BUSINESS COMPUTER SYSTEMS PLC.

MOLECULAR 18

SOFTWARE MANUAL

MK V

Released Nov. 1984

Re Order no. 660799

Issue 2.1 - 1:11.84

CONTENTS

SECTION 1	GENERAL
SECTION 2	MACHINE CODE PROGRAMMING
SECTION 3	GENERAL LIBRARY SUBROUTINES
SECTION 4	DATA RETRIEVAL SYSTEM
SECTION 5	SPOOLING & POSTING
SECTION 6	I/O STATION & PRINTER PROGRAMMING
SECTION 7	OPERATING SYSTEM UTILITIES
SECTION 8	OPERATING SYSTEM COMMANDS
SECTION 9	MISCELLANEOUS PROGRAMMER INFORMATION
SECTION 10	APPENDICES

SECTION 1

GENERAL

CONTENTS

	PAGE
SOFTWARE CONCEPTS	1.1
The Operating System	1.1
Program and Tasks	1.1
Spooling and Print Queues	1.2
The Configuration Table	1.3
The System Catalogue	1.4
The System Tables File	1.5
HARDWARE ORGANISATION	1.6
Overview	1.6
Primary Storage	1.6
Central Processing Unit	1.9
Peripheral interface	1.9
MEMORY ORGANISATION	1.10
Addressing Techniques	1.10
Operating System Requirements	1.12
Task Requirements	1.12
System Workspace Requirements	1.13

SOFTWARE CONCEPTS

The Operating System

All software functions are controlled by a memory-based Control Program or Operating System (OS). The term 'Memory-based' denotes that after loading (or Bootstrapping) the machine, the OS is memory resident and not utilising disk backing storage. This design gives efficiency in terms of programming simplicity and operating response times.

The basic OS occupies 12K words of memory. A 1K block of memory is 1024 words addressed 0 to 1023 and is often referred to by the term 'page'.

Some of the features available within the OS are as follows:

a) input and output procedures may be carried out independently and concurrently.
b) each conversational terminal or I/O station may act as an independant control device for output functions whilst still performing its normal task as an integral part of the system.
c) a comprehensive library of application - orientated subroutines which significantly reduce program development time.
d) system control functions.
e) a comprehensive set of utilities which allow on-line loading, debugging, amendment and listing (in annotated Assembler language) of application program as well as facilitating system initiation and enhancement.
f) easy recovery procedures from security disks when required.
g) facilities to regenerate the Operating System to meet varying configuration requirements and availabilities.
h) mains failure protection facilities.

Currently available versions of the Operating System cater for various peripherals. See COMA Utility for details.

Programs and Tasks

The series of instructions together with static data consitute what is referred to as a 'Program' sometimes called a 'Program module' or an 'overlay module'. The program is held on disk within a library of programs (referred to as the Overlay library) and is called into memory when required i.e. it overlays an area of memory. A program may be called by entering its name at an I/O Station e.g. VDU. Since there may be several I/O stations, there may be several programs running at once or the same program may be running more than once - in such circumstances a multi-programming environment exists.

Each I/O station is referred to as a 'Task' and is allocated its own task number, from one upwards. The terminology is simply derived from the fact that an I/O station running any program is performing a task. Other tasks may also be running in the background - one for each active printer or tasks not requiring any I/O functions.

Each task is allocated a memory partition for its exclusive use. This partition is fixed both in size and position; all task partitions are 2K words or 2 pages long and must begin on a page boundary. Obviously certain programs will be small so that the 2K task partition will not be fully utilised. Such 'spare memory' however is emphatically not available for use by any other task. Where the program is sufficiently large enough to require more than 2K words then the technique of overlaying must be employed.

Spooling and Print Queues

As mentioned previously, each I/O station and each printer is defined as a separate task. It follows therefore that the I/O station and printer programs are logically separate units.

If, for example, an application required that some information is input, via an I/O station, processed in some way and then output to a printer, this would require two programs to be written. The 'input' program would, for example, validate the information whilst the 'print' program would, process the information and output the results. The I/O Station task does not output directly to a printer; the information to be printed has to be transferred from the I/O station task ie. the 'input' program to the printer task ie. the 'print' program. The mechanism used to achieve this transfer is referred to as 'spooling'.

Within any task partition a particular area is reserved for holding data which is to be either transferred to or from a disk-held spool file. This area is referred to as the 'Spool Buffer'. The 'input' program will set up any data required to control printing within its spool buffer and subroutines available to the programmer will be used to transfer the contents of the spool buffer to the spool file. When data is transferred to the spool file it is said to have been 'posted'.

Once an I/O Station task has posted its printing requirements to the spool file it is then free to commence any other job without waiting for the slower print operation to complete. Any number of I/O Station tasks may be posting information to the spool file simultaneously and it is essential therefore that some machanism exists whereby:

a) Related print requirements may be linked together so that subsequent printing is not garbled.

b) Printouts required to appear on pre-printed stationery may be initiated when the relevant stationery is available and has been properly loaded onto an available printer.

c) The sequence of printing various reports may be controlled to suit individual user requirements.

The means by which this is accomplished is to link in a controlled manner all postings to the spool file. A number of 'starting points' are available within the spool file and as information is posted it may be linked to any of these starting points (controlled by the program or by the operator). Subsequent postings may be linked to previous postings already linked to one of the starting points. Within the spool file therefore will exist a number of 'chains' of print requirements which are waiting or 'queuing' to be printed. The chains of postings are therefore referred to as 'Print Queues' or 'Spool Queues'.

It is important to realise that the spool file is not partitioned; partitions would have to be of some defined size which may individually become full of postings thereby causing bottlenecks in processing. By organising the spool file into print queues the manner of its sub-division becomes totally flexible e.g. one print queue may occupy for example 95% of the spool file whilst another twenty print queues may occupy the remaining 5%.

The method of actually retrieving the postings from the print queues within the spool file is fully discussed in the section - SPOOLING AND POSTING.

The Configuration Table

The initial bootstrap procedure (described later in this section) will simply load a basic Operating System (OS) into memory. Contained within the Operating System is a routine known as the Initiator which is called as soon as the OS is loaded, and this reads a Configuration Table of 3 sectors from disk into memory at location 00/0400. The Configuration Table will contain the parameters defining the System Catalogue, System Tables, memory and peripheral requirements (including Disk backing storage) for the installation. By reference to this, the Initiator will mould the basic Operating System into one which is applicable to the requirements of the installation.

The moulding process is achieved simply by transferring information from the Configuration Table into appropriate areas of the Operating System.

At any installation it will be possible to perform a specific number of jobs or tasks simultaneously. There will be a memory requirement for any task which may run on the installation and the OS is geared to handling tasks which are held within a 2K 'partition'.

For the purpose of normal running the Operating System will require to accommodate service routines for each of the peripherals in the system, File Control Blocks containing all the parameters relating to a file, outer print buffers, disk labels, disk queue vectors for any disk accesses and various tables, lists and queues. Therefore, sufficient free memory should be available for this purpose.

Various applications may be available on any installation. The definitions of these 'Application Systems', comprising of a list of all Files available to that System, number of Print queues, system passwords, date etc., are held in the System Tables File.

To summarise, therefore, for any installation, the Operating System will require to know:

a) Number of tasks which may run simultaneously
b) Peripheral type allocated to each task.
c) Address of a 2K memory partition to be reserved for each task.
d) Disk Backing Storage information.
e) A description of the location of both the Catalogue and the System Tables.
f) Location of free memory areas available within the system.

The System Catalogue

The System Catalogue is used to map out the available backing storage, thus allowing easy maintenance of disk-based data, by means of defining the basic layout and usage of the 'Data-Sets'. These may be used to define Files (see later).
Associated with each Data Set Definition and included on the appropriate catalogue record will be a 12 character name.

The System Tables File

File accessing is always done via an octal file identifier; that identifier is then used to access a pointer to the details for that file, known as the File Control Block (FCB). All FCB pointers are in fact held as a 'File Table' within the System Table file, and each pointer is simply the Catalogue Record Number which holds the FCB details for that file identifier. The initiator will, upon boot-strapping, obtain the FCB details of all files from the Catalogue and establish them in free memory areas.

File identifiers are associated with the correct FCB details by use of a Programmer Utility known as MAPS i.e. Maintain APplication System(s). Within this utility it is possible to set up/amend the file table so that any slot reserved for a file identifier may have inserted into it the catalogue record number containing the FCB details. This is done by simply entering the file identifier and the Data-Set name with which it is to be associated. A search of the System Catalogue will then take place until a catalogue record is found containing the entered name. The number of that record within the catalogue is then inserted into the slot reserved for that file identifier within the file table.

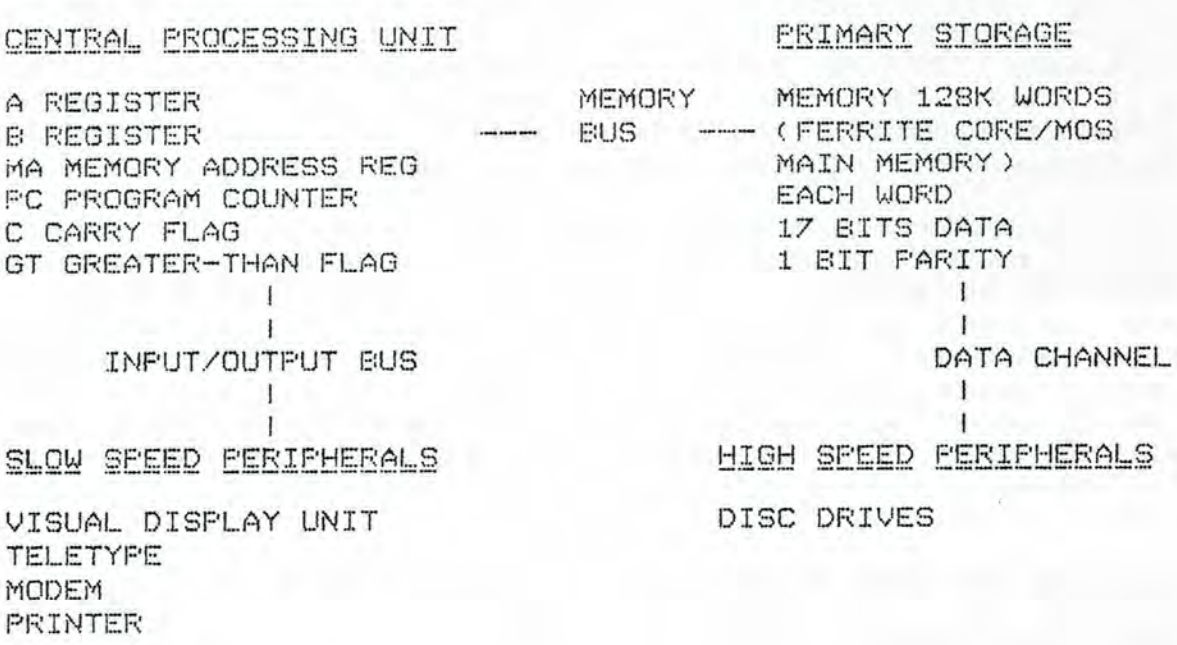
When the system is bootstrapped, one of the functions performed by the Initiator when 'moulding' the Operating System is the transfer of the file table of FCB pointers from the System table file to a location in free memory. At this point all the catalogue records specified within the file table are read into various free memory areas. The memory address of each Catalogue Record (or FCB) is then inserted into the file table replacing the initial catalogue record number. Reference to any file via a file identifier will then enable the address of the FCB details for that file to be accessed. With the FCB details located, any part of the file may be accessed.

As part of any task it will often be necessary to enable input of parameters etc. as a result of which some form of report or output document will require to be printed; this will require the use of one of the printers available to the installation and so any I/O Station must be given the ability to drive a particular printer when necessary.

Facilities are available, via Program MAPS, to associate I/O stations with certain printers and/or preclude their use by other I/O stations.

HARDWARE ORGANISATION

Overview



Primary Storage

Primary storage is a fast random access storage medium used to hold program instructions and immediate program data. It is supplied in 16K or 32K word stacks or units (1K = 1024 words). Each word consists of 18 bits where a bit is a binary digit which may have only 1 of 2 possible values ie 0 or 1. The physical medium may comprise of magnetic core or MOS chip memory, though the physical attributes of these media do not concern us here.

Only 17 of the 18 bits in each word are accessible to the program; the inaccessible 'parity' bit may be disregarded. The accessible bits are numbered 17 to 1 from left to right as follows:

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

As stated above each bit has the binary value of either zero or one according as it is 'off' or 'on' respectively. It is possible to represent the setting of each bit within a word by means of a 17 digit binary number.

eg. 00 100 010 110 100 001

indicates that bits 15, 11, 9, 8, 6 and 1 only are 'on'. However, it is more convenient to group the bits into threes as above and to represent the setting of each group by means of an Octal Number.

BINARY	OCTAL
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Thus, a word with bits 15, 11, 9, 8, 6 and 1 only 'on' is represented in octal by the six digit number 042641. All program instructions, addresses, parameters and constant factors have a binary representation but by effectively summarizing it into a 6 digit octal number the memorizing and handling of binary representations becomes much more convenient. As indicated, any word of memory may be interpreted as one of the following:

a) instruction
b) address
c) parameter
d) literal or text
e) constant factor

and depending upon the interpretation different rules will apply for determining the format of the binary representation of that word. The rules for determining the binary representation of an instruction and an address are fully discussed in Section 2 - MACHINE CODE PROGRAMMING although for convenience some discussion on addresses and addressing will take place later in this Section under MEMORY ORGANISATION. The binary representation of a parameter will be determined by either available subroutine requirements or by the particular requirements of the program, and that of a literal by the requirements of ASCII.

If the word is interpreted as a constant factor then the sum of the individual bits set 'on' will make up the value of the constant factor.

It can be seen therefore that a word with bits 16-1 inclusive set 'on' can have a maximum positive value of 65535 and a maximum negative value of -65536 if bit 17 only is set. For this reason bit 17 is described as the sign bit in words interpreted as constant factors; if it is set then the number must have a negative value, and if it is not set then the number is positive.

BIT	VALUE IF ON	CUMULATIVE VALUE
1	1	1
2	2	3
3	4	7
4	8	15
5	16	31
6	32	63
7	64	127
8	128	255
9	256	511
10	512	1023
11	1024	2047
12	2048	4095
13	4096	8191
14	8192	16383
15	16384	32767
16	32768	65535
17	-65536	-1

Again, it is more convenient to consider the binary representations of constant factors in terms of octal digits as follows.

CONSTANT	BINARY REPRESENTATION						OCTAL
8	00	000	000	000	001	000	10
10000	00	010	011	100	010	000	023420
1023	00	000	001	111	111	111	1777
-1	11	111	111	111	111	111	377777
50000	01	100	001	101	010	000	141520

It is useful at this stage to introduce the concepts of 'one's complement' and 'two's complement'; given that we have the octal represenation of a number and wish to determine the representation should the sign be reversed then application of the above concepts becomes most useful eg. to find the octal representation of -5

							OCTAL	DEC	
	00	000	000	000	000	101	000005	+5
One's complement	11	111	111	111	111	010			
Plus 1						1+			
Two's Complement	11	111	111	111	111	011	377773	-5

Central Processing Unit

This unit handles all machine input and output, arithmetic or logic operations, memory updates, program flow and execution.

The processor retrieves instructions from Primary Storage as referenced by the Program Counter; during execution of a program the Program Counter automatically points to the next instruction to be executed. Data manipulation is performed via Memory and two other registers - the A and B registers or Accumulators. Data movements to and from memory and/or peripherals are achieved by instructions which reference these accumulators. Two hardware indicators are provided for use in conjunction with the A and B registers; namely the CARRY and the GREATER THAN flags. These allow for more sophisticated arithmetic and boolean operations eg. double word arithmetic is accomplished by use of the CARRY flag.

Peripheral Interface

Most peripherals are very much slower than the CPU itself and, therefore, mechanisms must exist to allow the processor to 'start' an input/output (I/O) operation and then return to some other duty while waiting for the I/O device to complete. This gives the impression that the CPU is processing more than one task simultaneously.

The two main facilities to achieve this are Program Interrupts and Direct Memory Access (DMA) or Data Channelling. The interrupt facility allows peripheral devices to interrupt the normal flow of the CPU when a certain function is complete. High speed devices, such as disk drives, 'steal' machine cycles from the processor to directly input to or extract from memory. This action proceeds automatically, the only effect on the processor being to 'slow' its execution. These features are discussed more fully at the end of Section 2 - MACHINE CODE PROGRAMMING.

MEMORY ORGANISATION

As the Molecular 18 uses a 16 bit Memory Address register, it follows that a maximum of 64K words of memory are directly addressable by the hardware at any one time. This is ample for most situations but for some of the larger and more sophisticated applications there is a requirement for more memory than this.

To overcome this limitation a technique known as 'bank switching' has been implemented. The memory is thought of as 32K 'banks', the first bank of which contains the OS and any tables, queues, buffers etc. This is known as bank zero, addressed as pages 0 to 37 octal. Bank one is used to hold task partitions and is addressed as pages 40 to 77 octal. This constitutes the standard 64K machine. Extra 32K banks may now be added, numbered 2 to 8, each of which can contain task partitions but are also addressed as pages 40 to 77.

The OS changes or 'switches' memory bank depending on which task partition requires service next. This 'switching' is invisible to the application programs running at the time and therefore is invisible to the application programmer. The difference is apparent only when configuring the task partitions as a bank number is required as well as a partition base address.

Addressing Techniques

The term 'random access' means that any location can be accessed at any time and in any order. If follows therefore that for memory to be random access, every distinct unit or word must be uniquely addressable. Reference has already been made to Pages (or 1K blocks) of memory and within any page each word is addressed from 0 to 1023 decimal or 0 to 1777 octal.

Each page of memory if further sub-divided in what are referred to as 'columns' of 64 words each; it then follows that there are 16 columns per page of memory. For programming purposes each word within a column is referred to as a 'step'. To summarise therefore:

1 page	= 16 columns	addressed 0-17 octal
1 column	= 64 steps	addressed 0-77 octal

The column is a most important concept. It is equivalent to a complete M18 Program Coding Sheet; the contents of the coding sheet are therefore a 'window' into the memory itself.

As already mentioned, the OS supports multi-programming; it follows therefore that a program may be requested at more than one task partition and therefore must be executable in more than one physical location. Some method is then required to allow addresses within a program to be modified at execution time to refer to different areas of memory. This is accomplished by means of an OFFSET address. Any word interpreted as an address and found to have bit 17 set will be treated as an offset address; when presented to the OS the address given in bits 16-1 will be assumed to be relative, or offset, to the base of the task partition currently being processed. The notation used to indicate this is expressed as.

```
PP/CCSS-          Where PP  = Page number
                        CC  = Column Number
                        SS  = Step number
                  and   "-" = bit 17 set, or offset
                            address
```

Thus, if task 1 is allocated a partition at base 05/0000 and task 2 is allocated a partition at base 14/0000 and the same program is requested at both partitions, and within that program an address of 01/1500- occurs, then when presented to the OS it would be resolved to

```
06/1500 absolute by task 1
15/1500 absolute by task 2
```

It is often convenient to sub-divide a word into what are referred to as 'Bytes'. A byte comprises 8 bits and this is sufficient to hold the binary representation of any character in ASCII format i.e. it requires no further decoding in order that a printer might print it out. Programs are usually liberally sprinkled with texts (or literals) for displays and report headings etc. and these are packed 2 characters per word ie. 1 character per byte with bit 17 being irrelevant. When accessing literals it is often convenient to be able to refer to either character of byte of the word i.e. that in bits 16-9 (top byte), or that in bits 8-1 (bottom byte)

In order to access from the top byte the conventions detailed above are adequate, but in order to access from the bottom byte the convention used is to set bit 16 of the address word. The addressing notation to express this structure is as follows:

```
PP/CCSS.-  Where PP = page Number
                 CC = Column Number
                 SS = Step number
                "." = Bit 16 set, denotes bottom byte
                "-" = Bit 17 set, denotes offset address
```

Operating System Requirements

Assuming that on any configuration at least one I/O Station and one printer would be required and each would require a task partition of 2K words, the minimum practical configuration would require 16K words.

The OS will probably require extra areas of memory, depending upon the size of the installation and the complexity of the applications; this may be for file definitions, actual device service routines, disk file buffer areas etc. The information as to where these areas are and also, for example, the number, location and type of peripherals to be used is transmitted to the OS by means of the Configuration Table discussed earlier in this section under SOFTWARE CONCEPTS.

Task Requirements

Each task has its own memory partition. All partitions are of the same size and format. Because the partition maps are pre-defined and known throughout the system, there is a considerable saving in the number of parameters required by subroutines. Experience has shown a 2K partition size to be the optimum choice for Commercial Applications, and all offsets quoted in this Manual will refer only to a 2K partition with a base offset of zero ie first word of the partition is offset 0000- and the last word of the partition is offset 3777-

Each task partition will comprise the following:

a) a PROGRAM AREA occuping 0000- to 3177-
b) a MASTER BUFFER occuping 3200- to 3377-
c) a TASK SPOOL BUFFER 3400- to 3577-
d) an INPUT BUFFER or PRINT BUFFER depending upon whether this is an I/O STATION or a print program and occupying 3600- to 3677- for an I/O STATION task, or 3600- to 3703- for a PRINT task
e) a CONTROL AREA occuping 3704- to 3777-

The PROGRAM AREA will contain program code, data and work areas. Program and Overlay Modules are loaded into this area by FETCH subroutines.

The MASTER BUFFER is used by the FETCH, and OVERWRITE sub-routines and is large enough to hold 128 words (equivalent to 1 sector of disk storage) of information.

The TASK SPOOL BUFFER is used by the SPOOL, UNSPOOL, POST, SPOOL AND POST subroutines. The input task creates spool records here; the print task receives spool records here. This buffer may be used as workspace (or as an extension to the MASTER BUFFER) when spooling is not required.

The INPUT BUFFER is used by the GET and SPLIT subroutines and contains the most recent input from the I/O STATION (ASCII format terminated by a Nul byte).

The PRINT BUFFER is used by the PRINTER OUTPUT subroutine which always space fills the buffer. The buffer holds one line of ASCII print-out; a map of print positions against offsets is provided in the Appendices.

The CONTROL AREA is reserved for System use.

System Workspace Requirements

The amount of system workspace required for different sites varies considerably depending on the particular configuration. See Utility "COMA" for workspace requirements calculation.

SECTION 2

MACHINE CODE PROGRAMMING

CONTENTS

	PAGE
PROGRAM INSTRUCTION FORMATS	2.1
MEMORY REFERENCE INSTRUCTIONS	2.3
Jump - unconditional	2.4
Jump - Sub-routine	2.4
Increment and Skip if zero	2.5
Decrement and skip if zero	2.6
'And' to A	2.7
'Inclusive or' to A	2.7
'Exclusive or' to A	2.8
Add to A	2.8
Add to B	2.8
Subtract from A	2.9
Subtract from B	2.9
Add to A with Carry	2.9
Add to B with Carry	2.10
Subtract from A with Carry	2.10
Subtract from B with Carry	2.11
Load into A	2.11
Load into B	2.11
Compare Store with A	2.11
Compare Store with B	2.11
Store A	2.12
Store B	2.12
ADDRESSING	2.13
REGISTER INSTRUCTIONS	2.15
Introduction	2.15
Mode 01 - Shift/Rotate Group	2.16
Mode 10 - Clear and Complement Group	2.19
Mode 11 - Alter/Skip Group	2.20
Mode 00 - Odd Instructions Group	2.22
INPUT/OUTPUT INSTRUCTIONS	2.26
Introduction	2.26
Instruction format	2.26
Input/Output	2.27
Program Interrupts	2.28
Data Channel	2.29

PROGRAM INSTRUCTION FORMATS

There are three types of basic instructions which are grouped according to the bit format of the instruction word. These types are:-

a) Memory Reference Instructions which deal mainly with the transfer of information to/from Accumulators A and B to/from the memory stores.

b) Register Instructions which deal with shifts, clears, rotates, negative tests, zero tests, etc on the Accumulators.

c) Input/Output Instructions which deal with the input and output of data to/from the peripherals, and include the Interrupt system.

A comparison of the three formats is given in Figure 2.1 below, and more detailed coding is included with the instruction descriptions following:-

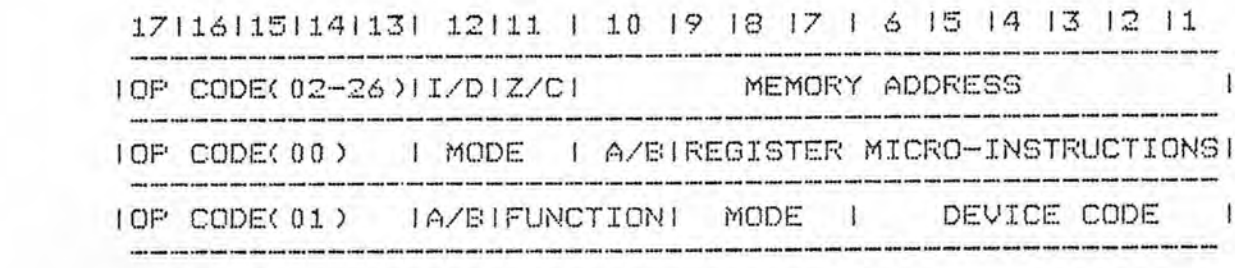


Figure 2.1

The first type comprises the Memory Reference instructions, using 10 bits (10 to 1) for a memory address, Bit 11 to specify Zero or Current page and Bit 12 for direct or indirect addressing. This leaves five bits (bits 17-13) to encode the 21 instruction commands in this group.

The other two types use these same five bits (17-13) to distinguish the Register and Input/Output Instructions, being all zeros for Register instructions and Bit 13 only being set for Input/Output.

The register type uses Bits 9 to 1 to combine in 'micro-instructions', with the resulting multiple instruction operating on the A or B accumulators as a single-word instruction. The Input/Output type uses Bits 12 to 7 for a variety of input/output instructions, and Bits 6 to 1 to make the instruction apply directly to one of the 63 possible input/output devices.

The following paragraphs describe in detail each of the instructions in the three groups. Functions of bits appearing in the form A/B, Z/C, I/D, L/R or T/F throughout these specifications are invariably obtained by coding a 1 or 0 respectively (1/0). Thus, for example, A is specified by a one-bit, and B by a zero bit. The following defines the abbreviations used:

A/B	Accumulator A/Accumulator B
Z/C	Zero Page/Current Page
I/D	Indirect/Direct
L/R	Left/Right
T/F	True/False

MEMORY REFERENCE INSTRUCTIONS

There are 21 Memory Reference Instructions which carry out some operation involving memory locations, such as transferring information in or out of a memory location or checking the memory location contents. The address referenced (ie. the absolute address) is determined by a combination of the task partition address, the ten memory address bits in the instruction word (10 to 1) plus two other bits (Bits 12 and 11). Bit 11 is reserved to specify either Page Zero or Current Page, and Bit 12 to specify direct or indirect addressing. The manner in which locations are specified for the Molecular 18 is discussed in detail under 'Addressing' later in this section.

All Memory Reference Instructions take a minimum of two machine cycles ('Fetch', to read the instruction word and 'Execute' to read the referenced memory location), except for JUMP, which takes a minimum of one machine cycle.

Note that since Accumulators A and B can be addressed, any Memory Reference instruction can apply to either of these registers, both directly and indirectly, as well as the Memory Stores. (Page zero must be specified for these operations, since the A and B register addresses, 0000 and 0001 are on Page Zero).

To summarise, a memory reference instruction uses a 10-bit address value to refer to a memory location, and then operates on the 17-bit binary number stored in the referenced memory location.

