAVE STATE OF OVERFLOW REGISTER),
   4, =V8L1020, (OVERFLOW AND TEST REGISTERS),
     Q2, =V4L1020, Q3, =V5L1020, Q4, =V6L1020, (SAVE),
     V6L3612, =M1, (AV0L0),
     E11M1, J50NEZ, (RESET FAILURE),
```

```
      DUP, J100NEZ, (PROGRAM FAILURE),
      ERASE,
      V7L1020, JP1024NEZ, (DECODE OUT),
      V6L3612, =M1, E2M1, J6=Z, (NO EDT),
    5, V12L1020, JS9P3600, (DUMP),
      V11L1020, JS10P3600, (UNDUMP BACKGROUND),
      SETAR8, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, (P8189 SCAN ONLINE-SYSTEM RETURN ADDRESSES),
     J13P3600, (J1P10 SCAN OTHER RETURN ADDRESSES),
    6, JS8P3600, (UPDATE THE CLOCK),
      ERASE, DUMMY, DUMMY, V6L3612, =M1, (AV0L0),
      E2M1, ZERO, =E2M1, J5NEZ, (EDT),
      V4L1020, =Q2, V5L1020, =Q3, V6L1020, =Q4, (RESTORE),
      VR, V8L1020, SHA1, =TR, (RESET OVERFLOW AND TEST REGISTERS),
     E2, =V9L1020, (PRESERVE ORIGINAL JUMP),
     V10L1020, =E2, (SET NEW JUMP),
      SET2, =K2, ERASE, (ALLOCATE PAPER-TAPE PUNCH),
      V5L3612, =M1, (AV0L300),
      V1L1020, E1M1, -, (SUBTRACT REST OF CLOCK),
      V3L1020, REV, V14L1020, J10, (LONG PATH CORRECTION),
    7, (TIME LIMIT EXCEEDED),
      =V3L1020, ZERO, NOT, NEG, (N1=1 FOR TIME LIMIT),
      V9L1020, =E2, J2, (REPLACE ORIGINAL JUMP),
    8, (RETURN FROM RENEWING EDT,LOOK FOR TERMINATION MARK),
      SETAR9, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, J13P3600, (AS BEFORE),
    9, V11L1020, JS9P3600, (DUMP BACKGROUND),
      V12L1020, JS10P3600, (UNDUMP),
      V2L3612, =M1, (BA OF CONSOLE STATUS WORDS),
      V4L3611, =M2, (TEMPORARY VARIABLES),
      M0M2, =M2, (CONSOLE NUMBER),
      M1M2, SHL31, J6GEZ, (NO INTERRUPT),
      SET8, J100, (TERMINATED BY USER DEPRESSING CX),
   50, ERASE, ZERO, =E11M1, (CLEAR RESET),
      ZERO, (RESET FAILURE),
  100, SET1020, JP1028, (FAILURE ROUTINE),
  END,
*USERCODE
```

```
 ROUTINE
 L1020,V16*
       V0=0, (LINK),
       V1=0, (PROGRAM RUN TIME),
       V2=0, (RUN TIME LIMIT),
       V3=0, (Q1 DUMP),
       V4=0, (Q2 DUMP),
       V5=0, (Q3 DUMP),
       V6=0, (Q4 DUMP),
       V7=0, (OUT),
       V8=0, (OVERFLOW AND TEST REGISTERS),
       V9=0, (JUMP FROM E2 OF EGDON DIRECTOR),
       V10=0, (JUMP TO P1020 FROM E2),
       V11=0, (ADDRESS OF DUMP AREA FOR BACKGROUND),
       V12=0, (ADDRESS OF DUMP AREA FOR PSEUDO DIR.),
       V13=5151, (SHORT PATH CORRECTION),
       V14=6812, (LONG PATH CORRECTION),
       V15=0, (BANOL),
       V16=0, (MAXIMUM ALLOWED ADDRESS+1),
 END,
*USERCODE
 ROUTINE
 P1022*
       (UNSUSPEND-EXECUTE),
       V2L3611, =M1, E5M1, ABS, (FILE TYPE AND CLASS),
       SHL42, SHC6, V0L1022, NEV, J50NEZ, (NOT A D FILE),
       SET115, SHL24, (L/S),
       V15L1020, SHL38, SHL-33, DUP,
       SHL16, OR, V16L1020, +, (0/LA/UA),
       JSP5035, (READ DOWN FILE),
       ZERO, =V0L1020, (SET INITIAL ENTRY),
       ZERO, =V4L1025, (SET NO OF WORDS IN PROMPT BUFFER TO ZERO),
      J5P1020, (ENTER PSEUDO-DIRECTOR),
  50, SET1, (NOT A D FILE),
 100, SET1022, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
```