Mnemonic	Octal OP Code
JUMP	02
JSBR	03
INSZ	04
DESZ	05
ANDA	06
IORA	07
XORA	10
ADA	11
ADB	12
SFA	13
SFB	14
ADAC	15
ADBC	16
SFAC	17
SFBC	20
LDA	21
LDB	22
CMPA	23
CMPB	24
STA	25
STB	26

Figure 2.2

JUMP = Unconditional Jump

The JUMP instruction loads the effective address to the instruction into the Program Counter (PC), thereby changing the program sequence since the PC specifies the next instruction to be performed. It then takes the next instruction from that location and continues operation from there. The JUMP instruction does not affect the contents of the Accumulators.

In the following example, execution of the instruction in column 02 step 70 (JUMP to 0270) causes the program to jump over the instructions in Steps 21 through 67 and immediately transfer control to the instruction in column 02 step 70.

Location	Content
Current page, 0220	020270 (this instruction transfers program control to location Column 02, step 70)
Current Page, 0270	250300

JSBR = Jump to Subroutine

On this instruction the program will jump to the word of memory specified, which is assumed to have an initial value of zero. The value of the Program Counter (which is the address of the JSBR instruction + 1), in other words the return address, is always stored in the first location of the subroutine, replacing the original contents (over-writing).

After the subroutine is executed, the pointer address identifies the next instruction to be executed. Thus, programmers have at their disposal a simple means of exiting from the normal flow of the program to perform an intermediate task and a means of return to the correct location upon completion of the task. (This return is accomplished using indirect addressing, which is discussed later in this section). This also facilitates nesting of routines.

One can also alter the program within the program by changing the PC + 1 which is placed at the step at the beginning of the subroutine.

INSZ - Increment, and skip if zero

This instruction adds 1 to the contents of the word of memory specified, all 17 bits, and then examines the result of the addition. If the result is zero, the instruction following the INSZ is skipped and the Carry Flag will be set. If the result is not zero the program will proceed normally to be instruction immediately following the INSZ (the next word of program in sequence).
The following points should be kept in mind when using the INSZ instruction:-

1. The contents of the A and B accumulators are not disturbed, unless the INSZ instruction is used to increment either of the hardware working registers 0000 and 0001.

2. The original data word in the reference memory location is replaced by the incremented value.

3. The INSZ performs the increment first and then checks for a zero result.

4. When using the INSZ for looping a specified number of times, the tally must be set to the negative (twos complement) of the desired number.

5. The Carry Flag is set whenever a transfer occurs between Bits 16 and 17 in any store as a result of the INSZ instruction.

DESZ - Decrement, and Skip if Zero

This instruction subtracts 1 from the contents of the word of memory specified, all 17 bits, and then examines the result of this subtraction. If the result is zero, the instruction following the DESZ is skipped. If the result is not zero, the program will proceed normally to the instruction immediately following DESZ (the next word of program in sequence). Should the specified store overflow as a result of this instruction, the Carry Flag will be set.

The following points should be kept in mind when using the DESZ instruction:-

1. The Contents of the A and B Accumulators are not disturbed, unless the DESZ instruction is used to decrement either of the two hardware working registers 0000 and 0001.

2. The original data in the referenced memory location is replaced by the decremented value.

3. The DESZ performs the decrement first and then checks for a zero result.

4. The Carry Flag is set whenever a transfer occurs between Bits 17 and 16 in any store, as a result of this transaction.

   (eg. If a word contains Bit 17 only and is decremented, it will afterwards contain Bits 16 to 1 inclusive, and the Carry Flag will be set.)

ANDA = 'And' to A

The ANDA instruction causes a bit-by-bit Boolean AND operation between the Contents of Accumulator A and the contents of the word of memory specified by the ANDA instruction. The result is left in Accumulator A, replacing its original contents, but the word of memory specified is not altered.

The following points sum up the ANDA instruction:

1. A '1' is left in Accumulator A only when a '1' is present in the corresponding position of both Accumulator A and the specified word of memory (mask)

2. The Carry Flag is not affected, neither is the Greater Than flag as the operation is performed on a bit-for-bit basis.

3. The specified word of memory remains unaltered.

Acc A.	Store	Result in Acc. A
0	0	0
0	1	0
1	0	0
1	1	1

IORA = 'inclusive OR' to A

The IORA instruction causes a bit-by-bit Boolean OR (or Inclusive OR) operation between the contents of Accumulator A and the contents of the word of memory specified by the IORA instruction. The result is left in A, replacing its original contents, but the word of memory specified is not altered.

The following points sum up the IORA instruction

1. A '1' is inserted in Accumulator A if a 1 is present in the corresponding position of either Accumulator A or the operand.

2. The Carry Flag is not affected, neither is the Greater Than flag.

3. The specified word of memory remains unchanged.

Acc A	Store	Result in A
0	0	0
0	1	1
1	0	1
1	1	1

XORA = 'Exclusive OR' to A

The XORA instruction causes a bit-by-bit Boolean 'Exclusive OR' operation between the contents of Accumulator A and the contents of the word of memory specified by the XORA instruction. The result is left in Accumulator A, replacing its original contents, but the word of memory specified is not altered.

The following points sum up the XORA instruction

1. A '1' is inserted in Accumulator A only if the two corresponding bits of the Accumulator and the operand differ

2. The Carry Flag is not affected, neither is the Greater Than flag

3. The specified word of memory remains unaltered.

Acc. A	Store	Result in A
0	0	0
0	1	1
1	0	1
1	1	0

ADA = Add to A

The ADA instruction performs a binary addition between the specified data word and the contents of Accumulator A, all 17 bits, leaving the result of the addition in Accumulator A. The specified word of memory will remain unchanged, but the result of the addition could set the Carry Flag.

ADB = Add to B

The ADB instruction performs a binary addition between the specified word and the contents of Accumulator B, all 17 bits, leaving the result of the addition in Accumulator B. The specified word of memory will remain unchanged, but the result of the addition could set the Carry Flag.

SFA = Subtract from A

The SFA instruction performs a binary subtraction, subtracting the contents of the specified word of memory from the contents of Accumulator A, all 17 bits, leaving the difference in Accumulator A. The specified word of memory will remain unchanged, but the result of the subtraction could set the Carry Flag.

Arithmetically, binary numbers may be directly subtracted in a manner similar to decimal subtraction. The essential difference is that if a 'borrow' is required, it is equal to the base of the system of 2.

ie.

```
110   =  6  (decimal)
101   =  5
001      1
```

To subtract 1 from 0 in the first column, a borrow of 1 was made from the second column, which effectively added 2 to the first column. After the borrow, 2 - 1 = 1 in the first column; in the second column 0 - 0 = 0; and in the third column 1 - 1 = 0.

SFB = Subtract from B

The SFB instruction performs a binary subtraction, subtracting the contents of the specified word of memory from the contents of Accumulator B, all 17 bits, leaving the difference in Accumulator B. The specified word of memory remains unchanged, but the result of the subtraction could set the Carry Flag.

ADAC = Add with Carry (To Accumulator A)

Multiple register addition is possible using the ADAC instruction. This type of arithmetic is needed when a number that is too large to be contained in one word (ie. more than 65,535 in decimal) has to be added to another similar number, or when the result of an addition may be too large for a single register.

eg. Two numbers 65,535 and 65,537 are to be added together

```
65,535 = 00000  0000  0000  0000  01111  1111  1111  1111
65,537 = 00000  0000  0000  0001  00000  0000  0000  0001
```

Let us assume that the 65,535 is double stored in stores 0235 and 0236, the 65,537 is double stored in Stores 0251 and 0252, and that the answer is to be stored in 0276 and 0277. The first bit of program would read.

```
CLC          (Clear Carry)
LDA 0236     (load A with 0236)   01111  1111  1111  1111
ADA 0252     (Add to A 0252)      00000  0000  0000  0001
             (result in A)        10000  0000  0000  0000
```

As Bit 17 is left on, the presence of this bit denotes a negative number, therefore it must be cleared before being stored in the answer store:

```
CLSA        (Clear sign of A)
STA         (Store A in 0277)     0000  0000  0000  0000
```

The first register has been added, and because a carry occured between Bits 16 and 17 the Carry Flag will be set. Using ADAC for the addition of the second pair of stores, the 'with carry' will cause the Carry Flag to be added to the A register before the addition is started. The addition is then completed normally, and on completion the Carry Flag will no longer be set.

```
LDA 0235          00000   0000  0000  0000
                                          1 - Carry Flag
ADAC 0251         00000   0000  0000  0001

STA 0276          00000   0000  0000  0010
```

The total left in Stores 0276 and 0277 now equals:

00000 0000 0000 0010 00000 0000 0000 0000 = 131,072

To sum up, on an ADAC instruction, the contents of the Carry Flag (if any) will be added to the contents of Accumulator A, the Carry Flag will be cleared, and the contents of the word of memory specified will then be added to Accumulator A. The store specified will remain unchanged. This instruction is used for double or multiple store arithmetic.

NB: It should be pointed out that the Carry Flag could be set again by the addition (or subtraction).

ADBC - Add with Carry (To accumulator B)

Exactly as for ADAC, except that the contents of the Carry Flag (if any) will be added to the contents of Accumulator B, the Carry Flag will be cleared, and the contents of the word of memory specified will then be added to Accumulator B. The store specified will remain unchanged. This instruction is used for double or multiple store arithmetic.

SFAC - Subtract Store From A with Carry

The contents of the Carry Flag (if any) will be subtracted from the contents of Accumulator A, the Carry Flag will be cleared, and the contents of the word specified will then be subtracted from Accumulator A. The Store specified will remain unchanged. This instruction is used for double or multiple store arithmetic.

SFBC = Subtract Store from B with Carry

The contents of the Carry Flag (if any) will be subtracted from the contents of Accumulator B, the Carry Flag will be cleared, and the contents of the word specified will then be subtracted from Accumulator B. The store specified will remain unchanged. This instruction is used for double or multiple store arithmetic.

LDA = Load into A

LDA stores the contents of the referenced location in Accumulator A, over-writing the original contents of Accumulator A. The specified word of memory remains unaltered.

LDB = Load into B

LDB stores the contents of the referenced location in Accumulator B, over-writing the original contents of Accumulator B. The specified word of memory remains unaltered.

CMPA = Compare Store with A (skip if unequal)

This instruction compares the contents of the word of memory specified with the contents of Accumulator A, all 17 bits. If the two words are different the next instruction will be skipped. (ie. the PC is advanced by two instead of one). If both words are identical, the program will proceed normally to the next instruction in sequence. The contents of both the specified word of memory and Accumulator A remain unaltered.

Should the contents of Bits 16 to 1 of Accumulator A be greater than the contents of Bits 16 to 1 of the word of memory addressed, the Greater Than Flag will be set (but NOT the Carry Flag).

Should the contents of bits 16 to 1 of Accumulator A be less than or equal to the contents of bits 16 to 1 of the word of memory addressed, the Greater Than Flag will be cleared (but not the Carry Flag).

CMPB = Compare Store With B (skip if unequal)

This instruction compares the contents of the word of memory specified with the contents of Accumulator B, all 17 bits. If the two words are different the next instruction will be skipped. (ie. the PC is advanced by two instead of one). If both words are identical, the program will proceed normally to the next instruction in sequence. The contents of both the specified word of memory and Accumulator B remain unaltered.

Should the contents of Bits 16 to 1 of Accumulator B be greater than the contents of Bits 16 to 1 of the word of memory addressed, the Greater Than Flag will be set (but NOT the Carry flag).

Should the contents of Bits 16 to 1 of Accumulator B be less than or equal to the contents of Bits 16 to 1 of the word of memory addressed, the Greater Than flag will be cleared (but NOT the Carry Flag).

STA = Store A

The STA instruction stores the contents of Accumulator A in the word of memory specified, over-writing the original contents of the referenced location. Accumulator A remains unaltered.

It is not possible to STA indirectly into Accumulator B through another store.

STB = Store B

The STB instruction stores the contents of Accumulator B in the word of memory specified, over-writing the original contents of the referenced location. Accumulator B remains unaltered.

It is not possible to STB indirectly into Accumulator A through another store containing zero.

ADDRESSING

When the Memory Reference instructions were introduced, it was stated that the format was five bits for the operation code and the remaining twelve bits are allocated to specify the operand (the address referenced by the instruction). However, a full 16 bits are needed to uniquely address the 65536 locations that are contained in a 64K Molecular 18. To make the best use of the available twelve bits, the following formula has been adopted.

Bit 10 to 1 = Address Within Page

This is the address to which the rest of the instruction refers. The Memory Reference instructions can directly address any word in the Current or Zero Page (from 0 to 1023 in Binary).

e.g. Column 13 Step 24 is shown in the 10 address bits as follows:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	0

which, when read off in octal is 1324. Similarly, column 01 step 77 would be entered as Octal 0177 from Bit 10 downwards.

Bit 11 = Zero or Current Page Indicator

All Memory reference instructions include a bit (bit 11) reserved to specify either Page zero (the base page) or the Current Page (the page in which the instruction itself is located). These page references for addressing are specified by Bit 11 as follows:

1 = Zero Page (Z)
0 = Current Page (C)

To address locations in any other page, indirect addressing is used, via the current or zero page, as explained below.

Bit 12 = Indirect/Direct Address Indicator

All memory reference instructions include a bit (bit 12) reserved to specify direct or indirect addressing. Direct addressing combines the instruction code and the effective address into one word, permitting a Memory Reference instruction to be executed in two machine cycles (Fetch and Execute). Indirect addressing uses the address part of the Page which is taken as a new memory reference for the same instruction. This new address is a full 17 bits long, 16 bits of address plus another bit, Bit 17 which is used as a further Indirect/Direct bit. The 16 bit length of the address permits access to any location in memory, using the following method to work out the address

eg. for page 51, column 13, step 24 insert the column and step as described above. Then dividing the rest of the bits into groups of three from bit 11 upwards, insert the page in octal as below:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	0	1	0	0	1	1	0	1	1	0	1	0	1	0	0

Thus the address in octal is 123324.

Direct or indirect addressing is specified by Bit 12 (or Bit 17) as follows:

1 = Indirect
0 = Direct

Owing to the multi-programming aspects of the machine it is not practical to set up absolute addresses within a program. For complete flexibility it is recommended the offset address, as discussed in Section 1, be set up within any program. Such addresses, when resolved to absolute by the OS will have the format shown above.

REGISTER INSTRUCTIONS

Introduction

The register instructions in general manipulate bits in the A and B Accumulators. There is no reference to memory, thus these instructions are executed in only one machine cycle. They are combinable to form a one word multiple instruction that can operate in various ways on the contents of Accumulators A and B. These 'micro-instructions' are divided into four sub-groups, the Shift-Rotate Group, the Alter-Skip Group, the Clear and Complement Group and the Odd Instruction Group.

As shown previously the bit structure is made up as follows:

Bits 17 to 13	specify the operation code (all zeros)
Bits 12 and 11	specify the Mode
Bit 10	specifies the Accumulator
Bit 9 through 1	are the Micro-Instruction bits.

Micro-instructions may be combined under the following general rules.

1. No instruction may be used which combine micro-instructions from different Modes.

2. Reference to both A and B registers cannot be mixed in the same micro-instruction.

3. If more than one of the micro-instruction bits of a particular Mode is set at any one time, the sequence of execution is left to right (Bit 9 to Bit 1).

4. Use zeros to exclude unwanted micro-instruction bits.

5. If two (or more) skip functions are combined, the skip will only occur if all conditions are fulfilled. One exception exists - In Mode 01 only, the skip will occur if either or both conditions are met.

6. Shift and Rotate or Increment and Decrement must not be mixed in the same micro-instruction - the effect of doing so is undefined.

There are four modes - the mode alters the meaning of the micro-instruction bits.

Mode 01	(Shift/Rotate Group)
Mode 10	(Clear and Complement Group)
Mode 11	(Alter-skip Group)
Mode 00	(Odd Instruction Group)

Register instructions for modes 01, 10, 11

MODE	MICRO INSTRUCTIONS										
Bits: 12	11	10	9	8	7	6	5	4	3	2	1
0	1	A/B 1/0	Clear Carry	Left/ Right	Shift	Rotate	With Carry	Dec.	Inc.	Skip b16=0	Skip b1=0
1	0	A/B 1/0	Clear One's Accum	Clear Comp. Carry	Comp. Carry	Skip	Swap	Clear Sign	Comp. Sign	Enter S/reg	
1	1	A/B 1/0	True/ False	Skip Neg.	Skip Not 0	Skip Carry	Clear Carry	Skip If >	Clear >Flag	Clear Accum	One's Comp.

Mode 01 - Shift/Rotate Group

There are 21 basic instructions in this group which can manipulate the contents of Accumulators A and B and the Carry Flag; these instructions can be combined with other instructions also in Mode 01.

Mode 01 Instructions

Mnemonics	Octal Code
CLC	002400
LSA	003300
LSB	002300
RSA	003100
RSB	002100
LRA	003240
RRA	003040
LRAC	003260
RRAC	003060
LRB	002240
RRB	002040
LRBC	002260
RRBC	002060
DECA	003010
DECB	002010
INCA	003004
INCB	002004
AMSB	003002
BMSB	002002
ALSB	003001
BLSB	002001

If, for instance, it is desired to clear carry and left shift accumulator A, to perform this task the program could include the following instructions (given in both mnemonic and octal form).

```
CLC 002400
LSA 003300
```

Since the CLC and LSA instructions occupy separate bit positions, they may be used in the same instruction, thus combining the two operations into one instruction. This instruction would be annotated as CLC,LSA which is 003700 in octal. In this manner, many more Register instructions, separated by a comma, can be combined to make the execution of the program more efficient.

The operation of each individual instruction specified by Bits 10 to 1 is described below.

CLC = Clear Carry - causes the Carry Flag to be cleared

LSA or LSB = Left Shift A or B - Bits 16 to 1 of the specified accumulator will be shifted one place to the left. Bit 17 (ie. the sign bit) of the accumulator to be shifted remains stationery.

If there is a '1' bit in position 16 of the accumulator, the Carry Flag will be set after a left shift.

If there is a '0' bit in position 16 of the accumulator, the Carry Flag should it be set already, will not be overwritten after a Left Shift.

RSA or RSB = Right Shift A or B - Bits 16 to 1 of the specified accumulator will be shifted one place to the right. Bit 17 (ie. the sign bit) of the accumulator to be shifted remains stationery. If there is a '1' bit in Position 1 of the Accumulator, the Carry Flag will be set after a Right Shift. If there is a '0' bit in position 1 of the accumulator, the Carry Flag should it be set already, will not be overwritten after a Right Shift.

LRA or LRB = Left Rotate A or B - This instruction rotates Bits 16 to 1 of the specified accumulator one place to the left.

One left rotate will perform as for one left shift, except that the figure which was in Bit 16 will re-enter the Accumulator at Bit 1. The Carry Flag is not affected. In other words, it treats Bits 16 to 1 of the specified accumulator as a closed loop, and performs what is commonly called a circular shift, meaning that any bit rotated off the left end will re-appear at the right end.

e.g.	One left rotate of	0110	1100	0110	0101
	gives	1101	1000	1100	1010

RRA or RRB = Right Rotate A or B - This instruction rotates Bits 16 to 1 of the specified accumulator one place to the right.

One right rotate will perform as for one right shift, except that the figure which was in Bit 1 will re-enter the Accumulator at Bit 16. The Carry Flag is not affected. In other words, it treats Bits 16 to 1 of the specified accumulator as a closed loop, and performs what is commonly called a circular shift, meaning that any bit rotated off the right end will re-appear at the left end.

e.g. one right rotate of 0110 1100 0110 0101
gives 1011 0110 0011 0010

LRAC or LRBC = Left Rotate A or B with Carry - This instruction causes any previous Carry Flag to be introduced into the gap left by the rotate, and the bit rotated of the left end will replace the Carry Flag. In other words it treats Bits 16 to 1 PLUS THE CARRY FLAG as a closed loop, and performs a circular shift as previously described.

eg. 0110 1100 0110 0100 Carry flag is 1
after one left rotate with Carry would read:
11001 1000 1100 1001 Carry Flag is 0

Rotate with Carry is used for multiple store shifting.

RRAC or RRBC = Right Rotate A or B with Carry - This instruction causes any previous Carry Flag to be introduced into the gap left by the rotate, and the bit rotated off the right end will replace the Carry Flag. In other words it treats Bits 16 to 1 PLUS THE CARRY FLAG as a closed loop, and performs a circular shift as previously described.

eg. 0110 1100 0110 0100 Carry Flag is 1
after one right rotate with Carry would read:
1011 0110 0011 0010 Carry Flag is 0.

Rotate with Carry is used for multiple store shifting.

DECA or DECB = Decrement A or B - This instruction will decrement the contents of the specified accumulator by 1.

INCA or INCB = Increment A or B - This instruction will increment the contents of the specified accumulator by 1.

AMSB or BMSB = Skip if bit 16 of A or B equals zero - This instruction causes the program to skip the next step if Bit 16 of the specified accumulator is zero.

ALSB or BLSB = Skip if bit 01 of A or B equals zero - This instruction causes the program to skip the next step if Bit 01 of the specified accumulator is zero, or, in other words, if there is an even number in the accumulator.

Mode 10 = Clear and Complement Group

There are 15 basic instructions in this group which can clear and complement the contents of Accumulators A and B, and the Carry Flag; as before, these instructions can be combined with others also in Mode 10.

Mnemonics	Octal Code
CLA	005400
CLB	004400
CPLA	005200
CPLB	004200
CLC	004100
CMPC	004040
SKIP	004020
SWPA	005010
SWPB	004010
CLSA	005004
CLSB	004004
CPSA	005002
CPSB	004002
ESRA	005001
ESRB	004001

The operation of each individual instruction specified by Bits 10 to 1 is described below:

CLA or CLB = Clear A or B - set the specified accumulator (all 17 bits) to zeros.

CPLA or CPLB = Complement A or B - causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; that is, all ones become zeros, and all zeros become ones.

e.g before one's complement 01000 1100 1110 1111
after one's complement 10111 0011 0001 0000

CLC = Clear Carry causes the Carry Flag to be cleared.

CMPC = Complement Carry causes the state of the Carry Flag to be complemented (ie reversed).

SKIP = Unconditional Skip - causes the program to skip the next step, unconditionally.

SWPA or SWPB = Swap A or B - causes the contents of the top half (bits 16 to 9) of the specified accumulator to be swapped with the contents of the bottom half (Bits 8 to 1) of the said accumulator. The sign bit is not affected.

CLSA or CLSB = Clear Sign of A or B - causes the sign bit (bit 17) of the specified accumulator to be cleared (set to zero).

CPSA or CPSB = Complement Sign of A or B - causes the state of the sign bit (Bit 17) of the specified accumulator to be complemented (ie reversed).

ESRA or ESRB = Enter Switch Register into A or B - causes whatever is set on the Data Switches of the Control Panel to be loaded into the specified accumulator.

If it is desired to set the sign of an accumulator or to set Carry, obviously the Mode 10 instructions may be used.

CLSA, CPSA (005006 Octal) is equivalent to 'Set the Sign of A'.
CLC, CMPC (004140 Octal) will cause the Carry Flag to be set.

Mode 11 = Alter/Skip Group

There are 18 basic instructions in this group which perform tests on Accumulators A and B, Carry Flag, and the Greater Than Flag; the next instruction is skipped or not depending upon the results of the test. The group is subdivided into 2 sections. Within each section the instruction may be combined but between sections they may not.

Mode 11 Instructions - Section 1

Mnemonics	Octal Code
ANEG	007600
BNEG	006600
AN0	007500
BN0	006500
SK=C	006440
CLC	006020
S=GT	006410
CLGT	006004
CLA	007002
CLB	006002
CPLA	007001
CPLB	006001

ANEG or BNEG = Skip if A or B is negative - causes the next instruction to be skipped if the contents of the specified accumulator are negative (ie. if the sign bit is set).

AN0 or BN0 = Skip if A or B is not zero - causes the next instruction to be skipped if the contents of the specified accumulator are not zero (ie. if any of Bits 17-1 is set).

SK=C = Skip if Carry - causes the next instruction to be skipped if the Carry Flag is set.

CLC = Clear Carry - causes the Carry Flag to be reset (cleared).

S=GT = Skip if Greater Than - the next instruction will be skipped if the Greater Than Flag is set.

CLGT = Clear Greater Than - causes the Greater Than flag to be cleared.

CLA or CLB = Clear A or B - sets the specified accumulator (all 17 bits) to zeros.

CPLA or CPLB = Complement A or B - causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; all ones become zeros and all zeros become ones.

Mode 11 Instructions - Section 2

Mnemonics	Octal Code
APOS	007200
BPOS	006200
A=0	007100
B=0	006100
SKNC	006040
CLC	006020
SNGT	006010
CLGT	006004
CLA	007002
CLB	006002
CPLA	007001
CPLB	006001

APOS or BPOS = Skip if A or B is positive - causes the next instruction to be skipped if the contents of the specified accumulator are positive (ie. if the sign bit is not set).

A=0 or B=0 = Skip if A or B is zero - causes the next instruction to be skipped if the contents of the specified accumulator (all 17 bits) are zero.

SKNC = Skip if Not Carry - causes the next instruction to be skipped if the Carry Flag is not set.

CLC = Clear Carry - causes the Carry Flag to be reset (cleared).

SNGT = Skip if not Greater Than - the next instruction will be skipped if the Greater Than Flag is not set.

CLGT = Clear Greater Than - causes the Greater Than Flag to be cleared.

CLA or CLB = Clear A or B - sets the specified accumulator (all 17 bits) to zeros.

CPLA or CPLB = Complement A or B - causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; that is, all ones become zeros, and all zeros become ones.

Mode 00 = Odd Instruction Group

There are 25 basic instructions in this group; with the exception of multiple shifts and rotates and set Greater Than very few will be used by the applications programmer as this mode covers processor control instructions. These instructions are not combinable.

Mode 00 Instructions

Mnemonics	Octal Code
NOP	000000
HALT	000001
RSTN (I/O RESET)	000017
MASK	000002
ACKI	000003
ION	000004
IOFF	000005
SION	000006
SIOF	000007
SMOF	000010
SMON	000011
PRTY	000012
PTCT	000013
BNDY	000014
MASW	000015
CONT	000016
STGT	001001
SLLAnn	0017nn
SLLBnn	0007nn
RLMAnn	0016nn
RLMBnn	0006nn
SRLAnn	0015nn
SRLBnn	0005nn
RRMAnn	0014nn
RRMBnn	0004nn

NOP = No Operation - If all bits are zero then no operation is performed and program control is transferred to the next instruction in sequence. A subroutine to be entered by a JSBR instruction should have a NOP as the first step. The return address is then stored in the location occupied by the NOP during execution of the program. Also any program should be liberally sprinkled with NOPs to allow for flexibility in inserting subsequent amendments.

HALT = Halt - If Bit 1 is set and all other bits are zero, the computer will stop at the conclusion of the current machine cycle.

RSIN = I/O Reset - It must be emphasised that this instruction must not be used in program by application programmers. It is a useful instruction only on switches during debugging, etc., as it clears the Flags of all Input/Output devices connected to the computer. To perform an I/O Reset via switches, the following steps should be taken.

1) Load 000017 in octal on the data switches
2) Load into A (via switches), then set switches to zero
3) Depress Instruction Step Switch once.

MASK = Mask Out - This instructions sets up the Interrupt Disable Flags of each device, according to a pattern or mask set up in Accumulator A - each device's Interrupt Disable Flag is set or cleared as the corresponding bit in the Mask is 1 or 0.

The Mask Bit numbers refer to Data Channel Bits 1 to 17 numeric with ascending Binary order. Every device is wired to a particular data line on the in-out bus and hence to a particular bit of the mask. Although slower devices are assigned to the higher numbered bits in the mask, there is no established priority as the program can use any mask configuration.