```
 L1022,V2*
       V0=B 00 00 00 00 00 00 00 44, (         D),
       V1=B 00 00 00 63 65 63 36 44, (    SUS-D),
       V2=B 00 00 63 65 63 42 51 56, (   SUSBIN),
 END,
*USERCODE
 ROUTINE
 P1023,R1*
       (UNSUSPEND-RESUME RESTART),
       SET1, =V3L1021, (SET RESUME MARKER),
       (UNSUSPEND-RESUME/INTERACT),
   1,  V2L3611, =M1, E5M1, ABS, (FILE TYPE AND CLASS),
       V1L1022, NEV, J50NEZ, (NOT A SUS-D FILE),
       SET115, SHL24, (L/S),
       V3L1025, DUP, SHL16, OR, SET103, +, (FORM 0/LA/UA),
       V16L1020, +, JSP5035, (READ DOWN FILE),
       JS1P1054, (UNDUMP VARIABLES),
       V3L1021, NEG, NOT, JP1027=Z, (RESUME , PROMPT AND SUSPEND),
       V32L1032, J1P1026=Z, (RESUME READ),
       J1P1032, (RESUME CARD READ),
  50,  SET1, (NOT A SUS-D FILE),
 100,  SET1023, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 P1024*
       (OUT ROUTINE),
       ZERO, =V7L1020, (CLEAR OUT MARKER),
       V0L1020, SETB57777, -, DUP, J1GEZ,
       SETB120000, SHC1, -,
   1,  =V0L1020, (ADD 3 SYLLABLES TO OUT SET ADDRESS),
       J50EN,
       SET100, -, DUP, J51LTZ,
       =C1, V0L1024, SHLC1, DUP, J52GEZ,
       BITS, NEG, NOT, =LINK, EXITAR2,
 *,2,  (OUT133), J53EN, JP1072, (DISPLAY),
 *,    (OUT131), J10P1030, (JOB CODE),
 *,    (OUT123), J54EN, JP1025, (WRITE),
```

```
 *,    (OUT114), JP1030, (DATE AND TIME),
 *,    (OUT112), JP1023, (RESTART AFTER FORTRAN FAILURE),
 *,    (OUT104), J55EN, JP1031, (PRINTER OUTPUT),
 *,    (OUT102), J56EN, JP1026, (READ),
 *,    (OUT101), J57EN, JP1032, (CARD INPUT),
 *,    (OUT100), JP1029, (TERMINATE),
  50, SET100, J100, (NO OUT NUMBER),
  51, SET100, +, J100, (OUT NUMBER LT 100),
  52, ERASE, SET100, C1, +, J100, (ILLEGAL OUT NUMBER),
  53, SET133, J100, (INSUFFICIENT OUT DETAILS FOR DISPLAY),
 54, SET123, J100, (INSUFFICIENT OUT DETAILS FOR WRITE),
 55, SET104, J100, (INSUFFICIENT OUT DETAILS FOR CARD/PRINTER OUTPUT),
  56, SET102, J100, (INSUFFICIENT OUT DETAILS FOR READ),
  57, SET101, (INSUFFICIENT OUT DETAILS FOR CARD INPUT),
 100, SET1024, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 L1024,V1*
      V0=B 7 2 0 0 5 0 0 1 0 0 2 4 0 0 0 0,
      V1=0, (AV0L133),
 END,
*USERCODE
 ROUTINE
 P1025,R1,R2,R4*
      (WRITE,ENTER WITH N1=1/LA/UA OF MESSAGE),
      =01, C1, JSP1052, (CHECK AND MODIFY BUFFER),
      NEG, NOT, J50NEZ, (ILLEGAL WRITE CONTROL),
      V3L1020, SHL-32, J3NEZ, (PROMPT),
      V0L1025, =RM2, (0/1/BA OF OUTPUT BUFFER),
      JSP1053, (TRANSFER MESSAGE TO OUTPUT BUFFER),
   1, C2, NEG, =V2L1025, (NUMBER OF WORDS IN OUTPUT BUFFER),
     V12L1020, JS9P3600, (DUMP),
      V0L1025, V2L1025, JSP5050, ERASE, (OUTPUT MESSAGE),
     V12L1020, JS10P3600, (RESTORE),
      V0L1025, V1L1025, =V0L1025, =V1L1025, (REV BUF),
   2, J6P1020, (RETURN TO PSEUDO-DIRECTOR),
   3, V3L1025, =RM2, (0/1/BA OF PROMPT BUFFER),
```

```
      JSP1053, (TRANSFER PROMPT TO PROMPT BUFFER),
    4, C2, NEG, =V4L1025, (NUMBER OF WORDS IN PROMPT BUFFER),
      J6P1020, (RETURN TO PSEUDO-DIRECTOR),
   50, SET1, (ILLEGAL WRITE CONTROL),
  100, SET1025, JP1028, (FAILURE ROUTINE),
  END,
*USERCODE
 ROUTINE
 L1025,V5*
      V0=0, (BA OF OUTPUT BUFFER 1),
      V1=0, (BA OF OUTPUT BUFFER 2),
      V2=0, (NUMBER OF WORDS IN OUTPUT BUFFER),
      V3=0, (BA OF PROMPT BUFFER),
      V4=0, (NUMBER OF WORDS IN PROMPT BUFFER),
      V5=0, (Q1 DUMP.N/LA/UA-READ),
 END,
*USERCODE
 ROUTINE
 P1026,R1*
      (READ,ENTER WITH N1=N/LA/UA OF BUFFER),
      DUP, =V5L1025, (SAVE),
      SHL-32, SET7, NEV, J50NEZ, (NOT TYPE 7-DIRECT),
      V12L1020, JS9P3600, (DUMP),
      JP1027, (INTERACT),
    1, V12L1020, JS10P3600, (RESTORE),
      V5L1025, =Q1, JSP1052, (CHECK AND MODIFY BUFFER),
     M1, =RM2, V1L3611, =M1, (BA OF INPUT BUFFER),
      JSP1053, (TRANSFER MESSAGES FROM INPUT BUFFER),
     ZERO, =V4L1025, (SET NO OF WORDS IN PROMPT BUFFER TO ZERO),
     ZERO, (READ O.K.),
      J6P1020, (RETURN TO PSEUDO-DIRECTOR),
   50, SET1, (NOT TYPE 7-DIRECT),
  100, SET1026, JP1028, (FAILURE ROUTINE),
  END,
*USERCODE
 ROUTINE
 L1026,V45*
 END,
```

```
*USERCODE
 ROUTINE
 P1027*
       (SUSPEND PROGRAM AND WRITE TO TOP CELL OF FNS),
       JSP1055, (OUTPUT PROMPT),
       JSP1054, (DUMP VARIABLES),
       V3L1021, V4L1021, OR, J1=Z, (ENTRY VIA EXECUTE),
      ZERO, NOT, JSP5031, ERASE, (NEST DOWN),
    1, ZERO, JSP5031, (NEST UP),
       DUP, J50LTZ, (BA OF PARAMETERS DESCRIBING FILE),
      SET115, SHL24, (L/S),
       V3L1025, DUP, SHL16, OR, SET103, +,
       V16L1020, +, SET1, SHC-16, OR, (1/LA/UA),
       JSP5035, (WRITE FILE IN FNS),
       C1, J51GTZ,
       =M1, (BA OF PARAMETERS),
       SET104, V16L1020, +, SET10, (CONVERT FROM WORDS TO CARDS),
      /I, ERASE, =E1M1, (SIZE OF FILE),
       SET1, =M0M1, (INTERNAL FILE),
       V1L1022, =E5M1, (TYPE AND CLASS),
       V2L1022, =E9M1, (NAME),
       SET1, =V1L1029, JP1029, (INTERACTIVE EXIT),
   50, DUP, NEG, REV, NOT, J100=Z, (TOO MANY FNS LEVELS),
      SET2, J100, (FNS AREA COULD BE EXCEEDED),
   51, ERASE, ZERO, NOT, JSP5031, (NO TRANSFER NEST DOWN),
      ERASE, SET3, (FNS AREA COULD BE EXCEEDED),
  100, SET1027, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 P1028*
       (FAILURE ROUTINE),
       V6L1028, REV, FRB, V7L1028, OR,
       SHL24, SETB17, SHL18, OR,
       V6L1028, CAB, FRB, V7L1028, OR,
       SHL30, SHC18, OR, =V2L1028, (ROUTINE/NUMBER),
```

```
      V8L1028, V0L1020, FRB, V7L1028, OR,
      SHL6, SETB17, OR, SHC6, SHL6, =V4L1028, (LINK),
      SETAV0L1028, SET6, JSP5050, (OUTPUT FAILURE MESSAGE),
     ERASE, JP1029, (RETURN TO ON LINE SUPERVISOR),
 END,
*USERCODE
 ROUTINE
 L1028,V8*
      (PROG. FAILURE XXXX/YYY),
      V0=B 07 02 60 62 57 47 37 00, (NCPROG. ),
      V1=B 46 41 51 54 65 62 45 00, (FAILURE ),
      (LINK - WWWWW/S),
      V3=B 02 54 51 56 53 00 36 00, (CLINK - ),
      V5=B 02 75 00 00 00 00 00 00,
      V6=B 12 12 12 12 12 12 12 12, (D-B RADIX),
      V7=B 20 20 20 20 20 20 20 20, (EXCESS 16),
      V8=B 10 10 01 02 10 10 10 10 ,
 END,
*USERCODE
 ROUTINE
 P1029*
      (RETURN TO ON-LINE SUPERVISOR),
   1, LINK, ERASE, J1NEJ,
   2, ERASE, J2NEN,
   3, ERASE, J3NEN,
      V0L1029, =LINK, V1L1029, ZERO, =V1L1029, EXIT,
 END,
*USERCODE
 ROUTINE
 L1029,V1*
      V0=0, (RETURN ADDRESS TO ON-LINE SUPERVISOR),
      V1=0, (EXIT NUMBER,0-NORMAL/1-INTERACTION REQUIRED),
 END,
*USERCODE
 ROUTINE
 P1030,R10*
      (FETCH DATE AND TIME),
      J3EN,
```