ACKI = Acknowledge Interrupt - This instruction determines which is the highest priority device awaiting service by reading its device code into Accumulator B assuming an Interrupt has been called. If Accumulator B is zero in this case, it means an internal interrupt has been called. It can read the code of only one device at a time, whichever of those waiting has the highest priority. This is normally determined by the positions of the I/O Boards in the chassis, ie. the further from the processor, the lower the priority.

ION = Interrupt On - This instruction sets the Interrupt On Flag to allow the processor to respond to interrupt requests.

IOFF = Interrupt Off - This instruction clears the Interrupt On Flag to prevent the processor from responding to interrupt requests, and also prevents internal processor interrupts, such as:

Memory Boundary Interrupt
Mains Failure Interrupt
Mains On Interrupt
Memory Parity Interrupt
Memory Protect Interrupt
Memory Address = Switch Register Interrupt
Continuous Interrupt Switch Interrupt

SION = Skip if Interrupt On - This will skip the next instruction if the interrupt is on, enabling the processor to respond to interrupt requests.

SIOF = Skip if Interrupt Off - This will skip the next instruction if the interrupt is off, preventing the processor from responding to interrupt requests.

SMOF = Skip if Mains Failure Interrupt - This will skip the next instruction in sequence when an internal interrupt is called by a power failure. If the skip is taken, the Interrupt will be reset.

SMON = Skip if Mains on Interrupt - This will skip the next instruction in sequence when an internal interrupt is called when power is restored. If the skip is taken, the interrupt will be reset.

PRTY = Skip if Memory Parity Interrupt - This will skip the next instruction in sequence when an internal interrupt is called in the case of a memory parity failure. If the skip is taken, then interrupt will be reset.

PRCT = Skip if Memory Protect Interrupt - This will skip the next instruction in sequence when an internal interrupt is called by the program selecting an address in a protected section of memory. If the skip is taken, the interrupt will be reset.

BNDY = Skip if Memory Boundary Interrupt - This will skip the next instruction in sequence when an internal interrupt is called by the program specifying an address which is outside the memory. If the skip is taken, the interrupt will be reset.

MASW = Skip if MA=Switch Register - This will skip the next instruction in sequence when an internal interrupt is called when the memory address equals the address previously set on the Data Switches. If the skip is taken, the interrupt will be reset.

CONT = Skip if Continuous Interrupt Switch Interrupt - This will skip the next instruction in sequence when an internal interrupt is called after each step.

STGT = Set Greater Than Flag - causes the Greater Than Flag to be set.

SLLAnn/SLLBnn = Multiple left shift of Accumulators - causes Bits 16-1 to be left shifted n times where n is an octal number 1-17 (i.e. decimal 1-15). Has same effect on the Carry Flag as single shift instruction.

SRLAnn/SRLBnn = Multiple right shift of Accumulators - causes bits 16-1 to be right shifted n times where n is an octal number 1-17 (i.e. decimal 1-15). Has same effect on the Carry Flag as single shift.

RLMAnn/RLMBnn = Multiple left rotate of Accumulators - causes bits 16-1 to be left rotated n times where n is an octal number 1-17 (i.e. decimal 1-15). The Carry Flag or bit 17 of the register is not affected.

RRMAnn/RRMBnn = Multiple right rotate of Accumulators - causes bits 16-1 to be right rotated n times where n is an octal number 1-17 (i.e. decimal 1-15). The Carry Flag or bit 17 of the register is not affected.

INPUT/OUTPUT INSTRUCTIONS

Introduction

These are a set of program instructions which, like the Mode 00 Register Instructions, are not used in general programming, as all programmed transfers of information are accomplished by certain set subroutines which are supplied to the programmer, as described later in this manual. However, there now follows a description of the Input/Output instructions which control all transfers of data to and from the peripherals and also perform various operations within the processor. They provide the following general capabilities:

a) Fix the state of the Busy and Done Flags.

b) Test the State of the Busy and Done Flags.

c) Enter data from a specified device into the A or B registers.

d) Output data to a specified device from the A or B registers.

Input/output instructions are recognised by the computer when the four most significant bits of the instruction word are 0000 and Bit 13 is a 1 (Octal code 01). Bits 6 to 1 select the device that is to respond to the instruction; the format thus allows for 64 codes. In all input/output instructions Bits 11 to 7 specify the complete function to be performed; bits 11 and 10 either controlling or sensing Busy and Done, as shown under Function in the Selection Chart, (fig. 2.17). If Bits 9 to 7 are all set (Mode 111) there is no transfer, and Bits 11 to 10 specify a skip condition. Bit 12, where relevant, specifies A or B register (A=1, B=0).

Instruction Format

Bits	17-13	denotes the Operation Code 01
	12	denotes which Accumulator the instruction refers to 1= Accumulator A 0= Accumulator B
	11-10	denote the function bits; the functions are as follows unless Mode 111 is applicable (see below) 00 = no operation 01 = set busy, clear done (start device) 10 = clear busy, clear done (idle device) 11 = input/output pulse
	9-7	denote the mode
	6-1	denote the device code ie the means whereby a device is addressed

The modes available are as follows:

000- this has no I/O transfer.
001- DTI1 - enables input from register 1 of the specified device (each device can have up to 3 registers or buffers) into Accumulator A or B.
010- DTI2 - enables input from register 2 of the specified device, usually into Accumulator B; used to inspect status codes.
011- DTI3 - enables input from register 3 of the specified device into Accumulator A or B.
100- DTO1 - enables output to register 1 of the specified device from accumulator A or B.
101- DTO2 - enables output to register 2 to the specified device from Accumulator A or B.
110- DTO3 - enables output to register 3 of the specified device from Accumulator A or B.
111- Skip Mode - if this mode is set there is no I/O transfer and the function bits then specify a skip condition.

00 = skip if busy 01 = Skip if not busy
10 = Skip if done 11 = Skip if not done

Input/Output

Every peripheral device has up to three buffer registers, an Interrupt Disable Flag, Busy and Done Flags and a 6-bit device selection network. This selection network decodes Bits 6-1 of the Input/Output Instruction (which contain the device address), thus ensuring that only the current device responds to signals sent by the processor over the in-out bus. The Busy and Done flags together denote the basic state of the device. When both are clear, the device is idle. The overall sequence of Busy and Done states is determined by both the program and the internal operation of the device.

The data-in or data-out instruction that the program gives in response to the setting of Done can also restart the device. When all data has been transferred, the program generally clears Done so that the device neither requests further interrupts nor appears to be in use. (Busy and Done both set is a meaningless situation).

Any device whose Interrupt Disable Flag is set, cannot cause an interrupt to start and is, therefore, regarded by the program as being of low prioity. The Interrupt Disable Flags are used in setting up a priority structure which enables higher priority devices to interrupt an interrupt already in progress. This priority is determined by the use of a mask which controls the states of the Interrupt Disable Flags in the different devices.

Program Interrupts

The use of Input/Output Interrupts in the current program sequence allows apparently simultaneous operation of the main program and a number of peripheral devices. The hardware also allows conditions internal to the processor (e.g. mains failure, memory parity etc.) to signal the program by requesting an Interrupt.

Information transfer is taken care of, in general programming, by the Input and Output subroutines, described later in the manual. To place a device in operation, the program 'starts' it by the instruction 'Set Busy, Clear Done'. If the device is to be used for output (ie. the transfer of data from the processor to the peripheral) the program simultaneously gives a DTO1 instruction which sends the first unit of data - a word or character depending upon how the device handles information - from the specified Accumulator to Register 1 of the device. When it has processed the unit of data, the device clears Busy and sets Done to indicate that it is ready to receive new data for Output, or that it has Data ready for Input. If the Interrupt Disable Flag is clear, the setting of Done signals the program by requesting an Interrupt.

As soon as any Interrupt occurs, the processor stops the execution of the current program and begins to execute the Interrupt Control Routine. It does this by carrying out an automatic JSBR, by hardware, to zero page, Column 00, step 02. The result of this action is that the Program Counter, which will contain the address of the next instruction to be performed in the mainstream program, is stored in this location, and the program then transfers control to the next word in sequence (000003). This is an Indirect Jump to the next step, which holds the address of the Interrupt Control Routine (Page 01, Column 12, step 00). When this has been accomplished, the processor is said to have 'honoured' the Interrupt.

The Interrupt Control routine 'services' the Interrupt by:

a) saving the contents of Accumulators A and B.

b) saving the Program Counter.

c) saving the status of the Carry and Greater Than Flags.

d) saving the current memory bank no.

e) determining which device requires service (by the use of a look up Table) or whether it's an internal interrupt. The routine identifies the device by giving an Acknowledge Interrupt instruction.

f) servicing the device, which includes checking the status and either entering data from the buffer into A or B, or putting data from A or B into the buffer, depending on whether it's an Input or Output device.

Once the device has been 'serviced', the Interrupt is 'dismissed' by restoring the contents of Accumulators A and B, the Program Counter and the states of the Carry Flag, Greater than Flag and Memory Bank. The last step of the Interrupt Control routine is an Indirect Jump via zero page, column 00, step 02 (the Program Counter) so that the original program may be resumed at exactly the point the Interrupt occurred. During the 'dismiss' routine Interrupt is turned off so that no further Interrupts are enabled.

Data Channel

The processor contains a data channel which transfers data directly to and from memory by inserting a special memory cycle, rather than by interrupt to a service routine. The data channel is used by devices requiring very high data transfer rates, such as disk.

SECTION 3

GENERAL LIBRARY SUBROUTINES

CONTENTS

	PAGE
Add	3.1
Add to Double Word	3.2
Calculate Check Digit	3.3
Clear Block of Memory	3.4
Compare Block of Memory	3.5
Complement Double Word	3.6
Compute	3.7
Convert to ASCII	3.8
Convert to Binary	3.9
Convert I/P to Binary and Test	3.10
Convert from Metacode	3.11
Convert to Metacode	3.12
Convert to Negative if Postive	3.13
Convert to Octal Address	3.14
Convert to Octal Digits	3.15
Convert to Postive if Negative	3.16
Convert to Upper Case	3.17
Divide with Remainder	3.18
Divide by 10 with Remainder	3.19
Divide and Round	3.20
Divide by 10 Rounded	3.21
Duplicate Block of Memory	3.22
Extract System Serial Number	3.23
Jump to Subroutine at Offset	3.24
Jump to Subroutine at a Resident Overlay	3.25
Load A from Applications Workspace	3.26
Load A from Offset	3.27
Load Byte	3.28
Move (and Pad) Character String	3.29
Multiply	3.30
Multiply by 10	3.31
Name and Address processor	3.32
Pack Date	3.34
Resolve Block of Offset Addresses	3.35
Resolve Offset Address	3.36
Space Fill Block of Memory	3.37
Store A in Applications Workspace	3.38
Store A at Offset	3.39
Store Byte	3.40
Subtract	3.41
Subtract from Double Word	3.42
Swap Blocks of Memory	3.43
Unpack Date	3.44
Zero Test Block of Memory	3.45

NOTE DATA HANDLING

All addresses within parameter blocks should be specified as offset addresses unless it is required to access the zero page of memory and in this case only should absolute addresses be specified as appropriate.

ADD

JSBR IZ 1711 037711
P1 = Address of first value
P2 = Address of second value
P3 = Address of result
P4 = Word Length

Adds the second value to the first value and stores the sum in the result.

All three numbers must have the same word length (specified by P4). The two values are preserved, unless overwritten by the result. No error indication is given if overflow occurs.

ADD TO DOUBLE WORD

JSBR IZ 1713 037713
P1 = Address of Accumulator
P2 = Address of Value

The value is added to the Accumulator. Both are Double word numbers. No error indication is given if overflow occurs.

CALCULATE CHECK DIGIT

JSER IZ 1740 037740

Calculates the 'Check Digit' corresponding to the number in the A register on entry. On return, the A register contains the check digit in ASCII (the top byte of A is NUL).

The standard calculation routine returns a letter A - Z.

CLEAR BLOCK OF MEMORY

JSBR IZ 1710 037710
P1 = Address of Block
P2 = Word Length

Sets each word of block to binary zeros.

COMPARE BLOCK OF MEMORY

JSBR IZ 1723 037723
P1 = Address of first block of memory
P2 = Address of second block of memory
P3 = Word Length

Compares the first block word for word with the second block. If an inequality is detected the routine returns to the step following P3. If both blocks are identical, the routine skips this step on returning.

The greater than flag will be set or clear accordingly as P1 is greater than P2. The comparison is arithmetic. (ie. bit 17 of each word is interpreted as a sign bit). Thus positive/negative numbers may be compared. It should be noted that all sign bits are included in this check.

COMPLEMENT DOUBLE WORD

JSBR IZ 1750 037750
P1 = Address of Binary Number

Subtracts the Double Word Binary Number from zero. The result overwrites the original number.

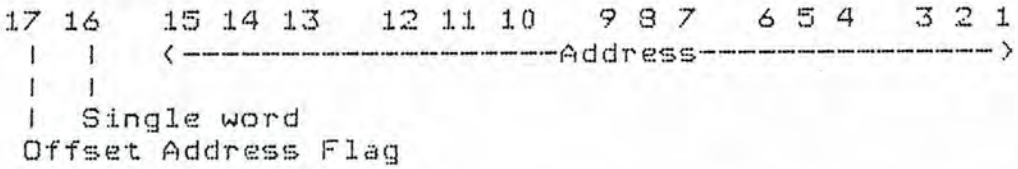
COMPUTE

JSBR IZ 1747 037747
P1 = Address and Word Length of Multiplicand
P2 = Address and Word Length of Multiplier
P3 = Address and Word Length of Divisor.
P4 = Address of result.

Multiplies the Multicand by the Multiplier and divides the intermediate result by the Divisor, storing the Result rounded to the nearest 1/2. If the Result is invalid for any reason (eg. Divisor is zero) the answer given will be zero. The Multiplicand, Multiplier and the Divisor are preserved unless overwritten by the Result.

Word Length may be specified as Single by setting bit 16 of the appropriate parameter; otherwise Word Length is taken as Double.

Parameter Format



Note that if absolute addressing is attempted using the full 64K memory capacity then up to 16 bits would be required for the address ie. there would be a conflict in the use of bit 16. For this reason, as will be found elsewhere, it is essential that 'offset addresses' are used which would require the use of bit 17 and bits 11 - 1 only. Where offset addresses are presented to subroutines the vital information (such as the value of bit 16 in this case) is extracted and saved before the address is resolved to an absolute address perhaps requiring bits 16-1 inclusive. Only if zero page is addressed should an absolute address be specified.

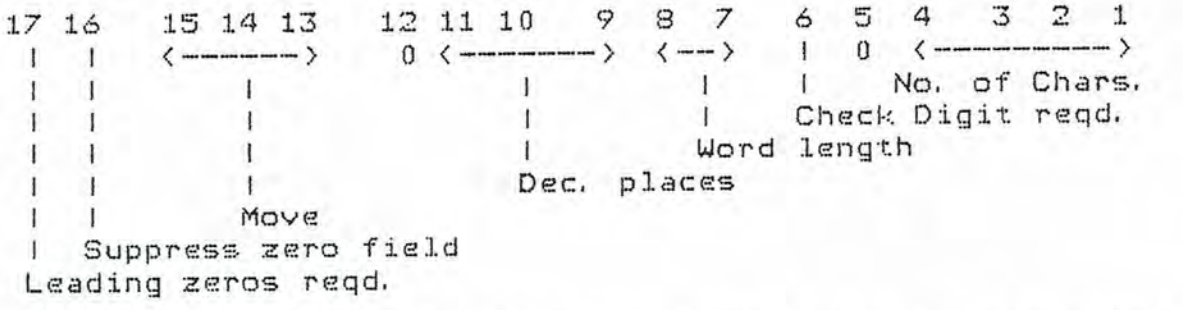
CONVERT TO ASCII

JSBR IZ 1765 037765
P1 = Control Word
P2 = Address of Binary Source
P3 = Byte Address of ASCII target

Converts the Binary Number into a fixed length decimal character string formatted as specified by the Control Word. The last character of the string will contain a hyphen if the number is negative, or a space or the 'check digit' (if requested) if the number is positive.

No error indication is given if the number is too large to fit the string.

Control Word



17 If 0, preceding zeros are replaced by spaces in the target string.

16 If set, leave target unchanged if binary source is zero.

15-13 Number of figures after the decimal point in the Binary source.

11-9 Number of figures after the decimal point in the Target (if 0, the decimal point is itself omitted from the string).

8-7 Word Length of the Binary Number. (NOT Zero)

6 If set, the routine will calculate the check digit corresponding to the number and insert it in the last character position.

4-1 Total length of the ASCII string, in characters, including the space, hyphen or check digit and (if non-integer) the decimal point.

The ASCII string must not overwrite the original binary number.

CONVERT TO BINARY

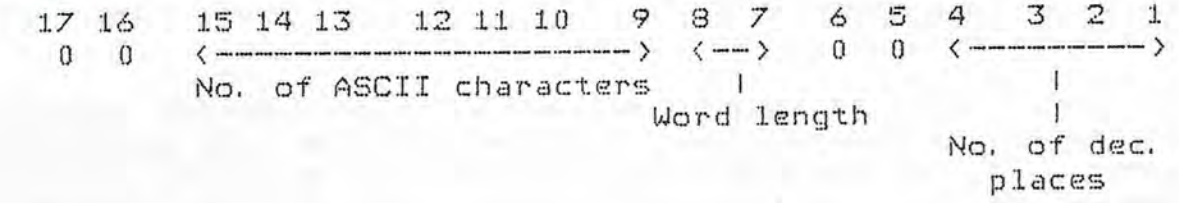
JSBR IZ 1762 037762
P1 = Byte Address of ASCII source
P2 = Address of Binary Target
P3 = Control Word

Converts the ASCII Source string to Binary as specified by the Control Word.

On return from this subroutine, Bit 17 of A register will be set if the number was too large to fit in the Binary Target. The B Register will contain the OFFSET Byte Address of the next character following the detected end of field. In the case of a variable length source, the end-of-field character will be in the bottom of the A Register.

Numeric characters are the digits 0 to 9, hyphen (Minus sign), period (decimal point) and comma (ignored).

Control Word



15-9 Set zero if the source string is variable length. Non-numeric characters within a fixed length source string are ignored except NUL which causes all following characters to be ignored.

8-7 Word Length must not be zero

4-1 Free format is allowed within the source string; any necessary adjustment to the precision is carried out automatically (excess digits are truncated).

CONVERT I/P TO BINARY AND TEST

JSBR IZ 1606 037606
P1 = Single Word Minimum Value
P2 = Single Word Maximum Value

Searches the ASCII Input Buffer for the Task and converts the next numeric field within it to a single Word Binary Number.

If the number is outside the limits specified by P1 and P2 the Error Handler is called ("ERROR" is sent to the Tasks I/O Station and upon acknowledgement the most recent "GET" subroutine call issued by the Task is restarted).

On return from this subroutine, the A register will contain the Single Word Binary Number extracted from the Input Buffer. The non-numeric character which terminated the field will be found in the bottom byte of absolute memory location 000043 (the top byte will be NUL); if this word is zero then the end of the source string has been reached.

The Source String may contain several numeric fields, each separated by a single non-numeric character, and successive calls to this subroutine will yield the next field (provided the Error Handler is not called).

If an attempt is made to extract a field after the source string has been exhausted, the Binary Number will be set to zero. If two non-numeric characters are adjacent in the Source String, an intervening zero field will be created. In both cases, control passes to the Error Handler if zero is outside the specified limits.

P1 must be positive or zero, P2 must not be less than P1.

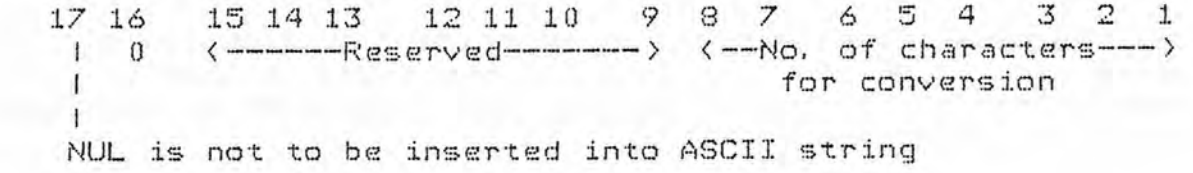
CONVERT FROM METACODE

JSBR IZ 1637 037637
P1 = Byte Address for ASCII
P2 = Address of Metacode
P3 = Control

Converts each Word of Metacode into three ASCII characters until either a NUL character is encountered in the Metacode or the character count is exhausted.

The Nul character (when encountered) is inserted into the ASCII string, unless Bit 17 of P3 is set.

Control Word



Bit 17 if set then NUL is not to be inserted into ASCII string.

16 indicates conversion to ASCII from Metacode.

8-1 maximum character length for conversion.

CONVERT TO METACODE

JSBR IZ 1637 037637
P1 = Byte address of Source ASCII
P2 = Address for Metacode
P3 = Control

Converts the source string to Metacode (a system of encoding three characters per 17-bit word) until either a NUL character is encountered in the source, or the source string is exhausted, or an unconvertable character is encountered in the source.

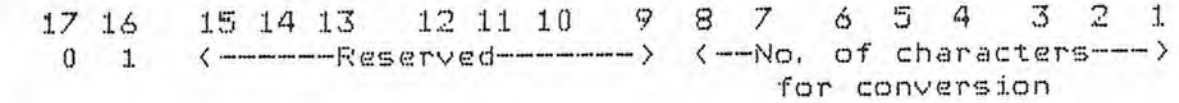
The NUL character (when encountered) is not inserted into the target.

The Metacode character set is restricted to a choice of 49 characters plus NUL. If an unconvertable character was encountered in the source string, the A register is non-zero on return from this routine.

The standard metacode character set consists of

26	Upper case letters	A to Z
10	numerals	0 to 9
13	symbols	Carriage return, space, inverted commas, percent, ampersand, apostrophe, open and close parenthesis, plus, comma, hyphen, period. oblique stroke.

Control Word



Bit 17 not applicable

16 indicates conversion to Metacode from ASCII

8-1 maximum character length

CONVERT TO NEGATIVE IF POSITIVE

JSBR IZ 1702 037702
P1 = Address of Binary Number
P2 = Word Length

Test the Binary Number and if positive subtracts it from zero.

On return the A register will be clear if, and only if, the number was already negative when the routine was entered.

CONVERT TO OCTAL ADDRESS

JSBR IZ 1605 037605
P1 = Byte Address of 9 character string

Converts the contents of the A register on entry into a 9 character string in the format 'PP/CCSS.-' where.

PP = Page number
CC = Column number
SS = Step number
',' = bit 16 set ie. bottom byte indicator
'-' = bit 17 set ie. indicates offset address.

Alternatively, if the address was interpreted as being absolute the format is PP/CCSS 9:

PP = Page number (00-77)
CC = Column number
SS = Step number
 9 = Current Bank number

CONVERT TO OCTAL DIGITS

JSBR IZ 1612 037612
P1 = Byte Address for Octal

Converts the contents of the A register on entry into octal digits stored in ASCII in the six character Target String. Leading zeros are not space-filled.

CONVERT TO POSITIVE IF NEGATIVE

JSBR IZ 1701 037701
P1 = Address of Binary Number
P2 = Word Length

Test the Binary Number and if negative subtracts it from zero.

On return, the A register will be clear if, and only if, the number was already positive (or zero) when the routine was entered.

CONVERT TO UPPER CASE

JSER IZ 1772 037772

Converts the character supplied in the A register to upper case if applicable. Accumulator B remains unchanged.

DIVIDE WITH REMAINDER

JSBR IZ 1732 037732
P1 = Address and word length of Dividend and Remainder
P2 = Address and word length of Divisor
P3 = Address and word length of Result

Divides the Dividend by the Divisor, storing the result and overwriting the Dividend with the Remainder. The Divisor is preserved. If the Divisor is zero and the remainder will equal the dividend.

The division is performed by a process of repetitive subtractions coupled with right shifts.

Parameter Format

```
17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
   <Words> <----------------Address---------------------->
```

17 Set for Relative addressing

16-15 Word Length of the parameter. If 0, single word length is assumed.

14-1 Address of Parameter

Note that if absolute addressing was attempted for any memory location above 16K then conflicts would arise in the use of bits 16-15. As with the COMPUTE subroutine discussed earlier it is therefore essential that offset addressing is always used with this subroutine, unless any of the addresses require to refer to zero page in which case absolute addressing should be used as appropriate.

IVIDE BY 10 WITH REMAINDER

BR IZ 1745 037745

'ides the contents of the A register by 10 and stores the ainder in the B register.

DIVIDE AND ROUND

JSBR IZ 1726 037726
P1 = Address and word length of Dividend
P2 = Address and word length of Divisor
P3 = Address and word length of Result.

Divides the Dividend by the Divisor and stores the result rounded to the nearest 1/2. The dividend and the Divisor are preserved. If the Divisor is zero, the result is set zero.

Parameter Format

As for 'DIVIDE with REMAINDER' subroutine.

DIVIDE BY 10 ROUNDED

JSBR IZ 1744 037744

Divides the contents of the A register by 10, rounding the result (5 goes up).

DUPLICATE BLOCK OF MEMORY

JSBR IZ 1707 037707
P1 = Address of Source
P2 = Address of Target
P3 = Word Length

The source is copied, word by word, to the target.

EXTRACT SYSTEM SERIAL NUMBER

JSBR IZ 1743 037743
P1 = Address of Target

The A register must contain the number code of the Serial Number to be extracted with bit 17 set if update is required.

The Value held within the file is deemed to be the 'next available' number. The routine deals with the reset condition (ie. when the serial number reaches the defined maximum value). The target address is defined as single or double word depending upon the defined maximum value (See Section 7 - Operating System Utility 'MAPS') and it is the programmers responsibility to ensure that the target area available within the program is adequate.

If a serial number is referenced within the range defined but it has not actually been created, a value of zero will always be returned.

JUMP TO SUBROUTINE AT OFFSET

JSBR IZ 1733 037733
P1 = Offset Address

This subroutine performs a 'JSBR Offset Address'. A parameter P2 to this subroutine will be interpreted as P1 to the target subroutine. The A and B registers are unchanged at entry to the target subroutine.

JUMP TO SUBROUTINE IN A RESIDENT OVERLAY

JSBR IZ 1632 037632
P1 = Address of location containing Resident O'lay no.

This subroutine performs a subroutine jump to the specified Resident Overlay, saving the Back Address in word 0 of the Overlay and resuming execution at word 1. A parameter P2 to this subroutine will be interpreted as P1 to the target subroutine. The A and B registers are unchanged at entry to the target subroutine.

LOAD A FROM APPLICATIONS WORKSPACE

JSBR IZ 1662 037662 -

The B-register must contain the word offset from the base of the Applications Workspace area (see Utility COMA), an offset of zero specifying the first word.

On return from the subroutine the A-register will contain the requested value and the B-register will contain the ABSOLUTE address of the selected word.

LOAD A FROM OFFSET

JSBR IZ 1721 037721
P1 = Offset Address

This subroutine returns to the step following P1.

The A register will contain the contents of the word addressed by P1 and the B register will contain the absolute value of P1.

LOAD BYTE

JSBR IZ 1771 037771
P1 = Byte Address of Operand

This subroutine returns to the step following P1.

Bits 8-1 of the A register will contain the byte addressed by P1 and Bits 16-9 will be NUL.

MOVE AND PAD CHARACTER STRING

JSBR IZ 1741 037741
P1 = Byte Address of Source String
P2 = Byte Address of Target String
P3 = Move Control Word

The Source string is copied, byte by byte, to the Target String until either a NUL Byte is encountered in the Source String or the Target String is exhausted.

The A register is non-zero on return if the Target String was exhausted without encountering a NUL Byte in, or immediately after, the Source String (this indicates truncation).