```
      SETAV1L1030, =RM1,
   1, =M0M10, J1NEN, (EMPTY NEST),
      I1=-1, C1, SET15, +, J50=Z, (NEST WOULD OVERFLOW),
 *,2, M0M10, *, J2C1NZS, (REFILL NEST),
   3, JS1P3600, (TIME IN SECONDS TO 23 INTEGRAL PLACES),
      SHL-24, SET60, /I, V6L1028, REV, FRB, (SECONDS),
      REV, SET60, /I, V6L1028, REV, FRB, SHL18, (MINUTES)
     REV, V6L1028, REV, FRB, SHL36, (HOURS),
      OR, OR, V0L1030, OR, (TIME - HH/MM/SS),
      V0L3612, =M1, M0M1, (DATE - DD/MM/YY),
      J6P1020, (RETURN TO PSEUDO-DIRECTOR),
      (FETCH JOB CODE),
  10, ZERO, V0L3611, =M1, (BA OF USER PARAMETERS),
      E1M1, SHL-1, SHL1, (JOB CODE LEFT JUSTIFIED),
  11, REV, SHLD6, REV, DUP, J11NEZ, (RIGHT JUSTIFY),
      ERASE, SHL12, SETB207, SHLD-12, (CONTROL CHARACTERS),
     ERASE, J6P1020, (RETURN TO PSEUDO-DIRECTOR),
  50, SET1, (NEST WOULD OVERFLOW),
 100, SET1030, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 L1030,V15*
      V0=B 20 20 37 20 20 37 20 20, (00.00.00),
 END,
*USERCODE
 ROUTINE
 P1031*
      (PRINTER OUTPUT),
      =01, =02, I2, J6P1020NEZ, (IGNORE CARD OUTPUT),
      M1, I1, -, NOT, NEG, I1, =RM1, =C1, (SIZE/1/LA OF BUFFER),
      V15L1020, SHL38, SHL-33, =+M1, (+BA OF USER AREA),
      M0M1, SHL-42, SETB11, NEV, J1=Z, (PROMPT),
      V0L1025, =RM2, J2, (0/1/BA OF OUTPUT BUFFER),
   1, ZERO, NOT, =V12L1031, (SET THE PROMPT MARKER),
      V3L1025, =RM2, (0/1/BA OF PROMPT BUFFER),
   2, SET320, =RC3, SET8, =M3,
      Q4, =V33L1032, (SAVE),
```

```
      ZERO, ZERO, ZERO,
    3, J6C1Z,
      ERASE, ERASE, MØM1Q, (FETCH NEXT CHARACTER WORD),
      SET8, =I4, ZERO,
    4, SHL-6, SHLD6, I4, DUP, NEG, NOT, =I4, J3=Z, (FETCH WORD EMPTY),
      SET6, /I, SHL3, =C4, (AMOUNT TO SHIFT WORD),
      =M4, SETAVØL1031, =+M4, (ADDRESS OF CHARACTER WORD),
      MØM4, SHLC4, SHL-4Ø, SETB77, J4=,
      ZERO, SHLD42, J5NEZ, (CASE SHIFT),
      DC3, J6C3Z, (NOT ENOUGH SPACE FOR 1 MORE CHARACTER),
      SHC6, JS7, J4, (STORE CHARACTER AND REPEAT),
    5, DC3, DC3, DC3, J6C3Z, (NOT ENOUGH SPACE FOR 3 MORE CHARACTERS),
      =V11L1031, SET6, JS7, (ADD CASE SHIFT CHARACTER TO BUFFER),
      ERASE, V11L1031, SHC6, JS7, (STORE CHARACTER),
      ERASE, SET7, JS7, J4, (ADD CASE NORMAL CHARACTER AND REPEAT),
    6, ERASE, SETB75, JS7, (TERMINATE TRANSFER),
    7, SETB75, J8NE, (LAST CHARACTER),
      LINK, ERASE, ZERO, =C3,
    8, SHC-6, CAB, SHLD6, PERM, (STORE CHARACTER),
      M-I3, J1ØC3Z, (TERMINATE),
      M3, J9NEZ, (STORE WORD NOT FULL),
      CAB, =MØM2Q, ZERO, PERM, SET8, =M3, (CHARACTER WORD TO BUFFER),
    9, EXIT 1,
   10, CØTOQ4, M3, J11=Z,
      M3, SHL1, DUP, SHL1, +, =C4, (SHIFT),
   11, ERASE, ERASE, SHLC4, =MØM2Q, (LJ AND STORE LAST WORD),
      V33L1032, =Q4, (RESTORE),
      V12L1031, NOT, J12=Z, (PROMPT),
      J1P1025, (RETURN TO PSEUDO-DIRECTOR VIA WRITE),
   12, ZERO, =V12L1031, (CLEAR PROMPT MARKER),
      J4P1025, (RETURN TO PSEUDO-DIRECTOR VIA PROMPT),
  END,
*USERCODE
  ROUTINE
  L1031,V12*
      V Ø=B 00037402 00602077,
      V 1=B 45516617 45314231,
      V 2=B 46717237 03610021,
```