When a NUL byte is encountered in the Source String, the subroutine will perform the following as determined by the Control word:

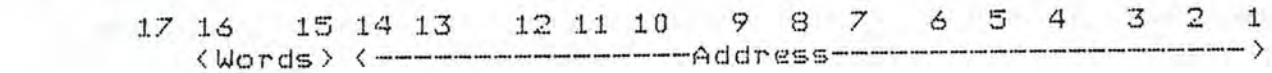
```
17 16  15 14 13  12 11 10  9  8  7   6  5  4   3  2  1
 |     <---Pad Character---> <-Target String Length->
 |       (NUL= Spaces)             (Characters)
 No Pad Required
```

MULTIPLY

JSBR IZ 1734 037734
P1 = Address and Word Length of Multiplicand
P2 = Address and Word Length of Multiplier
P3 = Address and Word Length of Result

Multiplies the Multiplicand by the Multiplier and stores the Result. If the Result is rendered invalid for any reason (eg. insufficient word length) the answer given will be zero.

Parameter Format



17 Set for Relative Addressing

16-15 Word Length of the parameter. If 0, single word length is assumed.

14-1 Address of Parameter

Offset addressing is essential unless it is required to access any area of zero page in which case the appropriate addresses must be absolute.

MULTIPLY BY 10

JSBR IZ 1761 037761
P1 = Address of Binary Number
P2 = Word Length

Multiplies the Binary Number by 10 and adds the contents of the A register at entry to the result of the multiplication.

Maximum word length is 3.

On return, the A register contains the overflow (if any) from the most significant word. The result overwrites the original Binary Number.

NAME AND ADDRESS PROCESSOR

JSBR IZ 1655 037655
P1 = Address of first parameter block
P2 = Address of Exit Routine
P3 = Number of calls to Exit Routine

Given any number of names and addresses, each held in ASCII in separate source strings, this routine creates an ASCII string containing the first line from each name and address. The routine then passes control to a user-written Exit Routine, which will normally perform an output operation and which must return control, whereupon the next line for each name and address is set up in the target string and the Exit Routine is called again. This process is repeated (provided the Exit Routine returns control) until the Exit Routine has been called the number of times specified by P3, whereupon control returns to the step following P3.

Each source string is ended with a NUL byte. Each line within each source string is ended with ASCII code 'CR' (optional on the last line). Each source string is allocated a field within the target string (the field position and length being user-defined) into which the next source line is copied byte by byte. The termination 'CR' is never stored in the target, neither is a NUL byte. If the target field becomes full, excess source characters are ignored. Each new source line is copied to the start of the appropriate target field, until a NUL byte indicates that the end of the source string has been reached.

The target string must be space filled initially and upon each return from the Exit Routine (NB Task Print Buffer is automatically space filled by the Printer Output Routine).

On entry to the Exit Routine, the A register will contain the Call Number (=1 first time). The Exit Routine will always be called exactly the number of times specified by P3. The Exit Routine may set up further fields within the target string before it is output; but in most cases it will simply output and return.

```
BA
DESZ Z A          First Time
LDA Z 0201        No therefore 1 line feed
JSBR IZ 1644      Print line (form feed on first time)
JUMP I BA         Return (to N/A Processor)
```

Parameter Block

There is one 5 - word parameter block for each source string, format as follows:-

P1 = address of next parameter block Zero if end of chain
P2 = Byte Address of Source ASCII
P3 = Byte address of Target ASCII field
P4 = Length (characters) of Target field
P5 = Workspace used internally as source pointer, does not require pre-setting.

PACK DATE

JSBR IZ 1607 037607

This routine uses the 'Convert Input to Binary and Test Limits' subroutine to extract a date from the Task input Buffer. The date must be input to the Buffer in ASCII as three numeric fields (day, month and last two digits of the year) separate by a single non-numeric character (usually a space).

If an error is detected in the input ASCII the Error Handler is called ("ERROR" is sent to the tasks I/O Station and upon acknowledgement the most recent "GET" subroutine call issued by the Task is restarted).

On return from this subroutine, the A register will contain a properly validated date packed in the format shown below and suitable for presentation to the "unpack" subroutine.

Format of Packed Date

```
17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
   <---------Year---------->  <--Month--> <----Date---->
```

RESOLVE BLOCK OF OFFSET ADDRESSES

JSBR IZ 1627 037627
P1 = Address of Block

Tests each word of the block for the presence of an Offset Address and, if found, overwrites the word with a resolved absolute address. The end of the block is indicated by a zero word.

Words within the block not containing Offset Addresses (ie. bit 17 not set) are unchanged.

RESOLVE OFFSET ADDRESS

JSBR Z 1630	033630
JSBR IZ 1630	037630

Tests the contents of the A register on entry for an Offset Address and if found, converts the address to absolute.

On return the A register contains the Absolute address. The B register is unchanged.

An Offset address is indicated by Bit 17 set.

SPACE FILL BLOCK OF MEMORY

JSBR IZ 1731 037731
P1 = Address of Block of Memory
P2 = Number of Words.

Sets each word of the block to ASCII spaces.

STORE A IN APPLICATIONS WORKSPACE

JSBR IZ 1663 037663

The B-register must contain the word offset from the base of the Applications Workspace area (see Utility COMA), an offset of zero specifying the first word.

On return from the subroutine the A-register will be unchanged and the B-register will contain the ABSOLUTE address of the selected word.

STORE A AT OFFSET

JSBR IZ 1725 037725
P1 = Address of Operand

The contents of the A register at entry are stored at the Word addressed by P1.

The subroutine returns to step following P1 with the A register unchanged and the B register containing the absolute value of P1.

STORE BYTE

JSBR IZ 1775 037775
P1 = Byte Address of the Operand

Bits 8-1 of the A register are stored at the byte addressed by P1. The A and B registers are undefined on return from this routine. It should be noted that Bit 17 of the word containing the byte is always cleared by this routine.

SUBTRACT

JSBR IZ 1712 037712
P1 = Address of first value
P2 = Address of second value
P3 = Address of result
P4 = Word Length

Subtracts the second value from the first value and stores the difference in Result.

All three numbers must have the same word length (specified by P4). The two values are preserved unless overwritten by the result. No error indication is given if overflow occurs.

SUBTRACT FROM DOUBLE WORD

JSBR IZ 1714 037714
P1 = Address to be Deducted from
P2 = Address of value to be Deducted.

Both parametes are addresses of double word locations.

SWAP BLOCKS OF MEMORY

JSBR IZ 1706 037706
P1 = Address of First block
P2 = Address of Second Block
P3 = Word length

The two blocks are swapped word by word.

A block of memory may be rotated by any desired number of words by swapping two suitably overlapping sub-blocks.

UNPACK DATE

JSBR IZ 1751 037751
P1 = Byte Address of ASCII Target

Converts the contents of the A register on entry into a 9 character string stored in the Target with the format:

dd mmm yy

where dd, yy are numeric and mmm is alphabetic.

The word presented in the A register should be a date previously packed by the 'Pack Date' subroutine.

ZERO TEST BLOCK OF MEMORY

JSER IZ 1724 037724
P1 = Address of block
P2 = Word Length

Tests the block, word for word with zero. If all words are zero, the routine returns to the step following P2. If a non-zero word is found, the routine skips this step on returning.

An equivalent but more efficient method for double words, providing the test applies to a memory Area on the Current Page, or zero page is

```
LDA  first word
IORA second word
AN=0
```

SECTION 4

DATA RETRIEVAL SYSTEM

CONTENTS

	PAGE
DATA-SET CONCEPTS	4.1
The System Catalogue	4.1
The System Tables File	4.1
Data File Definition	4.2
Data File Types	4.2
File Control Blocks	4.3
FILE HANDLING	4.4
Record Transfers	4.4
Direct Access Files	4.4
Shared Buffers	4.7
File Status Blocks	4.8
File Table Extensions	4.9
Indexed Sequential Files	4.10
Partially Linked Files	4.14
Introduction	4.14
Loading Records	4.14
Deleting Records	4.16
Disk File Updating	4.18
File Numbering Conventions	4.19
File Tables	4.20
DISK FILE SECURITY HANDLING	4.21
Disk Organisation	4.21
Disk Labels	4.21
Disk Layout	4.21
Disk File Security Procedures	4.22
Introduction	4.22
Security Program	4.22
Security Parameters	4.23
Recovery	4.23
FILE HANDLING SUBROUTINES	4.24
File Status Block Processor	4.24
Fetch - Direct Access (DA)	4.26
- Extended Direct Access (EDA)	4.28
- Indexed Sequential (IS)	4.30
- Sequential Access	4.31
- Program	4.33
- Overlay Module	4.34
Load record on Indexed Sequential File	4.35
Delete record from Indexed Sequential File	4.37
Indexed Sequential File Reorganisation	4.38
Overwrite	4.43
Rewrite	4.44
Stow Control Record	4.45

DATA-SET CONCEPTS

The System Catalogue

The System Catalogue is used to map out the available backing storage, thus allowing easy maintenance of disk-based data, by means of defining the basic layout and usage of 'Data-Sets'. These may be used to define Files (see later).
Associated with each Data Set Definition and included on the appropriate catalogue record will be a 12 character name.

The System Tables File

File accessing is always done via an octal file identifier; that identifier is then used to access a pointer to the details for that file, known as the File Control Block (FCB). All FCB pointers are in fact held as a 'File Table' within the System Table file, and each pointer is simply the Catalogue Record Number which holds the FCB details for that file identifier. The initiator will, upon boot-strapping, obtain the FCB details of all files from the Catalogue and establish them in free memory areas.

File identifiers are associated with the correct FCB details by use of a Programmer Utility known as MAPS i.e. Maintain APplication System(s). Within this utility it is possible to set up/amend the file table so that any slot reserved for a file identifier may have inserted into it the catalogue record number containing the FCB details. This is done by simply entering the file identifier and the Data-Set name with which it is to be associated. A search of the System Catalogue will then take place until a catalogue record is found containing the entered name. The number of that record within the catalogue is then inserted into the slot reserved for that file identifier within the file table.

When the system is bootstrapped, one of the functions performed by the Initiator when 'moulding' the Operating System is the transfer of the file table of FCB pointers from the System table file to a location in free memory. At this point all the catalogue records specified within the file table are read into various free memory areas. The memory address of each Catalogue Record (or FCB) is then inserted into the file table replacing the initial catalogue record number. Reference to any file via a file identifier will then enable the address of the FCB details for that file to be accessed. With the FCB details located, any part of the file may be accessed.

As part of any task it will often be necessary to enable input of parameters etc. as a result of which some form of report or output document will require to be printed; this will require the use of one of the printers available to the installation and so any I/O Station must be given the ability to drive a particular printer when necessary.

Facilities are available, via Program MAPS, to associate I/O stations with certain printers and/or preclude their use by other I/O stations.

Data File Definition

For the purposes of this manual a file is a random-access secondary storage medium into which programs may store, retrieve and update information contained within individual records. Programs may require many distinct files; one file may contain Customer Records, another Supplier records, another Product records. Each file is uniquely identified within a program by a 3 digit octal number in the range 000 to 377, and each record within a file is identified by a 'key' unique within that file. Records within the same file must all have the same length and the same type of key.

Data File Types

Two file types are available for Application use within the Operating System - direct access data files, and indexed sequential files.

Records within a direct data file are accessed using a record key which will correspond to the record slot number within the file eg. customer account 263 occupies the 263rd slot within the customer file. Records within an indexed sequential file are accessed using a record key which does not correspond to the record position within the files and resulting from this a two-tier index hierarchy is maintained consisting of:

a) A record key.

b) The slot number within the file of the record containing this key for each record currently in use on the file.

Although only two application file types are catered for, the Operating System offers considerable scope for the utilisation of alternative file handling schemes. A direct data file may, with varying degrees of programmer control be maintained in any of the following formats:

a) Direct Access.

b) Partially linked.

c) Sequential.

The file types and the methods of maintaining them are described later on in this section.

The indexed sequential file is, in reality, a composite of 3 files, namely a primary index, a secondary index and a data file. Access to any record within the data file may be gained via the indexes; however, the Operating System provides facilities to enable records within the file to be accessed sequentially, to access slots within the data file directly (although it must be stressed that these are simply record slots and the record key contained within them will be unknown, unless a more sophisticated file handling technique is employed by the programmer), to access slots within the secondary and primary indexes, or simply to access the secondary index entry for a record of a given key.

File Control Blocks

As stated in the previous section an FCB contains all the parameters which define a file, and during run time an FCB for each file used by the system (Application or Operating) will be permanently memory resident, having been set up in various 'free memory' areas by the Initiator upon boot-strapping the System.

It is of paramount importance that any file accessing be done indirectly via the file control block parameters as subsequent changes to that file e.g file extension, may then necessitate changes only to the FCB itself and not to any programs using it.

An FCB will consist of a variable number of parameters depending upon the type of file being accessed. Word 0 of an FCB is a mask defining the file type of which the following are currently recognised:

Mask	Type
201420	Undefined file.
201421	Binary Direct Access file ie. the record length can be expressed as a power of 2 enabling the Operating System to calculate the sector number more efficiently.
201422	Ordinary Direct Access file.
201423	Directory file containing program names.
201424	Overlay Module library File.
201425	Program file.
201426	File Table Extension (Hybrid).
201427	Indexed Sequential File.
201430	Extended Direct Access.

FILE HANDLING

Record Transfers

Transfer of records between primary and secondary storage (in practice always a disk cartridge) takes place, on the Molecular, only in 'sectors' of 128 words. Even if a record is less than 128 words long, a memory buffer capable of holding an entire sector is still necessary, and for this purpose all tasks are provided with a 'master buffer', one sector in length, which begins at step 3200- within the task partition. When records are held several to a sector the sector containing the required record is read into the Master Buffer by the "FETCH" subroutine. The calling program has the option to process the record directly in the buffer or to 'extract' it into a work area. When the buffer holds more than one record it is almost always best to have the required record extracted by the FETCH subroutine, because of the programming complications of accessing fields within records lying at different offsets within the buffer.

When space is at a premium, a technique used for enquiry on files is to extract the record to 3200- ie. the start of the Master Buffer. It will be appreciated that this practice is fatal if used when updating a file as irrecoverable file corruption will occur.

Direct File Access

The simplest type of key is a single word positive binary number (range 1 to 65,535) directly related to the physical position of the record within the file. Such a key identifies a record by its relative position, or 'slot number' within the file. The first slot is usually but not necessarily, numbered 1. A file with this type of key is called a Direct Access file (DA). It provides the fastest access to information in secondary storage, but may be prohibitively wasteful of storage space if too many of the slots are empty.

When slot numbers are allocated at random (by the user) you will need to be able to determine if a given slot is in use. The recommended method is to place the slot number into the first word of live records, since the system provides an automatic 'test' option for live records based on this principle.

Records within Direct Access files may be inter-related in many ways, and the slot numbers provide a convenient method of defining these relationships. For example, a product record may contain its suppliers record slot number and a supplier record may 'point' by slot number to one of its products which in turn points to another product, so that a logical 'chain' links all the supplier's products together (the end of the chain is indicated by a slot number of zero).

Records may be held several to a sector, but records exceeding one sector cannot be contained within the Task Master Buffer alone and in this case, it may be necessary to preserve the contents, if any, of the adjacent Spool Buffer prior to performing the access. Records which are less than 1 sector in length cannot overflow sector boundaries; whole records only will be held on each sector simply leaving any gaps at the end of the sector unused eg.

record length	=	14 words
records per sector	=	9
unused sector area	=	2 words

The contents of an FCB for a direct access data file will be as follows:

word	contents
0	Mask of 201421 if a Binary file. Mask of 201422 if an Ordinary File.
1	Bit 16.....Protected file. Bits 12-9.....No.of Sectors for transfer buffer. Bits 8-1.....Disk No.
2	Address of a Shared Buffer Control Block or zero if the record is to be read into the Task Master Buffer initially.
3	Minimum record number.
4	Maximum record number.
5	Record Length.
6	No. of records per transfer.
7	Absolute sector number of record no. 1 N.B. if the minimum record number is not 1, then this will NOT be the first sector of the file.

The FCB is created, by the OS, from a Data-Set Definition set up by a programmer utiltiy known as System CAtalogue Maintenance (SCAM) which requests the entry of a 12 character Data-Set name followed by the definition details. All of the details entered are written to the next free catalogue record.

Having set up the Data-Set Definition in a Catalogue record it is then necessary to use the programmer utility MAPS to associate a file identifier with the Data-Set details.

The number of records per transfer specified in word 6 of the FCB is straight forward for ordinary direct access files, but for binary files it will be expressed as follows

Number of Records Per Transfer	Equivalent Power of 2	Contents Of Word 6
1	0	0
8	3	3
32	5	5

The standard Direct Access file is capable of holding up to 65,535 records occupying a single disk extent. This is satisfactory for most applications, but occasionally two problems arise.

Firstly with the current range of disk media, the number of sectors on one disk extent is appreciably less than that required. For example a typical Customer record is one sector in length. This immediately forces a maximum of about 12,000 Customers within one Direct Access file when using the standard DD1600 disk drive.

Occasionally a particular file is required to have more than 65,535 records available. On some larger sites for example, the Sales Transaction file may be of this type. This file may or may not require more than a standard disk extent.

To overcome these problems, it is possible to create a number of Direct Access Data-Sets extending over different disks and then link them together to form a logical file of the desired proportions. This is known as an Extended Direct Access file (EDA).

The number of records allowed can be either in the range 1-65535, as a normal Direct Access file, or 1-131071 which still preserves the single word slot number but utilises Bit 17. Finally a full double word key can be specified although using this format means that any FSB processing required must be performed by the applications programmer.

The contents of the FCB for an Extended Direct Access file will be as follows:

word	contents
0	Mask of 201430.
1-2	Minimum record number.
3-4	Maximum record number.
5	Record Length.
6	Address of the first constituent Direct Access Data-Set FCB, the layout of which is the same as for standard Direct Access file, EXCEPT that an extra word (8) will point to the next constituent Direct Access Data-Set or be zero to indicate the end of the logical file.

Shared Buffers

The Task Master Buffer cannot contain records longer than 128 words, but a larger buffer may be established (by programmers) outside the 2K task partition.

Such a buffer belongs to the file NOT to a task, which must share the buffer with all other tasks. If a file has a Shared Buffer the FETCH subroutine will automatically use this buffer instead of the task MASTER BUFFER. Since all transfers of records in the file must pass through this single shared buffer it follows that all the records in the buffer are always completely up to date.

The FETCH subroutine can (and will) bypass the physical motions of reading a record from disk to memory if that record is already in the shared buffer (the REWRITE and OVERWRITE subroutines do not, however, bypass the act of writing the buffer back to disk).

The time thus saved can be substantial if the program has to read a number of records in slot number sequence, and this feature of the shared buffer is far more important in practical terms than its ability to contain records longer than the one-sector Task Master Buffer.

Files such as the key index of an Indexed Sequential File may be deliberately organised into slot number sequence to take advantage of this feature.

The existance of a Shared Buffer for any file is denoted by the contents of Word 2 within the FCB for that file. IF word 2 is zero then the task Master Buffer is used for disk transfer; otherwise word 2 contains the address of a Shared Buffer Control Block which is used to access the actual Buffer by the OS.

The following points are relevant with regard to the Shared Buffer:

a) The Buffer will be shared by all Tasks accessing this particular file.

b) To prevent another Task overlaying the buffer prematurely the O/S automatically LOCKS all transfers to/from a shared buffer.

c) On a disk read, if the Buffer contains the required information then no disk read takes place and the issuing Task retains control.

d) More than one file may share the same shared buffer provided they all reside on the same Disk Unit.

e) The LOCK operation will be broken by a SUSPEND and/or I/O routine call.

File Status Block

The file status block is a 4 word block containing the files' dynamic data. It may exist within the System Control Record, which should be written back to disk whenever the FSB is updated, or alternatively, be held within the Disk FSB file.

N.B.
The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial numbers are now disk based in separate reserved files.

Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

(A library subroutine, JSBR IZ 1665, exists to process the FSB for the above 4 word layout; see Disk Subroutine Library.)

The contents of the FSB are as follows:

Word	Contents
1	Count of live records
2	Count of chained records
3	First free chained record number
4	Chain Update Flag

All words of the FCB must be preset to zero when the data file is first allocated.

Chain files do not require preformatting into an empty chain.

File Table Extensions (Hybrids)

The primary funcion of a File Table Extension (FTE) is to overcome the restrictions imposed by a File Table capable of holding only 64 pointers to File Control Blocks. Before the introduction of the FTE it was possible to support a maximum of 64 files and not 256 as at present.

File Table Extensions are set up by use of the programmer utilities SCAM & MAPS; the contents of any such FTE will be as follows:

Word	Content
0	Mask of 201426
1	Address of FCB of file 0 nn
2	Address of FCB of file 1 nn
3	Address of FCB of file 2 nn
4	Address of FCB of file 3 nn

It is important to note that when SCAM is used to set up a FTE it will asssociate Data-Set names rather than file identifiers ie. at each step it will request the Data-Set names for what will be files 0xx, 1xx, 2xx and 3xx in that order. In other words, it assumes that where any Data-Sets are associated, then the last 2 digits of the file identifiers are common. For example, suppose the following Data-Sets are to be associated by a common root identifier of 20

i	rep names
ii	rep sales analysis
iii	rep commission analysis
iv	rep accounts

then by use of SCAM the Data-Set Definitions for each of these may be established in the catalogue along with some 12 character name.

(A library subroutine, JSBR IZ 1665, exists to process the FSB for the above 4 word layout; see Disk Subroutine Library.)

The contents of the FSB are as follows:

Word	Contents
1	Count of live records
2	Count of chained records
3	First free chained record number
4	Chain Update Flag

All words of the FCB must be preset to zero when the data file is first allocated.

Chain files do not require preformatting into an empty chain.

File Table Extensions (Hybrids)

The primary funcion of a File Table Extension (FTE) is to overcome the restrictions imposed by a File Table capable of holding only 64 pointers to File Control Blocks. Before the introduction of the FTE it was possible to support a maximum of 64 files and not 256 as at present.

File Table Extensions are set up by use of the programmer utilities SCAM & MAPS; the contents of any such FTE will be as follows:

Word	Content
0	Mask of 201426
1	Address of FCB of file 0 nn
2	Address of FCB of file 1 nn
3	Address of FCB of file 2 nn
4	Address of FCB of file 3 nn

It is important to note that when SCAM is used to set up a FTE it will asssociate Data-Set names rather than file identifiers ie. at each step it will request the Data-Set names for what will be files 0xx, 1xx, 2xx and 3xx in that order. In other words, it assumes that where any Data-Sets are associated, then the last 2 digits of the file identifiers are common. For example, suppose the following Data-Sets are to be associated by a common root identifier of 20

i	rep names
ii	rep sales analysis
iii	rep commission analysis
iv	rep accounts

then by use of SCAM the Data-Set Definitions for each of these may be established in the catalogue along with some 12 character name.

The data file will be set up as appropriate for the individual application and as described for direct access files. A 4 word file status block must be allocated so that the data file may be maintained in 'Free Chain' format. File status blocks are explained in the following section.

The secondary index will be set up as described for direct access files but specifically a minimum record length equal to the key length + 2 but preferably extended if necessary to be a Binary File for efficient access. A disk-based File Status Block must be allocated and a Shared Buffer Number must be specified. The number of records is usually at least equal to the maximum of data records allowed for.

The primary index will again be set up as described for direct access files and the record length will be at least equal to the key length + 2 and extended if necessary to be a Binary File for efficient access. No File Status Block is used, but a Shared Buffer Number must be specified. For efficient use, the shared buffer of the secondary index must be different from that of the primary index.

The size of the primary index will be chosen to produce an efficient access time. A rough guide is to create, where practical, a Primary Index such that each Primary element relates to as many Secondary Index records as would fit in its Shared Buffer.

ie: P = S/ST

Where:
P = Number of primary index records required
S = Number of secondary index records *
ST = Number of secondary index records per disk transfer

* The value of "S" is somewhat variable. It may be taken as the maximum number of entries allowed in the ISAM file eg 10,000 products. But it could be based on the expected or average number of records "live" during some application cycle, such as a sales transaction file having the capability of holding 60,000 entries but on average only containing 40,000 live records: ie. at month start there are only 15,000 entries, but by month end there are 55,000 entries. It follows therefore that this figure is closely related to the application in question and has to be estimated accordingly, perhaps by monitoring the state of the file over a period of work after the Customer's system has stabilized.

For example:

Data file size	:	2000 records
Secondary Index	:	2000 records
(associated Buffer holds 16 records/transfer)		
Primary Index	:	128 records(rounded)

The FCB for the indexed sequential access would be as follows:

Word	Contents
0	Mask of 201427
1	Data file identifier xxx (bit 17 set if none)
2	Secondary index identifier xxx
3	Primary index identifier xxx
4	Key length in words

Having set up 3 direct access files and a Catalogue entry to define an indexed sequential application by the use of SCAM, it is then necessary to associate the Catalogue records containing these details with the file identifiers used by the file accessing subroutines. Again, this is done by use of the MAPS utility. For example, if a customer file was to be accessed in indexed sequential mode then the following associations may be defined.

Catalogue Entry defining	File I/D
Customer Data File	11
Customer Secondary Index	12
Customer Primary Index	13
Index Sequential Definition	14

From this point onwards any subroutine specifying file identifier 14 would access via step 14 of the File table an IS type FCB. Within that FCB would be defined as the file identifiers of the 3 components which constitute the indexed sequential access. Access to those components would be achieved via steps 13, 12 and 11 (in that order) of the File table.

The problem with this approach to indexed sequential applications is that 4 valuable slots within the file table have been utilised in order to achieve access on, effectively, one file; the limitations imposed by the size of the File Table now become much more severe. In order to overcome these limitations use is made of the File Table Extension concept.

If the Catalogue entry defining the ISAM access was set up containing

```
Customer Data File I/D      ...  011
Secondary Index File I/D    ...  211
Primary Index File I/D      ...  311
Key length
```

and then a FTE definition was set up specifying:

```
For file 0xx  customer data-set definition
    file 1xx  IS type FCB indicating I/D's 11, 211, 311
    file 2xx  customer secondary index definition
    file 3xx  customer primary index definition
```

and then the FTE Catalogue entry name was associated with identifer 11 then only one slot of the File Table will be utilised; step 11 will contain the Catalogue record number of the FTE which will be resolved after Bootstrapping to the memory address of the FCB details.

Access on the customer file through the indexes and finally the data file for an IS would result in repeated entries to the FTE, the point of exit would be determined by the file identifier currently under consideration.

Partially Linked Files

1. Introduction

A partially linked file consists of several sets of records with each set having some unique association. The records of each set are 'chained' or 'linked' together by means of a pointer held on each individual record but there is no chaining between the sets of records. In short, the file consists of several sets of linked records, but it is not possible to link from the first record in the file through all other records in the file to the last; hence the term partially linked.

2. Loading records onto partially linked files

The association between records on a partially linked file is normally that they have a common source eg. a set of invoices have been sent by the same supplier, or a set of orders are all for the same product and so on. Therefore, when adding records to a partially linked file the following elements need to be considered.

a) Updating a live record count for the file and adding the new records to the file.

b) Chaining the additional records to the relevant sets.

c) Updating pointers on the source record so that from that source all of the associated records may be accessed.

Before describing the process involved in adding the new records to the file it is necessary to consider the set up of a File Status Block for the file. This may be assigned within the Control Record or in the Disk FSB File.

N.B.
The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial numbers are now disk based in separate reserved files.

Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

For a linked file the FSB would consist of a 4 word block as follows:

Word	Contents
1	Count of live records
2	Count of chained records
3	First free chained record slot
4	File update flag

(A library subroutine, JSBR IZ 1665, exists to process the FSB for the above 4 word layout; see Disk Subroutine Library. For completeness, a general description of FSB use follows.)

Word 1 would be incremented for each new record and the flag in word 4 would be set and unset as required. However, it is necessary to understand the significance of words 2-3 in order that they may be maintained correctly. In any file records may have been deleted; in OS applications those which have been deleted from a linked file will themselves be chained together as a 'free chain' to facilitate re-use of the slots. Therefore, if any gaps exist in a linked file word 3 will contain a pointer to the next gap and so on. The addition of a record to the file will result in the first gap being re-used and word 3 being amended to the next available gap for re-use. If the pointer was zero then all gaps would have been re-used and word 3 would be set to zero.