```
        V 3=B 04411424 05213027,
        V 4=B 06014477 06616035,
        V 5=B 07417477 10221043,
        V 6=B 11022446 11624051,
        V 7=B 12425454 13227057,
        V 8=B 14030462 14632065,
        V 9=B 15433470 16235077,
       V10=B 17636477 17600000,
        V11=0, (TEMPORARY STORE FOR CASE SHIFT CHARACTERS),
        V12=0, (PROMPT MARKER),
 END,
*USERCODE
 ROUTINE
 P1032,R1*
       (CARD READ),
       =V32L1032, (SAVE),
       V12L1020, JS9P3600, (DUMP),
       JP1027, (INTERRACT),
   1,  V12L1020, JS10P3600, (UNDUMP),
       Q4, =V33L1032, (SAVE),
       V32L1032, =Q1, ZERO, =V32L1032, (RESTORE AND CLEAR),
       I1, =RM2, V15L1020, SHL38, SHL-33, =+M2, (+BA OF USER AREA),
       SETB100, =I1, (SET CASE MARKER TO NORMAL),
       J2C1NZ,
       SET20, =C2, SET4, =RI4, J3, (BINARY READ),
   2,  C1, NEG, NOT, NEG, NOT, J50NEZ, (ILLEGAL READ CONTROL),
       SET10, =C2, SET8, =RI4, (ALPHA READ),
   3,  SET40, =RC3, V1L3611, =M3, (SIZE/1/BA OF INPUT BUFFER),
       ZERO, ZERO, ZERO,
   4,  ERASE, ERASE, J51C3Z, (END MESSAGE NOT FOUND),
       M0M30, SET8, =M1, ZERO, (FETCH CHARACTER WORD),
   5,  SHL-6, SHLD6, M1, DUP, NEG, NOT, =M1, J4=Z, (WORD FULL),
     SET6, J6=, (CASE SHIFT),
       SET7, J7NE, (NOT CASE NORMAL),
   6,  SET6, -, SHL6, =I1, ZERO, J5, (SET CASE MARKER),
   7,  SETB75, J8NE, (END OF INPUT),
       J9C1NZ, (ALPHA),
       J11, (BINARY),
```

```
  8, SET1, J52=, (COMMAND MARKER FOUND IN DATA),
     I1, OR, SET8, /I,
     SHL1, DUP, SHL1, +, =C4, (AMOUNT TO SHIFT WORD),
     =M4, SETAV0L1032, =+M4, (ADDRESS OF CHARACTER WORD),
    M0M4, SHLC4, SHL-42, SETB77, J5=, (ILLEGAL CHARACTER),
     J10C1Z,
     SHC-6, CAB, SHLD6, PERM, (STORE CHARACTER),
     I4, NEG, NOT, DUP, =I4, J5NEZ, (STORE WORD NOT FULL),
     CAB, =M0M20, J12C2Z, (BUFFER FULL),
     ZERO, PERM, SET8, =I4, J5, (FILL NEXT WORD),
  9, I4, SHL1, DUP, SHL1, +, =C4, (LJ LAST WORD),
     CAB, SHLC4, =M0M20, J12, (STORE LAST WORD AND EXIT),
 10, SET4, /I, SHL2, DUP, SHL1, +, =C4, (AMOUNT TO SHIFT),
     =M4, SETAV16L1032, =+M4, (ADDRESS OF CHARACTER WORD),
     M0M4, SHLC4, SHL-36, SETB7777, J53=, (ILLEGAL CHARACTER),
    SHC-12, CAB, SHLD12, PERM, (STORE CHARACTER),
     I4, NEG, NOT, DUP, =I4, J5NEZ, (STORE WORD NOT FULL),
     CAB, =M0M20, J12C2Z, (BUFFER FULL),
     ZERO, PERM, SET4, =I4, J5, (FILL NEXT WORD),
 11, I4, SHL2, DUP, SHL1, +, =C4, (LJ LAST WORD),
     CAB, SHLC4, =M0M20, (STORE LAST WORD AND EXIT),
 12, ERASE, ERASE, V33L1032, =Q4, (RESTORE),
     J6P1020, (RETURN TO PSEUDO-DIRECTOR),
 50, ERASE, SET1, J100, (ILLEGAL READ CONTROL),
 51, ERASE, SET2, J100, (END MESSAGE NOT FOUND),
 52, ERASE, ERASE, ERASE, SET3, J100, (COMMAND MARKER FOUND IN DATA),
 53, ERASE, ERASE, ERASE, SET4, (ILLEGAL CHARACTER),
 100, SET1032, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 L1032,V33*
     (ALPHA,CASE SHIFT EQUIVALENTS),
     V 0=B 00 77 77 77 77 77 77 77,
     V 1=B 77 77 77 77 77 77 77 77,
     V 2=B 77 77 77 77 77 11 77 77,
     V 3=B 12 13 77 14 77 07 15 16,
     V 4=B 77 77 77 77 77 77 77 77,
```