Word 2 may well reflect the total of live records which have been chained together in various associated sets, and the chained gaps; where word 3 is zero then word 2 will be equal to the number of live records in consecutive locations and hence the slot number of any new record will be the contents of word 2 incremented by 1.

Having established an additional record on the file then depending upon whether or not gaps existed in the file, either word 2 or the previous contents of word 3 of the FSB would indicate which slot the new record occupied within the file. It is then necessary to link that slot to the relevant associated step. Now contained on the source (master) record will be 2 pointers - the slot number of the first record of the associated set, and the slot number of the last record of the associated set. This last record's pointer is replaced by the slot number of the latest addition to this associated set. In this way the new slot is chained to the relevant associated set. It is vital the programmer ensures that the pointer on the new addition to the file is always zero to indicate end of set.

Having established the new record on file and linked it to the relevant associated set, it is now necessary to update the pointers on the source (master) record. Where an associated set existed for the master record then the first and last record slot pointers will be non-zero and it is simply necessary to replace the last record slot pointer on the master by the slot number of the latest addition to the associated set. In the case where there are no associated records for this master then the first and last record slot pointers would be zero; in this situation, both pointers would be replaced by the slot number of the new (and only) record of the associated set.

3. Deleting records from Partially Linked Files

Before describing the process involved in deleting records from the file it is necessary to consider the set up of a File Status Block for the file. This may be assigned within the Control Record or in the Disk FSB File.

N.B.
The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial numbers are now disk based in separate reserved files.

Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

For a linked file the FSB would consist of a 4 word block as follows:

Word	Contents
1	Count of live records
2	Count of chained records
3	First free chained record slot
4	File update flag

(A library subroutine, JSBR IZ 1665, exists to process the FSB for the above 4 word layout; see Disk Subroutine Library. For completeness, a general description of FSB use follows.)

Firstly it is necessary to remove the record from the chain of associated records and adjust where applicable the first and last record slot pointers on the master record. Three posibilities then arise.

1. The record deleted is the first of the associated set.
2. The record deleted is the last of the associated set.
3. The record deleted lies somewhere within the chain of associated records.

In the first case it is simply necessary to replace the first record slot pointer on the master record by whatever slot number the deleted record pointed to.

In the second case it is necessary to replace the last record slot pointer on the master record by the slot number of that record which had a pointer to the record being deleted; to ascertain that, it will be necessary to conduct a search through the associated record set using the 'next' pointers held on those associated records. Also, the 'next' pointer of the record slot found as a result of this search will be set to zero to indicate the end of this associated set.

In the third case the first and last record slot pointers on the master record would not be affected; however, the linking of records within the relevant associated set would be. The record slot which linked to the slot just deleted would have its 'next' pointer replaced by the 'next' pointer contained within the record slot deleted. In order to adjust the links in this manner it will again be necessary to find the record slot linking directly to the deleted slot by means of a search through the associated record set.

Having removed the record from the associated set, it must be linked into a chain of gaps.

When deleting records from a partially linked file the same 3 FSB elements need to be considered. The live record count in word 1 of the FSB would be decremented and the fact that an additional gap has been created in the associated file would be registered by the use of word 3 in the FSB. Where no gaps previously existed then word 3 would have contained zero; this would then be replaced by the slot number of the record which had just been deleted. Where gaps already existed then word 3 would contain the slot number of the first gap. This slot number would be written back to the record just deleted ie. it would be linked to the existing chain of gaps, and again word 3 of the FSB would be replaced by the slot number of the record which had just been deleted. It can be seen from this process that gaps in a file are re-used in the reverse order to that in which they were originally created.

Disk File Updating

In a multi-programming environment it will sometimes happen that, during the time it takes one program to read, update and write back a record, another program will require to update a record within the same sector. The second program uses the same read, update and write back sequence but unless its read is delayed until after the first program has written back, the first update will be overwritten. This delay is ensured by specifying the 'LOCK' option when calling the FETCH subroutine to read a record which is to be updated. The 'REWRITE' subroutine is used to write an updated record back to disk. The record involved is always that which has just been read by FETCH (with LOCK). It follows that a task can lock only one record at a time. If, and only if, the record was extracted by FETCH, it will be automatically re-buffered by REWRITE. There is an important safety precaution; REWRITE will halt the issuing task if it is called out of context, ie. if there is no record currently fetched and locked by the task.

The actual effect of the LOCK option is to prevent other tasks not just from reading the record involved, but from transfering any record in any file throughout the system; in other words the entire file access system is frozen until the lock is released. Normally, it will be the REWRITE subroutine that automatically releases the Lock, but all input/output library subroutines (including disk transfers and spooling), the 'SUSPEND' and 'HALT' subroutines automatically release the lock. Thus it is not possible for a task to retain a lock over a period of time; the record must be immediately updated and rewritten. In particular it is not possible to FETCH (with LOCK), GET an amendment from an input station and REWRITE. The correct procedure for amendments is FETCH (with TEST), GET, FETCH (with TEST and LOCK), update, REWRITE. At first sight this implementation of LOCK may seem somewhat drastic, but because updates must be performed immediately (without intervening input/output), the overriding pratical advantage is that no task has to wait for disk held data any longer than it takes to process any outstanding transfer requests.

It is possible and occasionally desirable, to prevent the updating of an entire file for a long period of time. Such a condition must be recognised and dealt with by the programs concerned, normally by use of the file update flag held in word 4 of the file status block.

File Numbering Conventions

All files are identified within a program by a 3-digit octal number in the range 000-377. File numbers x00-x10 are reserved for system use as follows:

000 - Spool File
100 - Print Queue Definitions
200 - Menu Notes
300 - System Serial Numbers
001 - Overlay Module Library Index - Master Library
101 - Overlay Module Library Index - Slave 1 Library
201 - Overlay Module Library Index - Slave 2 library
301 - Overlay Module Library Index - Slave 3 Library
002 - Overlay Module Library - Master
102 - Overlay Module Library - Slave 1
202 - Overlay Module Library - Slave 2
302 - Overlay Module Library - Slave 3
003 - I/O Station Program Directory - Master Library
103 - I/O Station Program Directory - Slave 1 Library
203 - I/O Station Program Directory - Slave 2 Library
303 - I/O Station Program Directory - Slave 3 Library
004 - Print Program Directory - Master Library
104 - Print Program Directory - Slave 1 library
204 - Print Program Directory - Slave 2 Library
304 - Print Program Directory - Slave 3 Library
005 - I/O Station Program Access
105 - Inhibit Flags/Passwords Recl=8 Min=1 Max=256
305 - System Tables File Recl=128 Min 1 Max 8
006 - Print Program Access
007 - System Catalogue - Data
107 - System Catalogue - I/S Access
207 - System Catalogue - Secondary index
307 - System Catalogue - Primary Index
010 - System Control File
110 - Disk FSB File Recl=8 Min=1 Max=256
310 - Spool Control File Recl=256 Min=1 Max=1

File Identifiers x11-x77 are available for application files. In order to set up and access index sequential files, certain conventions have been established with respect to the file numbers or file identifiers of the components of those files as follows:

a) The file identifier of the data file will be in range 11-77.

b) The file identifier of the secondary index will be in the range 211-277.

c) The file identifier of the primary index will be in the range 311-377

and in order to inform the file handling software that it is to search through a primary index, then a secondary index to access a record within a data file, the programmer must specify a file identifier in the range 111-177. Within the data file range, file identifiers 050-067 are normally reserved for the Purchase and Nominal Ledger package.

File Tables

Not only does the OS support a multi-programming environment but it is possible for information for more than one system (each with its own library of program modules and set of data files) to be processed simultaneously. This is accomplished by the use of File Tables. For any installation it is possible to allocate a file area which is used to contain one record for every system available at that installation; each record is a file table and defines a system in terms of program, catalogue and data files etc. The System Table File can contain as many records as it is possible to fit on the disk(s) at that installation.

Within the memory of the MOLECULAR 18 up to eight areas are reserved, each large enough to hold any 1 of the System file tables available at the installation. By use of the command Tn (See OPERATING SYSTEM COMMANDS) it is possible to associate any I/O Station with any of the file table areas reserved in memory. By calling the MAPS utility (See OPERATING SYSTEM UTILTIES) it is possible to load any available system ie. the file table for that system, into the file table area in memory now associated with the I/O Station. Obviously, where more than one I/O Station has been associated with a file table area within memory, they will be able to load only the same system and any attempt to do otherwise will be treated as an error.

Once the file table from the System Table file has been loaded into its file table area within memory then the catalogue for that system is accessed and all FCB's are loaded into free areas of memory. Processing may then commence.

DISK FILE SECURITY HANDLING

Disk Organisation

1. Disk Labels

Each disk is identified to the Operating System by a disk number and for programming convenience the number is always given in octal in the range 010-377. The disk number is held in the first word of sector 40 (octal) of the disk and this sector is therefore called the Disk Label. The second word of the label contains the number of the first unprotected sector on the disk (protection is a software feature designed to intercept any inadvertent write onto 'read only' areas of the disk - the Operating System provides only one such area per disk). For further information on the disk label see DLU utility - Section 7.

As far as numbering any particular disk is concerned the following conventions should be observed.

a) Disk numbers 300-377 are reserved for development and systems purposes, e.g. temporary fixed disk labels.

b) Disk numbers 010-277 are available for use by the installation; the first 2 digits identify the application of this disk and the 3rd digit the type as follows:

0	=	Master Disk
1-5	=	Copy 1 - Copy 5 of the master
6	=	Special Security
7	=	Working version of master disk existing on a fixed drive.

In order to establish a label on a disk it is necessary to use the Disk Labelling Utility (DLU) described in Section 7.

2. Disk Layout

622×32×2

A DD9600 disk has 39808 sectors numbered in octal from 0-115577; certain areas are reserved as follows:

Start Sector	No of sectors	Description
000000	1	Bootstrap (1)
000001	31	Operating System (1)
000040	2	Disk Label
000042	3	Configuration Table
000045	1	Bootstrap (2)
000046	2	Reserved for Operating System
000050	96	Operating System (2)
115400	128	Reserved for Operating System

Disk File Security Procedures

1. Introduction

Disk security facilities are acheived by setting up relevant details in each master disk label. (See DLU utility). Up to 7 copies can be maintained on a rotational basis as determined by a copy sequence number also held on the disk label. In fact, two copy sequences may be maintained simultaneously. One for daily security where each master disk will be copied at frequent intervals onto 2 or more security disks and the other sequence at critical stages during the users processing (i.e. for holding end of month details).

2. Security Program

The security module runs in an input station task partition and can be called by any user program containing a small parameter block. This parameter block is normally unique to each user and, apart from the disk labels, is all that has to be set up by the programmer in order to run the security routine.

There is no control by the programmer or user as to which device/drive is to be used and the security module will always try to use the optimum method of copying (i.e. from one drive to another of the same type). In order to achieve this all disk drives need to be up to speed and correctly labelled. If this is not possible then the fixed disk will be used and the security module will either check that the fixed disk has been previously copied or force the fixed disk to be copied first. After the fixed disk has been used then the data on the fixed will be recovered.

The security disks will always 'remember' the last method of copying (i.e. via fixed disk or from one drive to another etc.) so that if a disk/system has to be recovered then the security procedure will be reversed.

Where status errors exist on a disk then the security module will highlight them. Upon encountering a status error the OS will continually retry the disk transfer and every 20 seconds display a status error message at all I/O tasks. If the status is not cleared, then by setting switch 13 on the control panel the OS will only try the transfer twice before ignoring the status and continuing to the next transfer. Please note that this will normally invalidate the security for that disk and other suitable measures should be taken.

3. Security Parameters

An input program must be set up to contain the following parameter block. On completion of the security routine control will not be passed back to this program but to the OS prompt PROGRAM?

```
JSER IZ 1670                                        037670
P1   Link to overlay library module                  12002
P2   Address of module no                              264
P3   Byte address of program title
P4   Byte address of password
P5   Daily/Special security ind. & No. of masters
P6   Master disk no.
P7   Master disk no.
P8   etc.
```

Parameter 5

BIT	17	=	0	= Daily Security
BIT	17	=	1	= Special Security
BITS	6-1	=	The number of master disks to be copied in this run to a maximum of 32.	

Parameter 6 etc. may be repeated as required by the number of master disks as specified in P5.

4. Recovery

It is the user's/programmer's responsibility to ensure that the disks/system are recovered correctly with the aid of the DLU and RECY utilities.

The disk containing the operating system etc. will always be recovered as part of the bootstrapping procedure. At single drive sites it is probable that both exchangeable and fixed disks would be recovered as part of the bootstrap procedure. Thereafter each data disk may be recovered by reversing the security procedure using the utility RECY (see UTILITIES SECTION).

FILE HANDLING SUBROUTINES

N.B.
All subroutine parameters expressing memory addresses should, as a rule, use ONLY Offset Addresses.

FILE STATUS BLOCK PROCESSOR

JSBR IZ 1665 037665
P1 = Address of Control block

This routine deals with processing of the standard FSB, which is:

Word 1 - Count of live records
Word 2 - Count of chained records
Word 3 - First free chained record I/D
Word 4 - Update flag

Control block layout:

P0 = 'File Full' return address or Zero
P1 = Options, File I/D
P2 = Address of Key Area
P3 = Address of 'Extract' Area

Options:

```
17 16  15 14 13  12 11 10  9  8  7   6  5  4   3  2  1
 |  |   |  |  |   | <---> |  <--------File I/D-------->
 |  |   |  |  |   |   |     Protection Overide
 |  |   |  |  |   |  Reserved
 |  |   |  |  |  Free chain is maintained via the FIRST
 |  |   |  |  |  word of the specified file (default is
 |  |   |  |  |  using the LAST word)
 |  |   |  |  Insert into the First word of the Extract
 |  |   |  |  area the Record I/D of the Next Free Record
 |  |   |  |  before writing new record (used with Bit 14)
 |  |   |  Release for use Next Free Record, overwriting
 |  |   |  with contents of Extract area (P3). Record I/D
 |  |   |  will be returned in the Key area (P2)
 |  |  Return to Free Chain record nominated by the Key
 |  |  area. Record will be cleared except for Free Chain
 |  |  Pointer (First or Last word)
 |  Leave File Update Flag set on return to Calling Program
 Clear Update Flag only. Required if Bit 16 option has been
 previously used
```

When releasing records for use, the routine automatically checks the amount of free space remaining and, if less than 10%, warns the operator accordingly. If the file subsequently becomes full, parameter P0 may be used to define a Return Address to deal with this condition. Otherwise, if P0 is zero, the routine will Halt at 00/1377 with the A register containing the Back Address and B the above Options word.

N.B.
The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial Numbers are now disk based in separate reserved files.

Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

For Systems which must keep the Control Record a better method of use is to Fetch & Lock the Control Record (File 010, record 1) followed by the relevant updates and a REWRITE subroutine call.

FETCH - Direct Access File

JSBR IZ 1670 037670
P1 = Control Word
P2 = Address of Record Number
P3 = Address of "Extract" Area

The record identified by the File Identifier (specified in P1) and relative record number within the file (a single word binary field pointed to by P2) is read into the transfer buffer for the file (usually the issuing task's MASTER BUFFER).

If, and only if, the TEST option is specified in P1, the first word of the record is compared with the relative record number (pointed to by P2). If not equal, the subroutine returns to the step following P3 ('record not loaded'). If equal, the subroutine skips this step on return ("record loaded").

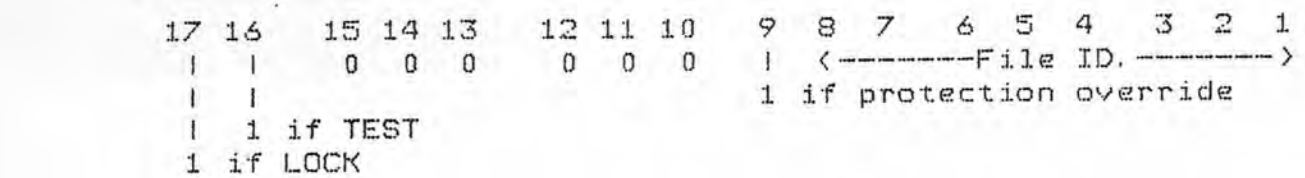
When the TEST option is not specified, the subroutine always returns to the step following P3.

The record may be automatically de-buffered by specifying in P3 the address to which it is to be copied (this process is referred to as "extraction"). If extraction is not required, P3 must be zero. Extraction does not take place if the record is found to be "not loaded" by the TEST option.

If the issuing task is fetching the record in order to perform an update, the LOCK option must be specified in P1.

On return from FETCH, 00/0151 points to the first word of the record within its transfer buffer, and 00/0152 contains the logical record length.

Control Word



17 Lock instructs the system not to initiate any disk transfers on behalf of any task once the record arrives in memory, until the issuing Task releases the "LOCK" (which occurs automatically upon next entry to the Task Scheduler followig return from JSBR "FETCH")

15-10 Reserved for internal use; must be pre-set 0 as shown.

9 Enables update of a data file flagged as 'protected'

NOTES.

1. Task HALTS at 00/1374 if the file identifier is undefined (A reg contains FETCH back address).

2. Task HALTS at 00/1377 if the Record number is outside the file limits (A reg contains FETCH back address).

3. If the disk containing the required record is not on-line a message to this effect is "flashed" and the task will be held waiting until the disk comes on-line.

4. If a hardware status is reported from the disk controller during FETCH, the system will re-try the transfer anticipating that the fault will clear. A message will be "flashed" at suitable intervals whilst the fault persists or until the disk is taken off-line.

5. Notes 3 and 4 above apply to all disk transfers by all subroutines within the OS.

FETCH - Extended Direct Access File

JSBR IZ 1670 037670
P1 = Control Word
P2 = Address of Record Number
P3 = Address of "Extract" Area

The record identified by the File Identifier (specified in P1) and relative record number within the file (a single or double word binary field pointed to by P2) is read into the transfer buffer for the file (usually the issuing task's MASTER BUFFER).

If, and only if, the TEST option is specified in P1, the first word of the record is compared with the relative record number (pointed to by P2). If not equal, the subroutine returns to the step following P3 ('record not loaded'). If equal, the subroutine skips this step on return ("record loaded").

When the TEST option is not specified, the subroutine always returns to the step following P3.

The record may be automatically de-buffered by specifying in P3 the address to which it is to be copied (this process is referred to as "extraction"). If extraction is not required, P3 must be zero. Extraction does not take place if the record is found to be "not loaded" by the TEST option.

If the issuing task is fetching the record in order to perform an update, the LOCK option must be specified in P1.

On return from FETCH, 00/0151 points to the first word of the record within its transfer buffer, and 00/0152 contains the logical record length.

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
  |  |    0  0  0    0  0  0   |  <-------File ID.-------->
  |  |                         1 if protection override
  |  1 if TEST
  1 if LOCK
```

17 Lock instructs the system not to initiate any disk transfers on behalf of any task once the record arrives in memory, until the issuing Task releases the "LOCK" (which occurs automatically upon next entry to the Task Scheduler followig return from JSBR "FETCH")

15-10 Reserved for internal use; must be pre-set 0 as shown.

9 Enables update of a data file flagged as 'protected'

NOTES.

1. Task HALTS at 00/1374 if the file identifier is undefined (A reg contains FETCH back address).

2. Task HALTS at 00/1377 if the Record number is outside the file limits (A reg contains FETCH back address).

3. If the disk containing the required record is not on-line a message to this effect is "flashed" and the task will be held waiting until the disk comes on-line.

4. If a hardware status is reported from the disk controller during FETCH, the system will re-try the transfer anticipating that the fault will clear. A message will be "flashed" at suitable intervals whilst the fault persists or until the disk is taken off-line.

5. Notes 3 and 4 above apply to all disk transfers by all subroutines within the OS.

6. The Test option is not available with double word keys.

FETCH - INDEXED SEQUENTIAL

JSBR IZ 1670 037670
P1 = Control Word
P2 = Address of ASCII key (with preceding single word workspace).
P3 = Address of Extract Area

The record identified by the File identifier (specified in P1) and ASCII key (pointed to by P2), if such record exists, is read into the file's transfer buffer (usually the issuing task's MASTER BUFFER).

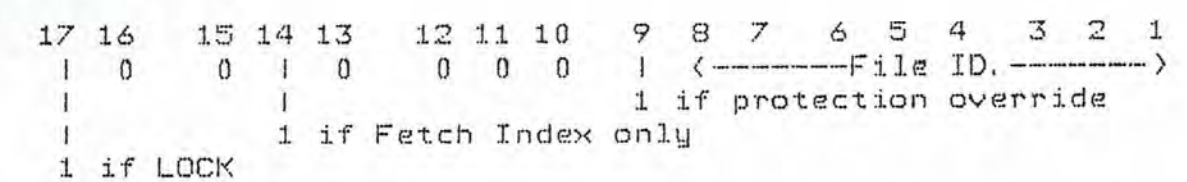
If the record does not exist, the subroutine returns to the step following P3 ("record not loaded") otherwise the subroutine skips this step on return ("record loaded").

A record may be automatically de-buffered by specifying in P3 the address to which it is to be extracted. If extraction is not required, P3 must be zero.

If the issuing task is fetching the record in order to perform an update, the LOCK option must be specified in P1.

On return from FETCH, 00/0151 points to the first word of the record within the transfer buffer, 00/0152 contains the logical record length and 00/0153 contains the record number within the data file (or index if 'index-only' Fetch requested). The ASCII key is unchanged, but the word preceding the key contains a pointer to the next record in the logical collating sequence (alphabetic key order).

Control Word



NOTES

1. Byte addressing is not allowed on P2 above, and short record keys must be space filled where necessary to the full key length specified in the File Control Block otherwise a 'record not loaded' condition will occur.

2. If a given key contains a NUL byte then a 'record not loaded' condition will always occur.

FETCH - SEQUENTIAL ACCESS

JSER IZ 1670 037670

P1 = Control Word
P2 = Address of ASCII Key Return Area (With preceding single-word workspace)
P3 = Address of Extract Area

The next record in the logical collating sequence (alphabetical order) within the file identified in P1, if such a record exists, is read into the file's transfer buffer (the current position in the collating sequence is defined within the workspace preceding the key area pointed to by P2).

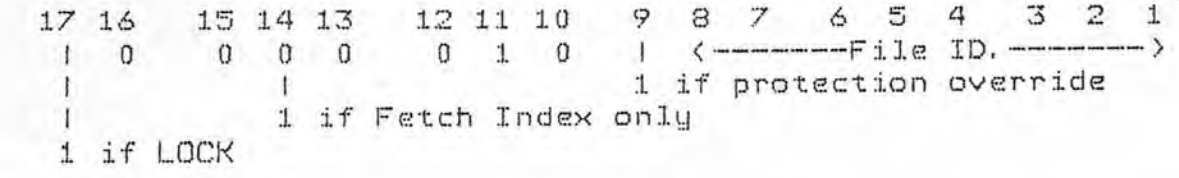
If no further record exists, the subroutine returns to the step following P3 ("end of file") otherwise the subroutine skips this step on return ("record found").

The record may be automatically de-buffered by specifying in P3 the address to which it is to be extracted. If extraction is not required P3 must be zero.

If the issuing task is fetching the record in order to perform an update, the LOCK option must be specified in P1.

On return from FETCH, 00/0151 points to the first word of the record within the transfer buffer, 00/0152 contains the logical record length, and 00/0153 contains the record number within the data file (or index if 'index-only' Fetch requested). The record key is copied into the Key Return pointed to by P2 and the word preceding the key contains a pointer to the next record in the logical collating sequence.

Control Word



NOTES

1. The current position in the collating sequence is undefined until at least one Access Call has been made using the key presented in the area pointed to by P2.

2. The current position may be re-defined at any time by a further Access Call with a given key.

3. When positioning for sequential access, it is not necessary that the given Access Key be loaded in the file. The first following Sequential Access call will fetch the record with the next higher key (or result in end-of-file).

4. GENERIC KEYS

To position the first record of a given class eg. the first record beginning BCL, the given Access Key is the class string terminated by a NUL byte eg. BCL NUL. Since valid keys do not contain NUL bytes this will result in 'Key Not Found'. It is the calling programs responsibility to verify, when the next higher record is fetched, that it actually belongs to the class eg. that there is at least one record beginning BCL.

5. To position to the beginning of the file the given Access Key is bit 17 set in each word of the key as negative keys are allowed.

FETCH - PROGRAM

JSBR IZ 1670 037670
P1 = Control Word
P2 = Address of four character ASCII name

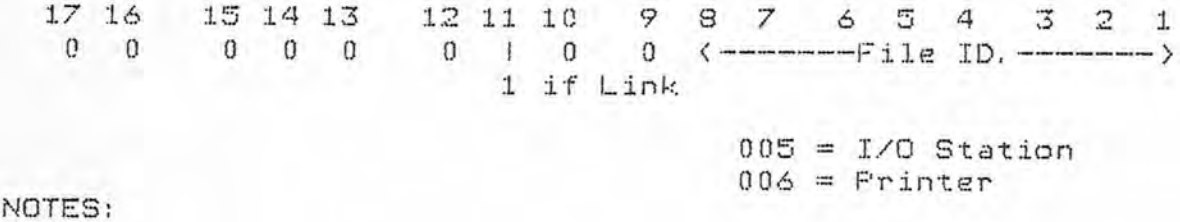
The Program identified by the File identifier (specified in P1) and name (a double word pointed to by P2) is read into memory.

If the "LINK" option is specified in P1, control is passed to the logical entry point within the overlay molule associated with the four character program name. The logical entry point is established by use of the Operating System Utility POMM (see UTILITIES section).

If the program is not found the subroutine returns to the step following P2. If the program is found but "LINK" is not requested the subroutine skips this step on return.

On completion of the transfer, 00/0153 contains the Overlay Module Number 00/0144 points to the first word of the Overlay module and 00/0154 points to the logical entry point.

Control Word



NOTES:

1. Byte Addressing of the program name is not supported. All four characters addressed by P2 comprise the name (a NUL byte does not act as terminator).

2. This subroutine is normally only used by the Operating System.

3. Subsequent 'Fetch Overlay' requests will extract the required Overlay from the Library (Master or Slaves 1 to 3) in which the Program was initially found.

FETCH - OVERLAY MODULE

JSBR IZ 1670 037670
P1 = Control word
P2 = Addres of Module Number

The overlay module identified by the File Identifier (specified in P1) and Module Number (a single word binary field pointed to by P2) is read into memory.

Control returns to the step following P2 upon completion of the transfer, unless the "LINK" option is specified in P1 in which case control is passed to the first word of the overlay module.

The absolute address of the first word of the module will be found in memory location 00/0144 upon completion of the transfer.

Control Word

```
17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
 0  0    0  0  |    0  |  0    0  <----File ID. = 002---->
               |       1 if Link
               Fetch from Utilities Library (Slave 3), if
               defined
```

NOTES

1. Automatic Offset Address Resolution does not occur; the first step of the module is assumed to contain an instruction.

2. Task HALTS at 00/1377 if the Module Number is invalid. (A register contains FETCH back address)

3. Task HALTS at 00/1373 if a software hashfail is detected. (A register contains FETCH back address)

LOAD RECORD ON INDEX SEQUENTIAL FILE

JSBR IZ 1661 037661
P1 = Control Word
P2 = Address of Record to be loaded (or zero if no data record to be loaded).
P3 = Address of key (with preceding 2 word workspace).

The routine returns to the step following P3 if a record with the specified key already exists ("record not loaded") otherwise the routine skips this step on return ("record loaded").

When the key does not already exist, the new key is inserted into the index at the appropriate point in the logical collating sequence. If a data file entry is to be loaded, the subroutine obtains an unused slot from the data file and loads the new record.

If the ISAM file has been previously defined as INDEX only, or if the 'load index entry' only option has been used, then P2 can be set to zero. This option is useful when a key is to be changed while leaving the data unchanged, or if it is required to set up multiple indexes for one data file. In the same way as a FETCH/LOCK and REWRITE, no entries to the Task Scheduler should occur between:

1. Delete index entry and load index entry in the case of a key amendment.

2. First index entry and subsequent index entries in the case of setting up a multiple index.

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
  0  |    0  0  |    0  0  0   |  <-------File ID.-------->
     |          |              Protection overide
     |          Put Data record I/D into first word of
     |          Data record when seized
     Load Index Entry only
```

17 set to zero

16 set if load index entry only is required.

15-14 reserved.

13 set if Data record requires its own slot number placed in the first word.

12-10 reserved.

9 protection overide.

8-1 file identifier.

NOTES

1. It is the calling programs responsibility to ensure that keys are space filled, if required and start on a word boundary.

2. All relevant FSBs will be automatically updated. The file handling routines will check for a file full condition on either the data file or secondary index file, causing the Task to HALT if either becomes full. By using the Restart Utility described in UTILITIES section and specifying an address of ACCEPT only a retry will be initiated. This will enable the Operator, if possible, to either perform a re-organisation or delete appropriate entries and then retry as above. Both the Secondary Index and Data file (if present) require 4 word FSBs. The Secondary index FSB MUST be Disk based.

3. On return from the LOAD, location 00/0153 contains the slot number of the new record within the direct access data file. In order to set up another index, the first word of the two word workspace area preceding the key simply needs to contain whatever was returned in 00/0153 and the key area to be set up as required. In addition 00/0154 contains the slot number of the new record within the direct access secondary index file if a new entry was created OR alternatively the slot number of the record which already exists.

4. The preceding two word workspace will contain:

   i. Pointer to data record if it exists.

   ii. Pointer to next logical entry (set up by this routine).

5. The input, or print buffer is used as workspace and is then space filled on return to the calling program.

DELETE RECORD FROM INDEX SEQUENTIAL FILE

JSBR IZ 1661 037661
P1 = Control Word
P2 = Address of Key (with preceding 2 word wordspace)

The routine returns to the step following P2 if no record with the specified key is found ("record not deleted"), otherwise the routine skips this step on return ("record deleted").

When the key is found, it is removed from the index and its associated record in the data file is released; both are made available for re-use.

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
  1  |    0  0  0    0  0  0   |  <-------File ID.-------->
     |                         Protection overide
     Delete Index Entry only
```

The delete index entry only option is useful for removing multiple index entries for one data file or for amending an index key. As with the load feature no intervening Task Scheduler calls should occur between the DELETE and subsequent LOAD of a revised key.

Note.

1. The input, or print buffer is used as workspace and is then space filled on return to the calling program.

2. All relevant FSBs will be automatically updated. Both the Secondary Index and Data file (if present) require 4 word FSBs. The Secondary index FSB MUST be Disk based.

INDEXED SEQUENTIAL FILE REORGANISATION

It is essential to re-organise the indexes to an Indexed Sequential File from time to time because:

a) New secondary index records, although logically linked into the correct position in the collating sequence, are physically loaded into the next free record and this tends to increase the number of disk transfers required to search for a key.

b) Primary index records tend to become unevenly spaced along the logical sequence.

This program requires Spool Workspace to run efficiently. The program scans the specified ISAM file description(s) and determines the extent of the largest Secondary index. This extent is the optimum amount of workspace required. If this extent is not available, the re-organisation(s) may still continue but the resulting execution time may be considerable depending on how little workspace is available.

The program can be requested in a number of ways:

Utility call sequence

This option allows "stand alone" programs to be quickly created.

```
JSBR IZ 1670
P1 = 012002              ) Fetch & Link, Utilities library
P2 = 0/0221              ) Module 021
P3 = ISAM File Identifier (Bit 17 if next parameter is
                           also a File I/D; up to 31
                           ISAM files can be presented
                           in one control block)
   |
   |
Pn = Byte Address of User Title
                          (ASCII terminated by NUL - up
                           to 47 characters)
```

NOTES:

1. The program is requested at an I/O Station; a suitable name should be placed in the I/O Station Directory.

2. The operator will be prompted for Password 3 from the Applications Passwords file.

3. The specified parameters are checked and the optimum amount of Spool workspace required is matched against the amount of Spool currently available. The user is informed if the performance of the re-organisation is likely to be impaired.

4. The request is posted to the currently assigned plain paper queue and the I/O Station returns to the "PROGRAM?" prompt.

5. If required the re-organisation can be performed immediately, within the current I/O Station partition by entering the letter "D" at the 'Process?' prompt.

6. Each of the specified files are re-organised in turn.

7. Only the indexes are re-organised, not the data file.

8. The program will set up the Primary Index for the file one record for every N1/N2 records in the Secondary Index where:

   N1 = number of 'live' record slots in Secondary Index.
   N2 = number of record slots in Primary Index.

9. The calling sequence, above, should always reside in the first page of the partition. i.e. before 1777 offset.

It is also possible to perform or request a file re-organisation from within an applications program:

Programmed call sequence

```
JSBR IZ 1670
P1 = 12002          ) Fetch & Link, Utilities library
P2 = 00/0222        ) Module 022
P3 = Pointer to parameter block
```

Parameter Block:

```
P1= Options, ISAM File Identifier
                          (Bit 17 if next parameter is also
                           a File I/D; up to 31 may be
                           specified in one control block)
  |
  |
P2= Byte address of user title
                          (ASCII terminated by a NUL - up
                           to 47 characters)
P3= User overlay number or Zero
                          (to return to after process is
                           complete)
P4= Spool Workspace base sector  ) Available only after the
P5= Spool Workspace extent       ) first call has completed
```

Options word: (These Options refer to all files specified)

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
  |  |    |  |  0    0  0  0   0  <-------File ID.------->
  |  |    |  Post the re-org. request.
  |  |    Use previously defined Spool Workspace.
  |  Overide 'abort if not enough Spool Workspace'.
  Next Parameter is an "Options, File I/D" parameter.
```

Bit	Description
17	Allows up to 31 files to be specified in sequence.
16	If the program finds that not enough spool is available to enable the re-organisation to perform efficiently then, by default, the complete request is aborted. Setting bit 16 overides this. Care should be taken with this option as the resulting execution time may be considerable.

15 Each call to this utility will require a new area of Spool workspace. If a file needs to be re-organised a number of times within the same processing sequence the Spool quickly fills up and eventually the re-organisation(s) will abort.
When the re-organisation request is specified as being Posted to some Spool queue (see bit 14 description) the Spool Workspace base sector and extent are returned in the A and B registers respectively. On subsequent calls, the setting of bit 15 signifies that the Spool workspace area to be used is defined as the two parameters following the User overlay parameter. Care should be taken when using this option as no checking is performed on the specified parameters. In particular the Spool should NOT be allowed to clear down between the individual calls.

14 Indicates that the requested re-organisation is not to be performed directly but is posted to a spool queue. This queue should be specified before the call otherwise the posting will default to queue 1. The user overlay parameter is ignored. The Program returns to the step following P3 of the original call. The A and B registers will contain the Spool workspace base sector and extent respectively. If the re-organisation was aborted the A register will be -1 (377777 octal).

Notes.

1. The 'abort overide' option should only be used in exceptional circumstances as the resulting program execution time may be considerable.

2. If the Re-organisation is performed directly, the whole partition will be used EXCEPT 2000- to 2177-. The specified user overlay is 'Fetch & Linked' to on completion of the re-organisation(s). If no overlay is specified then the program exits to the appropriate Control Program.

3. The specified parameters are checked and the optimum amount of Spool workspace required is matched against the amount of Spool currently available. The re-organisation will perform with less than optimum Spool but the resulting execution times may be considerable.

4. Each of the specified files are re-organised in turn.

5. Only the indexes are re-organised, not the data file.

6. The program will set up the Primary Index for the file one record for every N1/N2 records in the Secondary Index where:

   N1 = number of 'live' record slots in Secondary Index.
   N2 = number of record slots in Primary Index.

7. The calling sequence, above, should always reside in the first page of the partition, i.e. before 1777 offset.

OVERWRITE

JSBR IZ 1672 037672
P1 = Address of new record

The record slot just fetched (with LOCK) is overwritten in the file's transfer buffer by the new record pointed to by P1, and the buffer is written back to disk. The subroutine returns to the step following P1 upon completion of the transfer.

Notes:

1. Task HALTS at 00/1376 if "OVERWRITE" is called out-of-context (A register contains OVERWRITE back address) ie. if the preceding fetch did not specify the LOCK option, or if the LOCK was broken by an intervening entry to the Task Scheduler.

2. The preceding Fetch should specify the TEST option in order to verify that the slot to be overwritten is 'not in use'.

REWRITE

JSBR IZ 1671 037671

The record just fetched (with LOCK) is replaced in its transfer buffer, if, and only if, it was extracted by the FETCH. The buffer is written back to disk and control is returned to the step JSBR + 1.

There are no parameters; all the relevant information is defined in the preceding FETCH parameters. The subroutine will HALT at 00/1376 (called out-of-context) if the preceding FETCH did not specify the LOCK option, or if the LOCK was broken by an intervening entry to the Task Scheduler.

STOW CONTROL RECORD

JSBR IZ 1667 037667

Writes the system control record to disk and returns to the step following JSBR upon completion of the transfer.

There are no parameters.

The System Control Record is permanently memory resident at 0/0400 and is up to 3 sectors long. The record is the only record (number 1) in the Control File, File identifier 010.

N.B.
The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial Numbers are now disk based in separate reserved files.

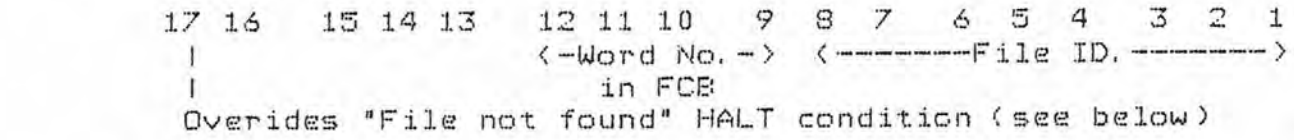
Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

For Systems which must keep the Control Record a better method of use is to Fetch & Lock the Control Record (File 010, record 1) followed by the relevant updates and a REWRITE subroutine call.

EXTRACT FROM FILE CONTROL BLOCK

JSBR IZ 1700 037700
P1 = Word Number, File Identifier

Parameter Format



The subroutine returns to the step following P1 with the A register holding the contents of the specified FCB word and with the B register pointing to the absolute memory location of the word.

If the File is not available a HALT occurs at 00/1376, A=Back Address, B= P1 of the offending call unless Bit 17 of P1 was set, and then the routine will return with 201420 in the A register, B undefined.

Examples for direct access file	A register on Return
P1 = 001420	Min. record number for file identifier 20
P1 = 002020	Max. record number for file identifier 20

SECTION 5

SPOOLING AND POSTING

CONTENTS

	PAGE
THE SPOOLING SYSTEM	5.1
Introduction	5.1
Job Definition	5.1
Spool Record Transfer	5.2
Spool File Maintenance	5.3
Print Queues	5.3
Posting to print queues	5.4
Extension Queues	5.5
Multi-stationery Programs	5.5
Document Identification	5.6
Reprinting documents	5.6
SPOOL FILE HANDLING ROUTINES	5.7
Spool	5.7
Spool with different Program name	5.9
Post to Print Queue	5.10
Spool and Post	5.11
Spool and Post with different Program name	5.12
Dequeue Posting	5.13
Unspool	5.14
Stow Spool Buffer	5.15
Stow Specified Spool Buffer	5.16
Specify I/O Station Spool Queue	5.17
Specify I/O Station Print Queue with reprint Option	5.18
Assign Spool Queue	5.19
Assign Unspool Queue	5.20
Skip if original	5.21
Get Next Spool Record Number, this Task	5.22

THE SPOOLING SYSTEM

Introduction

The function of the Spooling System is to pass jobs entered at I/O Stations for background processing by other, lower priority tasks. The I/O Station is thus able to go on to other work without having to wait for the background job to complete. In particular, jobs requiring printout have to be passed via the spooling system to a printer task; an I/O Station program cannot directly output to a printer. A less important function of the SPOOL is to provide workspace for large or complex applications such as File re-organisation, an area to accumulate a variable number of input data entries from an I/O Station such as Transactions within a Cash Posting application.

The data defining jobs awaiting background processing is held in a direct access file known as the Spool file. Records within this file are 128 word long (ie one sector) the last 4 of which are reserved for OS use: double word program name and two chain linking words. Only the job definition is normally placed into the Spool File eg. the start and end account numbers and, say, an area code for a debtors listing; it should not contain ASCII print lines since this is far too wasteful of disk space. The backgound task retrieves the definition from the Spool File and has the same access to data files (via the File Access System) as do I/O Station tasks.

Job Definition

As a general rule, the I/O Station task should create compact spool records and perform the minimum of processing beyond input validation; wherever possible, processing and updating should be carried out (at a lower priority level) by the printer task. This improves I/O Station response time, reduces the spool storage requirement and eases the problems of recovery after a cancellation request or machine fault.

For the majority of programs, one spool record is ample space to hold one job definition. A file print between limits, for example, requires only the first and last record identification. On the other hand an invoice may require an unpredictable number of spool records, because it may contain any number of lines. Yet both examples clearly constitute one 'job' to be presented to the background task.

Sometimes it is not so straightforward to define a print 'job'; if an I/O Station is entering cash postings, or stock adjustments, is each individual transaction one job, or should the whole batch be one job? The answer at one installation may not suit another, because the background task cannot process a job until it has been 'posted' and cannot interrupt a job once started.

If cash postings are posted as a batch, the batch will be printed and totalled as one report, but not until the I/O Station has completed the batch.

If posted individually each transaction may be printed immediately and transactions from more than one station may be merged into one report.

Spool Record Transfer

Because of the varying number of records to define one job, the system distinguishes between the writing of data into the Spool File and the 'posting' of jobs for background processing. All the data defining a job must first be written to the Spool File by means of one or more calls to the SPOOL routine; only then may it be presented by the POST subroutine for background processing. The SPOOL and POST subroutine combines these two functions where the job definition requires only one spool record.

All tasks are provided with a 'Spool BUffer', one sector in length, which begins at step 3400- in the Task Partition. The Spooling System (ie. SPOOL, UNSPOOL, POST, SPOOL AND POST etc.) always uses the Spool Buffer for record transfer (thus the File Access System may be used without interference in the same program). Records to be spooled must be in this buffer when the SPOOL subroutine is called. When the Spool Buffer is not required for spooling it may be used as workspace.

Any task I/O Station or printer, may write records into the System Spool by calling SPOOL and any task may subsequently retrieve that record by calling UNSPOOL provided it knows the record 'slot' number used by SPOOL. The spooling task may obtain this slot number by executing the subroutine:

```
JSBR IZ 1616
```

before calling SPOOL, and save the slot number (loaded into the A register by the above instruction) within its memory partition.

Each time the SPOOL subroutine is called, it writes a new record into the Spool File and updates the slot number for the next time round. In other words, the slot number is allocated in advance, a fact of vital importance because it enables the SPOOL subroutine to insert the next slot number into the record it is currently spooling. All records spooled by the same task are thus linked (by slot number) in the order in which they were spooled, and it is only necessary (for spooling purposes) to hold in memory the slot number of the first record in the chain, which will be called the 'Header' record.

In order to carry out this, and other, functions, the last four words of every spool record are reserved for system use. The next record slot number will be found in step 3577- of the Spool Buffer on unspooling the record.

Spool File Maintenance

Whilst the SPOOL subroutine always writes new records, the UNSPOOL routine does not release a record for re-use. Consequently, a record may be unspooled (or retrieved) any number of times. Indeed programs have no means of releasing Spool records; the spooling system automatically releases all spool records as soon as it detects that there is not information held in the spool file and all task partitions are idle. In fact, Operators are primed to ensure that such a situation occurs before the spool file becomes 100% full.

Print Queues

Because an I/O Station task writes records to the spool file this does not automatically ensure the job definitions contained in those records will be processed. The jobs must be assigned to print queues. As described in the SOFTWARE CONCEPTS section the spool file is defined into a number of print queues which are simply linked chains of jobs waiting or queuing to be processed. Once a job has been assigned to a print queue then the first record of that job ie. the Header Record, is 'posted' to a Print Queue Table. A Print Queue is simply a list of Header Record Slot numbers linked together. Each entry in the list represents one 'job'. If a task, having spooled one or more records defining a job, places (ie. posts) the header record slot number into one queue, another task, suitably programmed, may pick up the slot number and, removing it from the queue, unspool the header record and process the job.

Initially, it is sufficient only for a printer task to be instructed to service a print queue for processing to commence. The Printer Control program continually tests the print queues currently being serviced by that printer for the presence of job postings. If a job is found then it is retrieved (ie. unspooled) from the print queue and the appropriate program, as defined in 2 of the 4 reserved words at the end of the spool record, is overlaid into the printer's task partition (unless it is already in that partition) and control passed to it. It is the responsibility of the program to retrieve other spool records, if any, which completes the definition of the job. Once the job is processed, processing of the next job in the queue ie. it retrieves the next Header, and so on until either the print queue is exhausted or Operator intervention instructs the printer to service some other print queue.

For completeness and general interest, the manner in which the OS posts a Header Record to a print queue is described later in this section.

The number of print queues varies between systems; twenty queues is a common figure, but there must be one for each type of preprinted stationery (invoices, credit notes, statements, purchase orders, cheques, remittance advices etc), preferably more than one queue for plain-paper one for record deletions and usually one 'reprint' queue for each printer. Plain paper queues have the lowest queue numbers.

Zero page 00/0060 contains the system's highest plain-paper queue number. Zero Page 00/0057 contains the system's highest queue number, which is always the 'deletions' queue. The reprint queues are just below the deletions queue number.

The OS does not allow the deletions queue to be processed by any background task unless all other queues are empty. All record deletions requested by File Maintenance programs should be posted to the deletions queue, without deleting the record from file until processed by the printer task. This overcomes the possibility of a record being deleted whilst there is a queued job which references it still awaiting processing.

Plain-paper programs should usually allow the operator to select the queue to be used (by means of the 'U' command); this increases operational flexibility.

Posting to Print Queues

Because the number of jobs in a print queue is unpredictable and may become quite large, only the first and last Header record slot numbers for each print queue are held in a Print Queue Table.

Regardless of the job definition on that Header, the last 4 words of the record will contain the following information:

STEP	CONTENTS
3574-)	ASCII program name of 4 characters space
3575-)	filled if necessary
3576-	Slot number of next header record within the Spool File for this print queue.
3577-	Slot number of next spool record if any, for this job.

When a job is posted to a print queue it is always added to the end of that queue. If the queue is not empty the last Header slot (as given in the Print Queue Table) is retrieved and the new Header record slot number is inserted into step 3576-. (The task spool buffer is over-written by this process). The previous last Header is written back to the spool file and the slot number of the last Header in the print queue table is updated. This completes the posting of a Header record to a print queue.

Extension Queues

The concept of the Print Queue is simply as a means of transferring information and/or control to a Print Program.

One may also wish to accumulate data to be processed later at the end of some defined cycle. For example, during a days invoicing an audit trail may be built up, on disk, to be processed at end of day by some other program. One method to achieve this is to allocate a specific file as the audit trail file and write software to add to it, in the Invoicing Suite, and delete from it in the Audit Trail program. With this method programming and installation energies are used to write the software and use a specific file to hold this intermediate data.
A better method is to use an Extension Queue, to which the Invoicing SPOOLs and POSTs the data. The Audit Trail program DEQUEUEs this data. Here the related software becomes simpler and the specific data file no longer required.
The Extension Queue differs from the Print Queue in that the Posting itself cannot itself initiate a Print Program.

Multi-Stationery Programs

Certain I/O Station programs may require more than one type of stationery (eg. the B/N package payments program produces remittance advices, cheques, credit transfers and a plain-paper report). Each stationery type should be posted as a separate printer 'job' to its own print queue. If the order of printing does not matter, the I/O Station may post each job (reassigning the print queue number before each POST call). Each job must have its own header record; it is an error to post the same header to more than one queue (because there is only one link word at step 3576-). If the order of printing does matter, the I/O station may post the first job, which itself posts the second, and so on in cascade fashion.

Each header record will have the spooling tasks current four character program name at step 3574-.

Facilities are available to Spool to different Programs.

Document Identification

Computer produced documents (invoices, credit notes, delivery notes etc.) must be assigned a computer generated serial number.

Each document type will normally have its own number series; the number most recently assigned to a document being held in a file of such serial numbers (see MAPS within the UTILITIES Section).

The number assigned to a document must be displayed to the VDU operator for cross-referencing purposes. Thus, the serial number must be assigned within the VDU program and passed to the printer via the print spool; this should always be via step 3573- of the Spool Header Record of the document.

For audit purposes there must be no gaps in the number series. Thus, the serial number cannot be assigned until the operator has completed all input for the document (since he/she may elect to cancel it at any time during input).

Reprinting Documents

It is normal practice to reserve one print queue for each available printer for reprinting purposes; these are just below the 'deletions' queue number.

In order to set up document details for reprinting the applications program must make use of the subroutine 'Specify Print Queue with Reprint Option' ie. JSBR IZ 1413. When this is done items become available for reprinting (ie. are posted to the reprint queue for that printer) immediately printing of the original begins. Thus, if the printer breaks down during printing of the original it would be possible to obtain the reprint (provided another printer is available to the system) before the original.

It really has to be a matter of system operating efficiency and/or design constraints whether any file updating is done during printing. If a 'print and update' mode is operational then the program will also make use of the subroutine Skip if Original ie. JSBR IZ 1776 enabling the print program to detect if the document details are being printed for the first time or are being reprinted. Such a subroutine is a necessity if file updating is not to be repeatedly performed every time a reprint is requested.

It should be noted with a print and update mode of operation that in the situation where a printer breaks down and a reprint is done on another printer, that file updates may still be outstanding; indeed they would only be completed if the original printer can be repaired and restarted without having to re-bootstrap the System.

The operator identifies a document for reprinting by its serial number. This information is passed on to the utiltity RPT which searches the reprint queue for the appropriate printer and identifies the correct Header record by the Serial number which it assumes is held in step 3573- of the Task Spool Buffer.

SPOOL FILE HANDLING ROUTINES

SPOOL

JSBR IZ 1645 037645

Each task has a unique slot within the Spool File reserved for the next time it requires to spool any information. Once a task actually spools information to that slot, another (being the next slot free within the Spool File) is automatically reserved for it. The slots currently reserved for each task within the system are held in a Spool Table and initially this will be set up with task 1 having spool record 1 allocated to it, task 2 having spool record 2 allocated to it, and so on.

This subroutine will write the contents of the task spool buffer to the next slot within the Spool file reserved for that task. As previously indicated the spool buffer contains 128 words (ie. one sector) the last 4 of which are set up by the OS for the purposes of identification and retrieval.

Before writing a record to the Spool File this subroutine performs the following:

a) selects the next free sector available in the Spool File (halting if it is full).

b) inserts the next free spool file slot number into step 3577- within the task spool buffer.

c) Clears step 3576- within the task spool buffer.

d) Inserts the 4 character program name into steps 3574- to 3575- within the task spool buffer.

The contents of the task spool buffer are then written to the Spool File at the slot number indicated for this task within the Spool Table. For programmer reference, the the slot number to be written to may be accessed by a JSBR IZ 1616 which returns the slot in register A. The Spool Table is then updated whereby the next spool record slot number that was inserted into step 3577- of the task spool buffer is set up for this task in the Spool Table. In this way a record is written to the Spool File in a given slot containing a 'link' to another slot and that slot is allocated, via the Spool Table, to the same task.

Thus, records written to the Spool File by a task will be linked together and provided that the first spool record for the task is known, the records may be retrieved in the same order as they were written. The retrieving task may be a different one from the spooling task provided that a mechanism exists for 'posting' the first record slot number from one task to another.

Note that this subroutine cannot mark the end of a chain; since a new free slot is always obtained before writing the current record, the chain is effectively endless. Also a record may be spooled only once; if it is required to update a spooled record then this may be achieved by use of the UNSPOOL and STOW SPECIFIED SPOOL RECORD subroutines described later.

SPOOL WITH DIFFERENT PROGRAM NAME

JSBR IZ 1764 037764
P1 = Address of 4 character Program name

This routine performs exactly as the SPOOL subroutine except it uses the specified Program name instead of the current Program name.

Note.

Byte addressing NOT allowed.

POST TO PRINT QUEUE

JSBR IZ 1647 037647
P1 = Address of Header Record Slot Number

The Header Record Slot Number is the Spool File Slot number allocated to it.

The number of the print queue is that established by the Tasks most recent "SPECIFY PRINT QUEUE" subroutine call. The record(s) comprising the posting must already be "Spooled".

If the queue is not empty, the new posting is added to the end of the queue. (The slot number addressed by P1 is inserted into word 3576- of the Header Record of the posting currently at the end of the queue.) The subroutine uses the issuing Tasks Spool Buffer as workspace in carrying out this operation; ie. the subroutine destroys the contents of the Spool Buffer.

The printer control program will process the queue strictly in the order of posting, but if more than one printer is draining the same queue an overtaking situation can occur.

On return step 3400- of the Spool Buffer will contain the issuing task's task number. This is useful when the printer task requires to inform the originating task that it has completed printout of the details it has retrieved from the Spool File (see SPOOL AND POST following).

SPOOL AND POST

JSBR IZ 1651 037651

Combines the SPOOL and POST TO PRINT QUEUE subroutines where the item to be posted consists of a single spool record.

There are no parameters; the number of the print queue is that established by the Task's most recent SPECIFY PRINT QUEUE subroutine call. If this number is zero then neither spooling nor posting occurs.

The OS inserts the Task Number into the first word of the Task Spool Buffer when an I/O Station starts a new program, and after each SPOOL and each POST subroutine call. Thus the receiving print program may identify the station from which it has received work.

The facility does not reserve the first word of the Spool record.

This subroutine may destroy the contents of the Task Spool Buffer, but not before writing the Buffer to disk.

SPOOL AND POST WITH DIFFERENT PROGRAM NAME

JSBR IZ 1763 037763
P1 = Address of 4 character Program name

This routine performs in the same way as the SPOOL AND POST subroutine except the specified Program name is used, not the current Program name.

Note.

Byte addressing is NOT allowed.

DEQUEUE (EXTENSION) POSTING

JSER IZ 1650 037650

A previously defined UNSPOOL queue is checked for postings. The routine returns to the following step if the queue is empty, or skips this step if a posting is found and dequeued. The relevant Spool sector is placed at 3400-, and the Spool Header Record number will be found at 00/0153.

NOTES.

1. This routine is available to Printer partitions only.
2. A valid Unspool Queue must have been specified.
3. The Deletions Queue is not available with this routine.

UNSPOOL

JSER IZ 1646 037646
P1 = Address of Spool Record Slot Number.

The Spool record is read into the Task Spool Buffer and control returns to the step following P1. Word 3577- of a spool record will contain the slot number of the next record written by the spooling Task.

Word 3576- of the spool record is used by the Operating System to chain 'postings' together; it will contain the slot number or link to the next spool header record in this print queue (if any).

Words 3574- & 3575- of the spool record contain the program name (4 ASCII Characters) associated with the spooling Task and words 3400- to 3573- are defined by that task.

The Printer Control Program automatically unspools the header record of each 'posting' and uses the 4 characters program name to determine which printer overlay module is required. When this module is given control, the header record will already be in the Spool Buffer.

Unspooling a record does not release the disk space for re-use; the same record may be unspooled any number of times. Rather than maintain a free-record chain, the Operating System simply frees the entire Spool File area whenever all print queues are empty and all Tasks are idle.