```
        V 5=B 77 77 77 77 77 77 77 77,
        V 6=B 77 77 77 77 77 77 77 77,
        V 7=B 77 77 77 77 77 77 77 77,
        (ALPHA,CASE NORMAL EQUIVALENTS),
        V 8=B 00 77 77 77 77 77 77 77,
        V 9=B 77 77 77 77 77 77 77 17,
        V10=B 20 21 22 23 24 25 26 27,
        V11=B 30 31 77 77 77 35 36 37,
        V12=B 77 41 42 43 44 45 46 47,
        V13=B 50 51 52 53 54 55 56 57,
        V14=B 60 61 62 63 64 65 66 67,
        V15=B 70 71 72 77 77 77 77 77,
        (BINARY,ALPHA EQUIVALENTS),
        V16=B 0000 7777 7777 7777,
        V17=B 7777 7777 7777 0042,
        V18=B 7777 0102 1042 4042,
        V19=B 2102 2042 1102 1400,
        V20=B 1000 0400 0200 0100,
        V21=B 0040 0020 0010 0004,
        V22=B 0002 0001 7777 7777,
        V23=B 7777 4000 2000 4102,
        V24=B 7777 4400 4200 4100,
        V25=B 4040 4020 4010 4004,
        V26=B 4002 4001 2400 2200,
        V27=B 2100 2040 2020 2010,
        V28=B 2004 2002 2001 1200,
        V29=B 1100 1040 1020 1010,
        V30=B 1004 1002 1001 7777,
        V31=B 7777 7777 7777 7777,
        V32=0, (SAVE N/A1/A2 FOR CARD READ),
        V33=0, (SAVE Q4),
 END,
*USERCODE
 ROUTINE
 P1072,R1*
        (WRITE DISPLAY MESSAGE TO PDP8 QUEUE),
        (ENTRY N1=Q 0/LA/UA),
        (LA-UA SPECIFY MESSAGE INCLUDING A TWO WORD HEADER),
```

```
      JP1073, (CHECK THAT CONSOLE 3 IS BEING USED),
   1, JSP1074, (CHECK AND MODIFY BUFFER),
      JSP1075, (CHECK MESSAGE),
      V12L1020, JS9P3600, (DUMP),
      V11L1020, JS10P3600, (UNDUMP BACKGROUND),
   2, M1, V1L1024, =M1, E3M1, (V3L133, -1/0 PDP8 BUSY/NOT BUSY),
      REV, =M1, J3=Z, (NOT BUSY),
      SETAR2, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, (P8189),
      J13P3600, (J1P10),
   3, Q1, V1L1024, =M1, ZERO, NOT,
      =E3M1, (SET USER PDP8 BUSY-CLEARED BY PDP8 WRITE),
      M0M1, =Q1, PMHQ1, =Q1, (SET LOCKOUTS),
      V1L1024, SET8501, JS4P5051, (PLACE ENTRY IN PDP8 QUEUE),
     DUMMY, DUMMY, DUMMY,
   4, M1, V1L1024, =M1, E3M1, (V3L133, -1/0 PDP8 BUSY/NOT BUSY),
      REV, =M1, J5=Z, (NOT BUSY),
      SETAR4, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, (P8189),
      J13P3600, (J1P10),
   5, V11L1020, JS9P3600, (DUMP BACKGROUND),
      V12L1020, JS10P3600, (UNDUMP),
      J6P1020, (RETURN TO PSEUDO-DIRECTOR),
 END,
*USERCODE
 ROUTINE
 P1073*
      (CHECK THAT CONSOLE 3 IS BEING USED),
      V4L3611, =M1, M0M1, (CURRENT CONSOLE NUMBER),
      DUP, SET3, -, J1NEZ, (NOT THE PDP8 MONITOR TELETYPE),
      ERASE, J1P1072, (RETURN TO MAIN PATH),
   1, J2NEZ, (NOT THE KDF9 MONITOR FLEXOWRITER),
      V3L3612, =M1, E5M1, J1P1072NEZ, (THE PDP8 IS ON),
      SETAV14L1055, SET3, J3, (PDP8 NOT ON),
   2, SETAV5L1055, SET9, (USE PDP8 MONITOR),
   3, JSP5050, ERASE, (OUTPUT MESSAGE),
      ERASE, (IGNORE OUTPUT TO THE DISPLAY),
      J6P1020, (RETURN TO PSEUDO-DIRECTOR),
```