STOW SPOOL BUFFER

JSBR IZ 1657 037657

This subroutine writes the issuing Task's Spool Buffer to its preallocated record slot within the System Spool File; control does not return to the Task until the transfer is complete.

There are no parameters; the record slot number is the next available for this task but it is not updated by this subroutine.

This subroutine differs from the SPOOL subroutine in that no new spool record is seized and no information is placed into the last 4 reserved words of the record. It will be found useful when the Spool Buffer in question is not completed (ie. not ready for spooling) but the present contents need to be saved so that the buffer area can be used for some other intermediate job.

STOW SPECIFIED SPOOL BUFFER

JSBR IZ 1600 037600
P1 = Address of Spool Record Slot Number

The contents of the issuing task's Spool Buffer are written to the Spool File record specified. On return the spool buffer is unchanged and the current slot number allocated to this task is not updated.

This subroutine is useful in conjunction with UNSPOOL when a previously spooled record is required to be updated, for example, in the recall and modificaiton of Invoice line entries prior to Invoice completion.

SPECIFY I/O STATION SPOOL QUEUE

JSBR IZ 1613 037613

In order that a Task be able to post input details to a Print Queue it is necessary that a Queue be allocated to the Task. This routine enables the programmer to specify the print queue into which the Operating System is to place postings made by this task.

Print queues are identified by a number from 1 upwards, each number being assigned a particular stationery type. In the interests of operational flexibility several queues (from 1 to the number contained in memory location 00/0060) are assigned to plain paper. The highest queue number (contained in memory location 00/0057) is reserved for record deletions by file maintenance programs, and the highest queue number - 1 is reserved as a reprint queue. Queue number 0 provides a means of suppressing postings.

On entry to this routine, the A register must contain one of the following:

a) Print queue number to be used.

b) A negative number if any plain paper queue may be used.

c) Zero if any plain paper queue may be used or if posting may be suppressed.

d) Bit 16 plus the Extension Queue Number.

(Where a choice is permitted the routine will assign the queue number by reference to the OPERATING SYSTEM COMMAND SECTION).

The routine will output a message to the Task's I/O Station stating the queue number assigned. This number will be used in all subsequent postings by this Task until a return is made to Control. When a particular queue is assigned within the program itself by the use of this subroutine then this overrides any queue specification set up by the 'Use ' Command.

SPECIFY I/O STATION PRINT QUEUE WITH REPRINT OPTION

JSBR IZ 1413 037413

This routine enables the programmer to specify the print queue into which the Operating System is to place postings made by this Task as before. In addition, it instructs the System to make the postings available to the 'RPT' utility for possible reprinting. In fact, a flag will be set on the Header record so that when printing takes place, the Printer Control Program requeues the posting into its reprint queue.

Print queue numbers and the contents of the A register on entry to the routine are exactly as described previously except Extension Queues are not allowed.

ASSIGN SPOOL QUEUE

JSBR IZ 1614 037614

An I/O station Task will normally use the 'Specify I/O Station print queue' subroutine since this displays the assigned queue number on the screen. However, if the I/O Station Task is posting to more than one print queue then the 'assign print queue' subroutine should be called prior to each 'post' call and this will suppress the display of the print queue assigned.

The assignment remains in force until a new assignment is made, or until the program returns to the Control Program.

Obviously, where the printer program requires to assign a print queue, then it will use this subroutine thereby suppressing the display of the print queue assigned.

On entry of this subroutine the A register must contain the print queue number to be used in bits 7-1, with bit 17 set if the Printer Control Program is to re-queue the posting into its reprint queue or bit 16 if requiring an Extension Queue.

ASSIGN UNSPOOL QUEUE

JSBR IZ 1602 037602

When using the DEQUEUE subroutine, JSBR IZ 1650, the queue from which postings are to be taken must be specified to the OS.

On entry to this subroutine the A register must contain the Queue Number to be used in bits 7-1, with bit 16 set if this is an Extension Queue.

The A register will contain the absolute queue number that was previously specified on return from this subroutine. This is to allow the calling program to reinstate the original queue number before terminating.

This routine is only available to Printer partitions.

SKIP IF ORIGINAL

JSBR IZ 1776 037776

This routine enables a Print Program to determine whether it is processing original print details as opposed to reprinting them.

The subroutine returns to the step JSBR + 1 if the details are being reprinted, otherwise it returns to step JSBR + 2.

The contents of the A register are preserved on return from this subroutine but the B register is undefined.

GET NEXT SPOOL RECORD NUMBER

JSBR IZ 1616 037616

Returns with the next Spool record number in the A register.
The B register is preserved.

SECTION 6

I/O STATION & PRINTER PROGRAMMING

CONTENTS

	PAGE
PROGRAM ACCESS	6.1
PROGRAM DESIGN	6.2
Program Format	6.2
Program Efficiency	6.4
Program Neatness	6.5
I/O STATION PROGRAM CHECK LIST	6.6
I/O STATION LIBRARY SUBROUTINES	6.8
Error Handler	6.8
Flash	6.9
Flash Single Station	6.10
Get	6.11
Get - Character Mode	6.12
- Binary Mode	6.14
- Fetch Mode	6.16
Split	6.18
Get Password	6.19
Halt	6.21
Inhibit	6.22
Load Overrun	6.23
Put Characters	6.24
Put Spaces	6.25
Return To Control Program	6.26
Specify Default Restart Address	6.27
Specify I/O Station Escape Point	6.28
Suspend	6.29
Suspend Display	6.30
Suspend for number of cycles	6.31
PRINTER LIBRARY SUBROUTINES	6.32
Flash	6.32
Flash Single Station	6.33
Halt	6.34
Print Line	6.35
Return To Control Program	6.37
Specify Default Restart Address	6.38
Specify Print Termination Options	6.39
Specify Re-entry Point	6.40
Specify Termination Routine	6.41
Suspend	6.42
Suspend for number of cycles	6.43

PROGRAM ACCESS

Each Station has a permanently resident Control Program which enables it to call any program in the Input Station Program Directory independantly of all other stations. When the machine is started (from the control panel switches) each control program sends the "PROGRAM?" prompt to its own display screen. Upon receipt of a valid program name (up to four characters long), the Control Program loads the appropriate Overlay Module into its memory partition, resolves to absolute any offset addresses at the beginning of the module, and passes control to the designated Entry Point within the module .

As far as printer and background tasks are concerned, the Control Program will be instructed to service a particular spool queue. For printers, once it finds information in that queue ie. at least one Header Record, then it will automatically fetch that record into the Spool Buffer of its own task partition area. Within words 3574- to 3575- it will find a program name and by use of this name the Control Program loads the appropriate overlay module into its memory partition, if it is not already there, in the same way as an I/O Station Control Program.

All applications programs are held on disk as overlay modules each of which may occupy 1 to 13 (decimal) sectors and are contained within an Overlay Module Library. An installation may have various Applications, which should be segregated into different libraries. This includes the Utilities library, which MUST always be assigned as Slave library 3 on all available Application Systems.

A useful feature is that up to 3 Application Libraries can be made available to an individual System. This allows, for example, the Payroll to be accessible to the user alongside the main bespoke Application whilst maintaining it as a stand alone package for the ease of the Programmer/Installer.

It should be noted that for the purposes of program maintenance, each library MUST be declared as the Master library on a System.

PROGRAM DESIGN

Program Format

When writing a program it is strongly recommended that a few simple guidelines are adhered to. If they are, then considerable advantage may be obtained later in terms of

a) subsequent modifications
b) familiarisation by other personnel
c) re-familiarisation by oneself
d) program tidiness.

As the programmer gains expertise and experience he/she will find that they automatically develop 'programming rules' to follow. Initially however, consideration should be given to the following:

1 Program Offset Address

Program instructions may refer directly only to the 'current' page or to the 'zero' page. The Program however, may occupy up to 2K or 2 pages and much of the information on the second page, eg. contents of spool buffer or Master Buffer, may require to be referenced by instructions on the first page. As setting up absolute indirect addresses invalidates the multi-programming function, it is advisable to use Offset addressing via subroutines such as "Load A reg from offset: JSBR IZ 1725" which allow instructions on the first page to reference data on the second (and also the reverse).

All OS subroutine parameters relating to a memory address should ALWAYS be expressed as an Offset address as otherwise problems will be experienced when running programs in memory above 32K.

Reference to data on the first page from instructions on the second page or in subsequent overlays has to be accomplished. A common pitfall for many programmers (not necessarily beginners) is to try and reference data on page 00- when the program has just transferred to the next page either by a JUMP or JSBR instruction, or by the natural sequential flow of instructions within that program eg. LDA 0005 executed from 01/0001- will access the contents of 01/0005- and NOT 00/0005-.

2. Spacing

As a general rule always try to leave gaps in the coding bearing in mind that the gaps one leaves initially when writing the program may be completely or almost completely used up by the time the program is fully debugged. A fully debugged program should be liberally sprinkled with gaps to facilitate possible future amendments without causing a degeneration into a morass of program patches.

3. Literals

The format of literals within a program can be made to vary according to the amount of program space available. If there is ample space within the program then it is better to have literals written out fully, including all spaces. Changing of report headings is often less complicated when this approach is used.

When space is becoming, or is likely to become, a luxury within a program then it is recommended that literals have large areas of spaces excluded. It is the responsibility of the program then to copy a number of 'literal components' to a space filled display buffer or print buffer as appropriate. This could result in considerable savings in terms of space required by literals but is offset by the necessity of including within the program a number of JSBR IZ 1707 (duplicate words) and JSBR IZ 1731 (space fill words) calls.

Also certain repetitive character strings are best generated (ie propagated from a small character string) by use of JSBR IZ 1707 or JSBR IZ 1741 (Move and Pad) Subroutine calls, particularly if they need to be very long.

Some literals, particularly in Print programs may be entered as Metacode giving a space saving of a third. Then, instead of using "Move and Pad" to move the literal to the Print buffer, the "Convert Metacode to Ascii" subroutine is used as a direct replacement.

4. Grouping

Familiarisation and subsequent debugging can often be greatly assisted if within any program module items such as parameter blocks, workareas and literals are grouped together as far as possible by type. If such items are widely dispersed throughout the programs this will invariable result in lots of program JUMP instructions to segments of program, and, where subsequent modifications have been made, a tendency to patch more heavily than otherwise.

5. Overlays

The 13 sector overlay area within a task partition will be sufficient for most programs. Indeed it will often be found convenient and practical to place related programs into the same module so that they may share common subroutines eg. customer load, amend, enquiry may all share the same display routines.

If the overlay area is likely to be insufficient then the program must be segmented into two or more natural overlays eg. process header, process line, cancel document, process totals. The recommended technique is to retain small control segment (say 4 columns long) at the beginning of the partition and use the FETCH OVERLAY MODULE subroutine to load and link to overlays within the remainder of the overlay area as and when required.

It is useful when using overlays if the module numbers used can be kept continuous; this facilitiates more logical filing and easier retrieval subsequently.

6. Record Buffer Areas

When record buffer areas are allocated within a program take the trouble to document each field; program familiarisation/debugging/amendment can be seriously hampered if a considerable amount of processing is being performed on a 'blank' file area.

Program Efficiency

It is impossible to discuss all ways to making a program more efficient; much will depend on the application itself. However, it will be useful if at this point the beginner is shown certain basic ways of increasing program efficiency (in terms of execution time) in the hope that this will act as a 'catalyst' during subsequent program development.

a) Never use JSBR IZ 1711 or JSBR IZ 1712 on one word locations - the number of instructions executed by the subroutine call is execessive when compared with LDA, ADA, STA.

b) Avoid using JSBR IZ 1724 (zero test) for double or triple word blocks - far fewer instructions are executed by

```
LDA  Address
IORA Address
IORA Address
AN=0
```

c) Think carfully when a requirement exists to multiply a single word number by another; the multiply sub-routine is extremely slow in terms of execution time and a combination of LSA, ADA instructions may achieve the result quickly and effectively.

d) Avoid converting one digit to ASCII using JSBR IZ 1765- it is much more efficient to insert the padding necessary using ADA eg.

```
LDA    0001 ..... A reg now contains 000003 say
ADA  Z 0260 ..... A reg now contains 000063
ADA  Z 0342 ..... A reg now contains 020063
```

in ASCII 020063 represents (SPACE) 3.

Program Neatness

Always remember that the coding sheets written by the programmer are the source document and cannot easily be regenerated if ripped, dirtied or doused in coffee. So look after them and take heed of the following:

Write as neatly as possible

b) Document the coding sensibly eg. do not use obvious comments like 'add' against JSBR IZ 1711 subroutine call - explain what is being accumulated and why.

c) It is not feasible sometimes to document individual lines, however, grouped together they accomplish some effect - explain the effect with the aid of formulae if appropriate.

d) Where gaps are left in the program do not write NOPS in these they only have to be either rubbed or snow-paked out when required for use.

e) When amending a program do not always look for the easy way out by a quick patch - sometimes the rewrite of a column or two can save considerable space and drastically improve the logical flow of the program.

I/O STATION PROGRAM CHECK-LIST

1. If there is an associated Print Program, the "SPECIFY I/O STATION PRINT QUEUE" subroutine should be called. This displays on the screen the print queue number into which the print jobs will be posted and advises the system that the Print Spool is in use.

2. The program title should be "PUT" onto the display screen.

3. If the program is to be protected against unauthorised or unintentional use, the "GET PASSWORD" or alternatively the "INHIBIT" subroutine should be called. The OS provides four System passwords (up to 9 characters long) and as many application passwords as required (up to 15 characters long) all amendable by the utility program "MAPS", addressable from any partition as follows:

   System passwords:

Location	Conventional use
00/1300 Password 1	(sales system and disk security)
00/1305 Password 2	(bought and nominal Ledger system)
00/1312 Password 3	(Not reserved)
00/1317 Password 4	(payroll system)

   Application passwords:

   These passwords are disk based and are identified by a number 1 to n. See subroutine "Get Password" later in this section.

   Monthend/Yearend routines must always be password protected.

4. Unless otherwise programmed, pressing the Escape Key has no effect other than to pass control to the Error Handler. However, this key should always be programmed to provide the operator with the means to abandon the job in hand. Thus, escape from the beginning of a logical block should lead back to the next higher logical block (usually programmed by presenting a PO in the 'GET' parameter block) whilst escape from within the block should lead back to the beginning of the block (usually programmed by the 'SPECIFY I/O STATION ESCAPE POINT' subroutine). Avoid the temptation to program the Escape Key to complete an operation; no further file updates should take place after pressing the key and indeed it may be necessary to reverse earlier updates and reset program variables and workspace.

5. If there is an associated printer program it should carry out all file updates unless there is an overriding system requirement for immediate update (eg. real-time stock updates to prevent two operators ordering the stock item). This rule simplifies programming of the Escape Key and increases the probability of avoiding a "return to security" in the event of a machine fault.

   File maintenance routines. Since a print out is not normally provided of new records loaded, the update operation to load the new record must be carried out by the I/O Station Program. But a printout should always be provided of records deleted, thus the I/O Station program merely requests that the record be deleted by the printer program.

6. Check "Digits" (an option of the "GET" subroutine). The check digit is required as input when the operator is identifying a record to be updated , but should never be required on an enquiry.

7. Null input feature (an option of the "GET subroutine) allows the program to establish a "default" reply to a prompt.

8. Alpha feature (an option of the "GET" subroutine). Enables input of non-standard replies that would otherwise be intercepted by the Error Handler. Example; if prompted for the low limit for a file print between limits, the operator may reply "A" for ALL instead of having to remember the file limits. The program may itself call the Error Handler if the reply is invalid.

   The program completes by returning to the Task Control Program:

```
JUMP Z 1402   (output "PROGRAM?" prompt)
JUMP Z 1400   (clears screen and outputs "PROGRAM?")
```

   These returns may also be specified as an Escape Point.

```
00/1402       (output "PROGRAM?" prompt)
00/1400       (Clears screen and outputs "PROGRAM?")
```

INPUT/OUTPUT STATION LIBRARY SUBROUTINES

N.B.
All subroutine parameters expressing memory addresses should, as a rule, use ONLY Offset Addresses.

ERROR HANDLER

JUMP IZ 1641	027641
JSBR IZ 1641	037641

The message 'ERROR' and/or BEL code is sent to the issuing task's display screen. The BEL code only is sent where the 'special ETX' option has been invoked within the GET subroutine (See GET Parameter options later in this section). When the error is acknowledged by the operator (by pressing the ETX key) the most recent GET or split subroutine call will be restarted automatically.

If the 'special ETX' option is active, then instead of redisplaying the prompt, any operator input from the rejected GET/SPLIT is removed from the screen, but leaving the initial prompt intact . This technique allows the operator greater control over the screen format. Obviously, certain input would disallow the use of this option eg. use of C/R; this is the programmer's responsibility.

The coding and the parameter block for the most recent GET/SPLIT call must still be in memory on entering the Error Handler ie. they must not have been overlaid.

The text "ERROR" above is in fact variable and can be chosen by the installer/customer. See Utility "COMA".

FLASH

JSBR IZ 1653 037653
P1 = Byte Address of Literal

Requests that the ASCII character string be transmitted to all I/O Stations. The literal is ended with a NUL byte.

In order that the transmission should not unduly confuse operators the string should start with the characters 'SO BEL' and end with 'SI NUL' (thus displaying the intervening text in inverse video if this is provided by the receiving device). The text should include reasonable identification of where the message has come from (eg. program name and Task Number).

Control returns to the issuing Task immediately the request has been placed, thereby ensuring its continuation even if the operating system is unable to honour the request. There is no guarantee that the string will arrive at any given station; if it is off-line no re-try attempt is made; if an output is currently in progress the OS will wait until it is completed; an input is interrupted immediately.

Since Flash requests are not queued, and since each receiving station handles the request at its own pace, it is possible to lose a message if it is superseded by another flash request. In particular, if a task issues two Flash Requests the first will certainly be lost unless a 'SUSPEND' subroutine call intervenes to allow the Scheduler to honour the first request.

FLASH SINGLE STATION

JSBR IZ 1654 037654
P1 = Byte Address of Literal

Requests that the ASCII character string be transmitted to the Task whose number is contained in the 'A' register on entry (this must be a valid I/O Station task Number). The Literal is terminated by a NUL.

Implementation is identical to that of the general FLASH subroutine.

GET

JSBR IZ 1640 037640
P1 = Address of Control Word in Parameter Block.

Outputs a prompt message to the Task's I/O Station and awaits a reply. The reply will be validated and processed according to the instructions given in the Control Word. There are three principle processing modes - Character, Binary and Fetch - separately described below.

In all cases control returns to the step following P1 upon satisfactory completion of input and processing. The original characters input from the station will be found in ASCII in the Task's Input Buffer (terminated by NUL) and the number of characters input (excluding terminator) in Absolute Memory Location 00/0045. The ESCAPE key aborts the call, control passing to the Escape Point nominated by P0 or, in default, by the 'Specify Escape Point' subroutine.

GET - CHARACTER MODE

Parameter Block

P0	Address of ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0.
P1	CONTROL WORD	
P2	(Byte) Address of Prompt Message	Variable length ASCII/MCODE character string terminated by NUL.
P3	(Byte) Address of TARGET AREA	Optional; required only when 'Convert to Metacode' or 'Move and Pad' requested.

Control Word

```
17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
 1  0    |  |  |    |  |  0   |  0  <No. of chars. input>
         |  |  |    |  |      1 if escape reqd.
         |  |  |    |  1 if special ETX reqd.
         |  |  |    1 if lower case retained
         |  |  1 if no flashback
         |  1 if move and pad      )  If both set then
         1 if convert to metacode  )  input is padded
                                      before conversion to
                                      METACODE
```

15 Set if Input is to be converted to Metacode. The Metacode is stored at the address specified by P3. Byte addressing is NOT allowed in this case.

14 Set if input is to be moved to the byte address specified by P3 and if short, space filled to the maximum length specified in bits 7-1. Only the target area is space filled, not the input buffer.

If both Bits 15 and 14 are set, the input is space filled to the maximum length (Bit 14) then converted to Metacode (Bit 15).

13 Set if input is not to be flashed back (echoed) to the Task's output display. (Used to preserve password security).

12 Set if any lower case alphabetic input is not to be converted to upper case when stored in the input buffer. (Not allowed with Metacode options)

11 If set then where the Operator's input produces an error, NO error text is displayed and the Bell only is sounded; upon receipt of an ETX the Operator's input so far is removed leaving the cursor in its original start position. This allows complex screen layouts to remain intact. Note that the use of the CARRIAGE RETURN key will destroy the effect.

9 Set if the ESCAPE address specified in P0 is to override that most recently specified by the SPECIFY ESCAPE POINT subroutine.

7-1 The terminator (accept key) is not counted when determining the length.

N.B. To prohibit Escape P0 = 001405

GET - BINARY MODE

On receipt of a reply to the prompt, the issuing task's Input Buffer is presented as a Variable Length source string to the ASCII to Binary Conversion routine. If the resulting value meets the requirements specified in the parameter block, it is placed into the binary target.

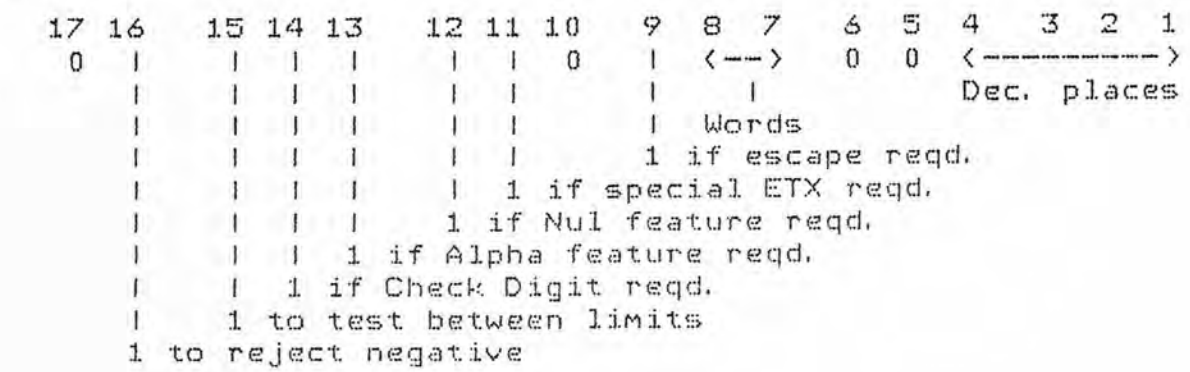
The input string is considered to be numeric if the first non-numeric character is NUL or the correct check digit. A special feature (ALPHA feature) optionally enables control to return to the calling program (rather than to pass to the Error Handler) when non-numeric input is detected; the program may then perform its own validation on the input character string.

A further special feature (NUL feature) optionally enables a preset target to remain unchanged if the reply is ACCEPT only.

Parameter Block

P0	Address of ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	(Byte) Address of Prompt	Variable Length ASCII/MCODE character string terminated by NUL
P3	Address for Binary	
P4	Address of MINIMUM VALUE	) Required only when testing) between limits (for positive) or negative numbers).
P5	Address of MAXIMUM VALUE	)

Control Word



16 Set if Negative Input to be rejected

15 Set if Input is to be tested between limits. P4 and P5 must point to the minimum and maximum values respectively. These values must have the Word Length specified in Control Word Bits 7-8; P5 must not be less than P4.

14 Set if the Input must include the correct check digit.

13 ALPHA FEATURE. Set if input which starts with a non numeric character is not to result in ERROR. The calling program may detect this occurance by testing the A register on return; if zero then the input was numeric (with correct check digit if applicable), otherwise the A register contains the first input character in ASCII in the bottom byte (the top byte is NUL) and the Binary Target will be unchanged. If there was no input (other than ACCEPT) this is interpreted as non-numeric and the A register sign bit 17 is set, unless the NUL feature is also specified in which case the A register will be zero. If the input was purely numeric when a check digit was required then this is interpreted as non-numeric and the A register bit 16 is set.

12 NUL FEATURE. Set if input consisting of ACCEPT only is to leave the Binary Target unchanged. The Target will, however, still be subjected to any tests specified by Control Word bits 15 and 16. If bit 12 is not set, input of ACCEPT only will be interpreted as input of the value zero unless the Alpha feature is active (see above).

11 If set then where the Operator's input produces an error, no error text is displayed and the Bell only will be sounded; upon receipt of an ETX the Operator's input so far is removed leaving the cursor in its original start position. This allows complex screen layouts to remain intact. Note that the use of the CARRIAGE RETURN key will destroy this effect.

8-7 Word length; if zero a default of 1 is assumed.

4-1 Free Format is allowed on input; any necessary adjustment to the precision is carried out automatically (excess digits are truncated).

N.B. To prohibit escape P0 = 001405

GET = FETCH MODE

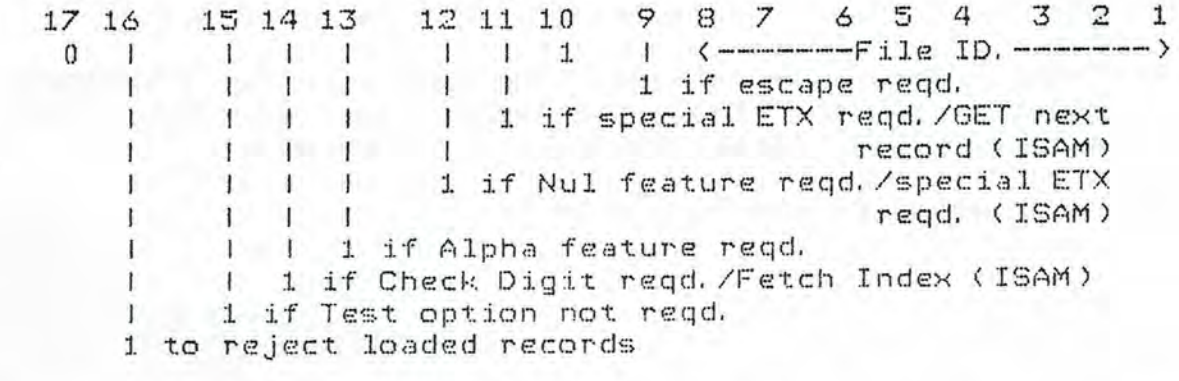
On receipt of a reply to the prompt, the characters in the issuing task's Input Buffer are converted, if valid, into a record "key" appropriate to the file identified in P1. A valid key is moved into the key area addressed by P3. Invalid input either causes control to pass to the Error Handler or to return immediately to the calling program (as determined by the ALPHA feature).

The record identified by a valid key will be fetched into its transfer buffer and tested for a load/not loaded condition (unless the test is inhibited by P1 bit 15). Control returns to the calling program if the record is loaded (unless action is reversed by P1 bit 16) and after it has been moved to the extract area addressed by P4.

Parameter Block

P0	Address at ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	(Byte) Address of Prompt Message	Variable Length ASCII/MCODE string terminated by NUL
P3	Address of KEY AREA	Direct access; single word binary field. ISAM; ASCII key string preceded by single word work-space. Extended DA; single or double word binary field.
P4	Address of RECORD EXTRACT AREA	Set zero if extraction is not required.

Control Word



16 Normally set 0 to reject records not loaded. This bit allows file loading programs to handle the converse situation; it is ignored if bit 15 is set.

15 Normally set 0 to indicate that direct-access records without the record number in the first word are to be regarded as "not loaded". This bit allows files that do not follow this convention to be handled by "GET".

14 For Direct access files, set if input must be suffixed by the correct check digit.
For ISAM files, allow the Secondary Index record to be FETCHED instead of Data record.

13 Operates for Direct Access files as described under "GET" Binary Mode.
For Indexed Sequential files, an input of a single character will be taken as "Alpha" input.

12 For Direct Access files, operates as described under "GET" Binary Mode.
For ISAM files, if input is restarted by using the ETX key then the previous input is suppressed and the prompt is NOT re-displayed.