```
 END,
*USERCODE
 ROUTINE
 P1074*
      (CHECK AND MODIFY THE DISPLAY BUFFER),
      =Q1, J50C1NZ,
      M1, DUP, V16L1020, -, J51GEZ, (UA TOO HIGH),
      I1, DUP, J52LTZ, (LA TOO LOW),
      -, DUP, J53LEZ, (MUST BE A TWO WORD HEADER AT LEAST),
      DUP, SET506, -, J54GEZ, (BUFFER TOO LARGE),
      V15L1020, SHL38, SHL-33, (BA OF USER AREA),
      DUP, =+I1, =+M1, Q1, (0/LA+BA/UA+BA),
      V1L1024, =M1, =M0M1, =E4M1, (SET V0L133 AND V4L133),
      V12L1020, JS9P3600, (DUMP),
      V11L1020, JS10P3600, (UNDUMP BACKGROUND),
   1, Q1, V1L1024, =M1, M0M1, =Q1, (0/LA+BA/UA+BA,OF BUFFER),
      TLOQ1, =Q1, J3NTR, (NO LOCKOUTS ON BUFFER),
   2, SETAR1, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, (P8189),
      J13P3600, (J1P10),
   3, SETAR4, JS11P3600, (SET RETURN ADDRESS),
      JS12P3600, (P8189),
      J13P3600, (J1P10),
   4, Q1, V1L1024, =M1, M0M1, =Q1, (0/LA+BA/UA+BA,OF BUFFER),
      TLOQ1, =Q1, J2TR, (LOCKOUTS ON BUFFER),
      V11L1020, JS9P3600, (DUMP BACKGROUND),
      V12L1020, JS10P3600, (UNDUMP),
      EXIT 1,
  50, SET1, J100, (COUNTER NON-ZERO),
  51, ERASE, SET2, J100, (UA TOO HIGH),
  52, ERASE, ERASE, SET3, J100, (LA TOO LOW),
  53, ERASE, SET4, J100, (BUFFER TOO SMALL),
  54, ERASE, SET5, (BUFFER TOO LARGE),
 100, SET1073, JP1028, (FAILURE ROUTINE),
 END,
*USERCODE
 ROUTINE
 P1075*
```

```
        (CHECK DISPLAY MESSAGE),
        V1L1024, =M1, M0M1, (0/LA+BA/UA+BA),
        SHL-16, =M2, M0M2, SHL-36, (FIELD AND TYPE),
        SETB1005, NEV, J50NEZ, (NOT FIELD 1,TYPE 5),
        E1M2, DUP, SHL36, SHC12, (A-1,A=PDP8 START ADDRESS),
        DUP, SET31, -, J51LTZ, (ADDRESS TOO LOW),
        REV, SHL24, NOT, SHL-36, (NUMBER OF PDP8 WORDS IN MESSAGE),
        DUP, CAB, +, (UPPER ADDRESS OF MESSAGE BODY),
        SET2047, -, J52GTZ, (UPPER ADDRESS TOO HIGH),
        SET3, +, SHL-2, (NUMBER OF DISPLAY WORDS),
        E4M1, NEG, NOT, NEV, J53NEZ, (NE TO DISPLAY FILE SIZE),
       EXIT 1,
   50, SET1, J100, (FIELD AND TYPE NOT 1005),
   51, ERASE, ERASE, SET2, J100, (PDP8 START ADDRESS TOO LOW),
   52, ERASE, SET3, J100, (UPPER ADDRESS TOO HIGH),
   53, SET4, (NUMBER OF DISPLAY WORDS NE SIZE OF DISPLAY FILE),
  100, SET1075, JP1028, (FAILURE ROUTINE),
  END,
*USERCODE
 ROUTINE
 P1052*
        (CHECK AND MODIFY TELETYPE BUFFER),
        (ENTRY Q1=N/LA/UA),
        (EXIT Q1=SIZE OF BUFFER/1/LA+BA OF USER AREA),
        M1,V16L1020, -, J50GEZ, (UA TO HIGH),
        M1, I1, -, DUP, NOT, NEG, =C1, (SIZE OF BUFFER),
        DUP, J51LTZ, (LA GT UA),
        SET40, -, J52GEZ, (BUFFER TOO LARGE),
        I1, DUP, =M1, J53LTZ, (LA TO LOW),
        I1=+1, V15L1020, SHL38, SHL-33, =+M1, (LA+BA OF USER AREA),
        EXIT 1,
   50, SET1, J100, (UA TOO HIGH),
   51, ERASE, SET2, J100, (LA GT UA),
   52, SET3, J100, (UA-LA GE 40),
   53, SET4, (LA TOO LOW),
  100, SET1052, JP1028, (FAILURE ROUTINE),
  END,
*USERCODE
```

```
 ROUTINE
 P1053*
      (TRANSFER MESSAGES TO EM FROM AREA A TO AREA B),
      (Q1=SIZE OF A/1/BA OF A,Q2=0/1/BA OF B),
      (LAST CHARACTER OF B BECOMES EM IF ONE NOT FOUND),
   1, M0M10, DUP, =M0M2Q, (TRANSFER MESSAGE),
   2, ZERO, SHLD6, SETB75, NEV, J3=Z, (EM FOUND),
      DUP, J2NEZ, (CHARACTER WORD NOT EMPTY),
      ERASE, J1C1NZ, (NOT END OF MESSAGE),
      M-I2, M0M2, SHL-2, SHL2, SETB75, OR, =M0M2, ZERO,
   3, ERASE, EXIT 1,
 END,
*USERCODE
 ROUTINE
 P1054,R1*
      (DUMP-UNDUMP VARIABLES),
      V12L1020, =M1, SET46, =RM2,
      V0L1020, =M1M2Q, (PROGRAM LINK),
      V3L1020, =M1M2Q, V4L1020, =M1M2Q,
      V5L1020, =M1M2Q, V6L1020, =M1M2Q, (Q-STORES),
      V8L1020, =M1M2Q, (VR/TR),
      V4L1025, =M1M2Q, (NUMBER OF WORDS IN PROMPT BUFFER),
      V5L1025, =M1M2Q, (N/LA/UA FOR READ),
      V32L1032, =M1M2Q, (N/A1/A2 FOR CARD READ),
      EXIT 1,
   1, V12L1020, =M1, SET46, =RM2,
      M1M2Q, =V0L1020, (PROGRAM LINK),
      M1M2Q, =V3L1020, M1M2Q, =V4L1020,
      M1M2Q, =V5L1020, M1M2Q, =V6L1020, (Q-STORES),
      M1M2Q, =V8L1020, (VR/TR),
      M1M2Q, =V4L1025, (NUMBER OF WORDS IN PROMPT BUFFER),
      M1M2Q, =V5L1025, (N/LA/UA FOR READ),
      M1M2Q, =V32L1032, (N/A1/A2 FOR CARD READ),
      EXIT 1,
 END,
*USERCODE
 ROUTINE
 P1055*
```

```
        (OUTPUT PROMPT),
        V4L1025, JINEZ,
        SETAV0L1055, SETS, J2, (STANDARD PROMPT),
    1,  V3L1025, V4L1025, (USERS PROMPT),
    2,  JSP5050, ERASE, (OUTPUT),
        EXIT 1,
 END,
*USERCODE
 ROUTINE
 L1055,V16*
        (NOT ENOUGH DATA. TYPE MORE TO RESUME),
        V0=B 07 02 56 57 64 00 45 56, (NCNOT EN),
        V1=B 57 65 47 50 00 44 41 64, (OUGH DAT),
        V2=B 41 37 00 64 71 60 45 00, (A. TYPE ),
        V3=B 55 57 62 45 00 64 57 00, (MORE TO ),
        V4=B 62 45 63 65 55 45 02 75, (RESUMECE),

        (PLEASE USE THE PDP8 MONITOR TELETYPE),
        (WHEN YOU OUTPUT A DISPLAY FILE.),
        V5=B 07 02 60 54 45 41 63 45, (NCPLEASE),
        V6=B 00 65 63 45 00 64 50 45, ( USE THE),
        V7=B 00 60 44 60 30 00 55 57, ( PDP8 MO),
        V8=B 56 51 64 57 62 00 64 45, (NITOR TE),
        V9=B 54 45 64 71 60 45 00 67, (LETYPE W),
       V10=B 50 45 56 00 71 57 65 00, (HEN YOU ),
       V11=B 57 65 64 60 65 64 00 41, (OUTPUT A),
       V12=B 00 44 51 63 60 54 41 71, ( DISPLAY),
       V13=B 00 46 51 54 45 37 02 75, ( FILE.CE),
        (THE PDP8 IS NOT ON.),
        V14=B 02 07 64 50 45 00 60 44, (CNTHE PD),
        V15=B 60 30 00 51 63 00 56 57, (P8 IS NO),
        V16=B 64 00 57 56 37 02 75 77, (T ON.CE ),
 END,
```