11 For Direct Access files, if input is restarted (by using ETX key) previous input is suppressed, prompt is NOT re-displayed.
For ISAM files input of ACCEPT only will "Fetch Next" from the file. If no Indexed Sequential Fetch preceeds this option, the single word work-space allocated with the key area MUST be set to zero.

N.B. When using the special ETX option AND either Reject Loaded/Not Loaded, if the reject condition is met, then the appropriate message (NOT FOUND or LOADED) will still be displayed. It is, therefore recommended that this combination of options is not used.

To prohibit escape P0 = 001405

SPLIT

JSBR IZ 1640 037640
P1 = Address of Control Word in Parameter Block

Outputs a prompt message to the Task's I/O Station and awaits a YES/NO reply. Optionally allows for additional character input when the reply is YES.

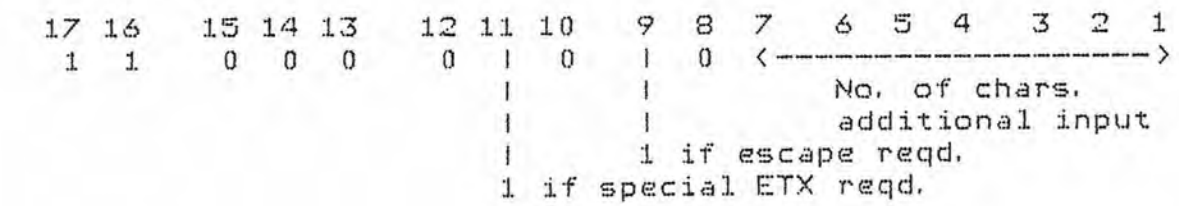
The subroutine returns to the step following P1 if the reply is NO: the subroutine skips this step on return if the reply is YES. In the latter case any additional input will be found (terminated by NUL) in ASCII in the Task's Input Buffer and the number of characters input (excluding terminator) in Absolute Memory Location 00/0045. Also the A register will be zero if the reply was YES or -1 (all bits set) if the reply was NO.

A YES reply is indicated by pressing the stations ACCEPT key (after entering any additional data), a NO reply by pressing the CANCEL/ETX key. The ESCAPE key aborts the call, control passing to the Escape Point nominated by P0 or, in default by the "Specify Escape Point" subroutine.

Parameter BLock

P0	Address of ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	(Byte) Address of Prompt Message	Variable Length ASCII/MCODE character string terminated by NUL.

Control Word



GET PASSWORD

JSBR IZ 1635 037635
P1 = Address of Control word in Parameter Block,
or byte address of actual password

Outputs the prompt "Password?" to the issuing Task's I/O Station, and compares the input with the specified Password; either a disk based application password or a memory based password.

If an inequality is detected, control is passed to the ERROR handler (the prompt will be re-output). If the input matches the password, control returns to the step following P1.

Parameter Block

P0	Address of Escape Point	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	Address of Password key area	Contains the required Password number

Control Word

```
17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
 0  |                          |  |
    Indicates disk based       |  Indicates disk based
    password; otherwise        |  password; otherwise
    assumes this byte is the   |  assumes this byte is the
    first byte of a memory     |  first byte of a memory
    based Password             |  based Password
                               Optional Escape Point
                               present (disk based
                               Passwords only)
```

The end of the password is indicated by a NUL byte.

Flashback is suppressed on input (the password does not appear on the I/O display).

Four standard System passwords are addressable from all tasks:

Location	Password	Normal Usage
00/1300	1	Sales/orders/stock system
00/1305	2	Purchase & Nominal system
00/1312	3	Free
00/1317	4	Payroll

For disk based passwords, the maximum password length is 15 characters; for memory based, 9 characters. If the ESCape key is pressed during the input of a memory based password, control is passed to the I/O Station Control program and "PROGRAM?" will be displayed.

For disk based passwords the above has the same general form as a standard GET subroutine call.

See Utility "MAPS" for description of Password(s) maintenance.

N.B.
For ease of use and general flexibility it is strongly recommended that the use of ALL memory based passwords is discontinued.

HALT

JSBR IZ 1777 037777

Halts execution of the issuing task. A message is flashed to all display screens in the format:

TASK NN AAAA, System 99 HALTED AT PP/CCSS A=XX/XXXX B=777777

where NN	=	Octal Task Number
AAAA	=	Program Name (in ASCII)
99	=	Application system number currently active
PP/CCSS	=	Address of JSBR + 1
XX/XXXX	=	Contents of A register on entry
777777	=	Contents of B register on entry

If the File Access system was 'locked' by the issuing task on entry to HALT then it will be unlocked. The task may be retarted at any chosen point by the Utility program 'R' described in Section 7 - OPERATING SYSTEM UTILITIES.

The Utility 'SE' will also display the above information until the task is restarted.

INHIBIT

JSBR IZ 1636 037636

This subroutine provides security protection to I/O Station programs without the need for authorised operators to be given a password.

On entry, the A register must contain a mask identifying the program "class" (e.g. Bit 2 = Customer File Maintenance). Control returns to the step following JSBR if the class is permitted to run, otherwise the station will BEL, display "INHIBITED" and return to the "PROGRAM" prompt.

Class Inhibit Flags may be switched by the utility 'MAPS' described in Section 7 - OPERATING SYSTEM UTILTIES. Flag bit 1 is reserved for the system utilities.

LOAD OVERRUN

JSBR IZ 1742 037742

The operating system is able to detect the keying in of characters at an I/O Station whilst no input is being accepted by the I/O Station program, ie. whilst no GET subroutine is active.

Application programs may use this subroutine to detect characters keyed outside the GET subroutine.

On return from this subroutine the A register will contain (in bits 8-1) the character most recently keyed at the input station since the last GET or LOAD OVERRUN subroutine call; if no input detected then the A register will be zero.

PUT CHARACTERS

JSBR IZ 1652 037652
P1 = (Byte) Address of ASCII Character String,
or = Offset Address, plus Bit 15, of Metacode String.

Transmits the Character string to the issuing task's display screen. The string is terminated with a NUL Byte.

This routine is intended for I/O Stations when no reply is required; it cannot be used to output to a printer. Since no reply is required, the issuing task may continue immediately output is complete.

If P1 is an offset address then setting bit 15 denotes that the source output string is Metacode, giving automatic conversion prior to output. This facility is useful where program space within an overlay is at a premium but incurs the penalty of being slightly slower in operation. In this mode, byte addressing is NOT allowed.

PUT SPACES

037656

JSBR IZ 1656
P1 = Number of Spaces.

Displays the required number of spaces from the current cursor position.

RETURN TO CONTROL PROGRAM

JUMP Z 1400	023400
JUMP Z 1402	023402

For an I/O Station either of the above may be used; the PROGRAM? prompt is sent to the issuing task's display screen, after clearing the screen in the former case, and without clearing the screen in the latter.

For a printer task only the latter is applicable and completes processing of the current print queue posting.

SPECIFY DEFAULT RESTART ADDRESS

JSBR IZ 1643 037643
P1 = Default Restart Address

Enables the programmer to specify the address of the instruction at which the issuing task is to restart in response to a request from utility program 'R' (Restart task).

The specification remains in force until superseded by a new call to this subroutine specifying a different address, or until the program returns to the Task Control Program (by "JUMP Z 1402").

If the issuing task is an I/O Station, P1 may be set zero to indicate that the address to be used is the escape point most recently specified by the "SPECIFY I/O STATION ESCAPE POINT" Routine.

NOTES

1. On program entry from the Task Control Program, the default restart address is 00/1402

2. A task may be restarted by Program "R" if:

   a) it is halted by a "HALT" subroutine call.
   b) it is waiting for a disk not on-line.

The operator may restart the task at any address, but if the need to restart is anticipated this routine is available to reduce the chances of an incorrect restart.

Example: If a program halts whilst an update flag is set, the restart should clear the flag.

SPECIFY I/O STATION ESCAPE POINT

JSBR IZ 1634 037634
P1 = Address of Escape Point

Enables the programmer to specify the address of the instruction to which control is to pass if the ESCAPE key is pressed at the issuing task's I/O Station. This event will be automatically detected by the Operating System.

The specification remains in force until superseded by a new call to this subroutine specifying a different address, but may be overriden on individual calls to the "GET", "SPLIT", or "SUSPEND DISPLAY" subroutines.

To prohibit Escape attempts, set P0 = 00/1405. (This escape point calls the Error Handler, and is automatically established when the task starts a new program).

The following standard should be adopted whenever practible:

1. Escape from within an operation (eg. creation of a customer record) should cancel that operation (program reverts to the start of the operation).

2. Escape from the beginning of an operation should lead back to the next higher logical level.

On return from this subroutine the A register will contain the address of the previous Escape point.

SUSPEND

JSBR IZ 1625 037625

Temporarily relinquishes control of the CPU to the OS Task Scheduler, which may allocate processing time to other tasks.

Control is returned to the step following "JSBR". Once a task gains control of the CPU it retains control until it relinquishes it to the Task Scheduler. This occurs automatically upon issuing any Input/Output request (Fetch, Spool, Get etc), but instances of heavy computation between I/O requests can arise, and these should be 'segmented' by "SUSPEND" calls.

In practical terms, printer programs should be literally sprinkled with SUSPEND calls on account of their low priority and buffering facilities (eg. after every "COMPUTE" call, after every three or four Binary to ASCII Conversions) whereas I/O Station programs are normally adequately segmented by I/O requests alone (it being the policy to place the computational work load onto printer programs).

It will be apparent that it is the programmers responsibility to ensure CPU time segmentation; a program loop will "hang" the entire system if it does not include an I/O or SUSPEND call.

Note that entry of the Task Scheduler automatically unlocks "LOCKED" direct access records, and may result in the contents of shared memory locations (such as 00/0045) being overwritten by other tasks.

SUSPEND DISPLAY

JSBR IZ 1601 037601
P1 = Address of Escape Point

This subroutine, when used in a display loop, allows the operator to stop/start a display by pressing the 'space' bar, giving the operator the ability to "hold" a display at will.

If the 'space' bar is pressed during a display the task is effectively suspended until the 'space' bar is again pressed. Alternatively, if 'ESCAPE' is pressed the program will resume at the specified address at (P1). If (P1) is zero the program will resume at the address specified at the last use of the subroutine 'Specify I/O Station Escape Point' (JSBR IZ 1634).

SUSPEND FOR NUMBER OF CYCLES

JSER IZ 1722 037722

The A-register, bottom byte, contains the number of cycles the calling program will suspend for. See SUSPEND subroutine.

In effect, the issuing task remains in the Suspend Queue through n Suspend calls, where n is the value of the A - register. The difference is that the calling program is not re-entered until all cycles are complete thus saving programming a count and also less CPU time is used in processing the call.

PRINTER LIBRARY SUBROUTINES

FLASH

JSBR IZ 1653 037653
P1 = Byte Address of Literal

Requests that the ASCII character string be transmitted to all I/O Stations. The literal is ended with a NUL byte.

In order that the transmission should not unduly confuse operators the string should start with the characters 'SO BEL' and end with 'SI NUL' (thus displaying the intervening text in inverse video if this is provided by the receiving device). The text should include reasonable identification of where the message has come from (eg. program name and Task Number).

Control returns to the issuing Task immediately the request has been placed, thereby ensuring its continuation even if the operating system is unable to honour the request. There is no guarantee that the string will arrive at any given station; if it is off-line no re-try attempt is made; if an output is currently in progress the OS will wait until it is completed; an input is interrupted immediately.

Since Flash requests are not queued, and since each receiving station handles the request at its own pace, it is possible to lose a message if it is superseded by another flash request. In particular, if a task issues two Flash Requests the first will certainly be lost unless a 'SUSPEND' subroutine call intervenes to allow the Scheduler to honour the first request.

FLASH SINGLE STATION

JSBR IZ 1654 037654
P1 = Byte Address of Literal

Requests that the ASCII character string be transmitted to the Task whose number is contained in the 'A' register on entry (this must be a valid I/O Station task Number). The Literal is terminated by a NUL.

Implementation is identical to that of the general FLASH subroutine.

HALT

JSBR IZ 1777 037777

Halts execution of the issuing task. A message is flashed to all display screens in the format:

TASK NN AAAA, System 99 HALTED AT PP/CCSS A=XX/XXXX B=777777

where NN	=	Octal Task Number
AAAA	=	Program Name (in ASCII)
99	=	Application system number currently active
PP/CCSS	=	Address of JSBR + 1
XX/XXXX	=	Contents of A register on entry
777777	=	Contents of B register on entry

If the File Access system was 'locked' by the issuing task on entry to HALT then it will be unlocked. The task may be retarted at any chosen point by the Utility program 'R' described in Section 7 - OPERATING SYSTEM UTILITIES.

The Utility 'SE' will also display the above information until the task is restarted.

PRINT LINE

JSER IZ 1644 037644

The line set up within the issuing task's inner print buffer is scheduled for output. The inner print buffer is then space-filled and control returns to the step JSER + 1.

On entry all print control features are held in the A register as follows:

(N.B. see Appendix for full list of printer options.)

Bit	Control feature when Set
17	If set then a vertical tab has been requested prior to output and bits 15-1 are ignored.
16	No carriage return; this is useful if only control characters are to be sent to the device (see below) but its effect depends upon the destination device.
15	Characters per Inch request ie. the code to determine the number of characters printed per inch is held in bits 13-11. For example the DRE 8840 options are: 0 - 10 per inch (default) 1 - 12 per inch 2 - 13.3 per inch 3 - 15 per inch 4 - 17.1 per inch
14	Lines per Inch request ie. the code to determine the number of lines printed per inch depth is held in bits 10-9. Codes currently available are: 0 - 6 per inch (default) 1 - 8 per inch
13-9	Reserved according to bits 15-14
8-7	Reserved
6-1	If zero, then a form feed is required prior to printing, otherwise the number of linefeeds required prior to printing (1-63).

The functions requested in bits 15-1 may be combined in any manner.

In conjunction with the above, certain control characters may be inserted in the inner print buffer.

For example:

Code	Function
BEL	Alarm Bell
BS	Backspace
SO	Double Width on
SI	Double Width off

These codes are recognised only if bit 8 is set within the relevant byte within the inner print buffer; any ASCII control character encountered without bit 8 set in the inner print buffer will simply be printed as a space.

The use of the above options is dependent on the printer in use and responsibility for their use rests with the programmer. Printers without the required options will ignore the control codes. Where control characters are allowed and used then the programmer must remove any options he/she invokes and also take care in designing print layouts as any control characters found in the inner print buffer will not be reflected in the actual printout.

Standard line length is 132 characters and the inner print buffer (which the programmer may assume is space-filled initially) is located within any program at offset address 3600-. Information within an inner print buffer is transferred to one of two available 'outer print buffers' from which output is effected asychronously.

This subroutine handles the 'suspend', 'restart' and 'cancel' commands (See - OPERATING SYSTEM COMMANDS) automatically. In the event of a cancellation, return is not to the calling program but to the control Program. To disallow cancellation bit 17 on the Printer Termination Options control word should be set on program entry and this bit will remain. in force until a return is made to the Control Program.

RETURN TO CONTROL PROGRAM

JUMP Z 1402 023402

Completes the processing of the current print queue posting.

SPECIFY DEFAULT RESTART ADDRESS

JSBR IZ 1643 037643
P1 = Default Restart Address

Enables the programmer to specify the address of the instruction at which the issuing task is to restart in response to a request from utility program 'R' (Restart task).

The specification remains in force until superseded by a new call to this subroutine specifying a different address, or until the program returns to the Task Control Program (by "JUMP Z 1402").

If the issuing task is an I/O Station, P1 may be set zero to indicate that the address to be used is the escape point most recently specified by the "SPECIFY I/O STATION ESCAPE POINT" Routine.

NOTES

1. On program entry from the Task Control Program, the default restart address is 00/1402

2. A task may be restarted by Program "R" if:

   a) it is halted by a "HALT" subroutine call.
   b) it is waiting for a disk not on-line.

The operator may restart the task at any address, but if the need to restart is anticipated this routine is available to reduce the chances of an incorrect restart.

Example: If a program halts whilst an update flag is set, the restart should clear the flag.

SPECIFY PRINTER TERMINATION OPTIONS

JSBR IZ 1617 037617

The A-register should contain the required options, as follows:

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4  3  2  1
  |  |    |  0  0    0  0  0   0  0  0   <--------------->
  |  |    |                               Flash 'XXXX Printed
  |  |    |                               by n' message to
  |  |    |                               specified task only
  |  |    |                                 (0 = no flash)
  |  |    Force re-load of Program
  |  Flash 'XXXX Printed by n' message to all I/O Stations
  |              (Bit 16 overrides Bits 6-1)
  Print cancellation not allowed
```

This routine should be called at entry to the relevant Print program.

SPECIFY RE-ENTRY POINT

JSER IZ 1610 037610
P1 = Address of re-entry point

This subroutine allows the programmer to specify the address of the instruction to which the printer control program is to pass control if the next posting in the current print queue has the same program name as the current posting. The specified address overrides the entry point held in the printer program directory.

Whenever the printer control program encounters a posting in the same print queue with the same program name as the posting most recently processed, it re-enters the task partition without fetching a fresh copy of the program from the overlay library. If no re-entry point has been specified then that held in the printer program directory is used.

This subroutine can be very useful in terms of saving paper for instance. Under normal conditions the first items printed by a 'print' program are Form feed and Page Headings and if 'ad hoc' small sections of the same report are required then considerable waste is incurred in paper usage. By use of this subroutine, subsequent re-entries to the print program can be so directed as to omit form feeding and page heading until for example a program line count dictates that they are required.

SPECIFY TERMINATION ROUTINE

JSER IZ 1611 037611
P1 = Address of Termination Routine

This subroutine enables the programmer to specify the address of a routine which is to be called automatically by the printer control program when the batch of postings of the current program within the current print queue is terminated.

Termination of a batch occurs either when the Operator issues a 'P' command (See - OPERATING SYSTEM COMMANDS), or when the printer control program encounters a posting with a different program name.

The termination routine must be in memory when required and starts with a 'back-address' word. Return to Control is indirectly via the back address. The termination routine is ommitted in the event of an Operator-cancelled printout.

SUSPEND

JSBR IZ 1625 037625

Temporarily relinquishes control of the CPU to the OS Task Scheduler, which may allocate processing time to other tasks.

Control is returned to the step following "JSBR". Once a task gains control of the CPU it retains control until it relinquishes it to the Task Scheduler. This occurs automatically upon issuing any Input/Output request (Fetch, Spool, Get etc), but instances of heavy computation between I/O requests can arise, and these should be 'segmented' by "SUSPEND" calls.

In practical terms, printer programs should be literally sprinkled with SUSPEND calls on account of their low priority and buffering facilities (eg. after every "COMPUTE" call, after every three or four Binary to ASCII Conversions) whereas I/O Station programs are normally adequately segmented by I/O requests alone (it being the policy to place the computational work load onto printer programs).

It will be apparent that it is the programmers responsibility to ensure CPU time segmentation; a program loop will "hang" the entire system if it does not include an I/O or SUSPEND call.

Note that entry of the Task Scheduler automatically unlocks "LOCKED" direct access records, and may result in the contents of shared memory locations (such as 00/0045) being overwritten by other tasks.

SUSPEND FOR NUMBER OF CYCLES

JSBR IZ 1722 037722

The A-register, bottom byte, contains the number of cycles the calling program will suspend for. See SUSPEND subroutine.

In effect, the issuing task remains in the Suspend Queue through n Suspend calls, where n is the value of the A - register. The difference is that the calling program is not re-entered until all cycles are complete thus saving programming a count and also less CPU time is used in processing the call.

SECTION Z

OPERATING SYSTEM UTILITIES

CONTENTS

		PAGE
AID	Menu Notes Maintenance	7.1
BASH	Backing Storage Handler	7.2
BOOT	Bootstrap	7.16
C	Clear Line	7.17
CDL	Check Digit List	7.18
COMA	Configuration Maintenance	7.19
DATE	Advance System Date	7.39
DAY	Day of Week	7.40
DLU	Disk Label Utility	7.41
FE	File Enquiry	7.43
FLSH	Remove Flash Inhibit	7.44
FMOD	File Modifications	7.45
KICK	Kick/Reset Devices	7.46
MADP	Machine Data Printout	7.47
MAPS	Maintain Application Systems	7.48
OP	On-Line Program Amendment	7.66
POMM	Program and Overlay Module Maintenance	7.76
Q	Print Queue Enquiry	7.95
R	Restart Task	7.96
RECY	Recovery	7.97
RPT	Reprint	7.98
S	Send Message	7.100
SCAM	System Catalogue Maintenance	7.101
SE	System Enquiry	7.120
SMS	Swap Message Status	7.122
SUDS	Special Utility for Disk Security	7.123
SUFR	Special Utility for File Re-organisation	7.124
SUM	Arithmetic Calculator	7.125
TACK	Text Print on Labels	7.126

AID - MENU NOTES MAINTENANCE

This utility allows the User to enter their own menu and any useful information. It requires the presence of a direct access file (file I/D 200) consisting of up to n records of 32 words each. Each record contains one line of information and there may be up to 16 lines per menu. (Consequently the number of records allocated in the file must be in multiples of 16.)

Step	Prompt	Operator Action
1	MENU NUMBER	Enter the menu number required to be displayed and ACCEPT. Enter ACCEPT only for the next menu to be displayed. Enter L n and ACCEPT (n=0-15) to load line n of the menu. Continue at step 3.
2	16 lines of menu are displayed.	None; return to Step 1.
3	Cursor positions to start of required line	Enter up to 80 characters and ACCEPT. Continue at step 1.

BASH - BACKING STORAGE HANDLER

This program contains a variety of options concerned with the checking and maintenance of Disk backing storage. Due to the powerful nature of this program, access to it should be carefully controlled on customer sites.

The following menu is displayed on entry and on completion of any option:-

1 Read Disk or File
2 Zeroise File/Sector
3 Copy Sectors
4 Copy Operating System
5 Copy Configuration Table
6 Compare Sectors
7 Read Disk based F.S.B.
8 Scan Sectors

Step	Prompt	Operator Action
1	Option?	Enter required option number and ACCEPT. Enter ESC to return to PROGRAM?.

BASH

OPTION 1 READ DISK OR FILE

This utility is designed to display or print the contents of any disk or file in 6 digit octal numbers. The utility also includes facilities to follow through data in sequence either singularly or continuously, to make local or global amendments and to decode certain data elements.

The four digit number on the left of each row is the octal number of the first word in that row.

Step	Prompt	Operator Action
1	D-isplay or P-rint?	Enter D to display data. Enter P to print data. ACCEPT only to display. ESC to return to menu.
2	D-isk or F-ile?	Enter D for disk & continue at step 3. Enter F for file & continue at step 4. Enter ACCEPT only for file & continue at step 4. ESC to return to step 1.
3	Disk no.	Enter a valid disk no. & continue at step 9. ESC to return to step 2.
4	File ID.	Enter a valid file ID. ESC to return to step 2.
	If the file selected is an indexed sequential file then ISAM ACCESS will be displayed and continue at step 6.	
5	Test loaded records?	ACCEPT if a "not loaded check" is to be performed on the record. ie. the slot number is held in the first word of the record. REJECT if the loaded test is to be suppressed. ESC to return to step 4. Continue at step 9.
6	ACCEPT for Data or REJECT for Index only	ACCEPT to display/print the data record. REJECT to display/print the index record only. ESC to return to step 4.

7 Is the key in Ascii? ACCEPT for entry of an ascii key.
REJECT to allow entry of the key in words & continue at step 9.
ESC to return to step 4.

8 ACCEPT for a space-filled key or REJECT for a zero-filled key

ACCEPT for input to be space filled.
REJECT for input to be zero filled.
ESC to return to step 4.

9 ACCEPT for single access or REJECT for continuous access.

ACCEPT to access records/sectors singularly allowing operator intervention after each one.
REJECT to access records/sectors continuously in a specified sequence until the end is reached or the sequence is terminated.
ESC to return to step 2.

10 The following prompts will be displayed depending on the previous requirements and subsequent sector/record accesses.

Enter Slot no.	Enter the decimal record slot no. or '0' & the octal record slot no. for a direct access file.
Enter Sector no.	Enter the decimal sector no. or an '0' & the octal sector no.
Enter Ascii key	Enter the ascii key.
Enter Key word 1	Enter the contents of word one of the key in decimal or '0' & octal. This prompt will be repeated as required for the key size of the selected file or may be terminated by ACCEPT only.
A-mend	Enter A to amend the displayed sector/record & continue at step 17.
S-equence	Enter S to initiate a sequential access & continue at step 15.
ACCEPT to continue	ACCEPT only to access the next sector/record in the prescribed sequence or to re-read current sector/record.
	ESC to return to step 9.

If the print option was selected continue at step 19.

If the single access mode was selected the sector/record will be displayed. The space bar may be used to stop and start the display as required. ESC will return the program immediately to step 10.

If continuous access mode was selected.

11 Global amendments? ACCEPT to allow amendments to every sector/record accessed.
REJECT to inhibit amendments & continue at step 14 .
ESC to return to step 10.

12 Offset Enter octal offset for single word amendment.
Enter 'D' & octal offset for double word amendment.
Enter 'T' & octal offset for triple word amendment.
ESC to return to step 11.

13 Contents Enter decimal or '0' & octal value.

The sector/records will be displayed in the required sequence. The space bar may be used to stop and start the display as required.
ESC will stop the display after the current sector/record.

14 A-mend or ACCEPT to continue
Enter A to amend the displayed sector/record as described in steps 17 - 18. This amendment will be carried out on the displayed sector/record only before the routine returns to step 14.
ACCEPT to continue with the display & return to step 14.
ESC to return to step .

15 If an ISAM file was selected then a sequential access will be set up based on the key.
If sector access was selected then the sequence automatically becomes physical.

L-ogical or P-hysical sequence
Enter L to initiate a logical (chained) sequence.
Enter P to initiate a physical sequence and return to step 10.
ESC to return to step 10.

16	Offset	Enter octal offset of word containing next slot. ESC to return to step 15.
17	Offset	Enter octal offset for single word amendment. Enter 'D' & octal offset for double word amendment. Enter 'T' & octal offset for triple word amendment. ESC to return to step 10.
18	Contents	Enter decimal or '0' & octal value. Enter D to unpack word at offset as a date. Enter L to unpack from word at offset as ascii. Enter M to unpack from word at offset as metacode. Enter N to unpack from word at offset as a decimal value. ESC to return to step 17.

The contents of the displayed sector/record may be amended and the sector/record will be redisplayed with the amendments.

19 If the continuous access mode was selected the operator will be prompted to enter an end sector/slot/key to terminate the print.

Proceed?	ACCEPT to print. REJECT to return to step 1.

The sectors/records will be printed including the global amendments if any were entered.

BASH

OPTION 2 ZEROISE FILE/SECTOR

The following sub-menu is displayed:

1. File
2. Range of files
3. Sectors

Step	Prompt	Operator Action
1	Select one option	Enter option number and ACCEPT. Enter ESC to return to main menu.

Zero single File

Step	Prompt	Operator Action
1	File ID	Enter octal file identifier and ACCEPT. The file type referred to should be either a Direct Access file or an Indexed Seq. file. Enter ESC to return to sub-menu.

If an Indexed Sequential file was selected, then the constituent Direct Access files will automatically be cleared; go to step 4.

2	First rec no.	Enter the first record number to be cleared and ACCEPT. Enter ALL if the whole file is to be cleared and ACCEPT; go to step 4. Enter ESC to go to step 1.
3	Last rec no.	Enter the last record number to be cleared and ACCEPT. Enter ESC to go to step 2.
4	If the requested file(s) have disk based FSBs then: All disk based FSBs to be cleared?	Enter ACCEPT to clear the FSBs. Enter REJECT to leave the FSBs unchanged.
5	Proceed?	Enter ACCEPT to clear the file. Enter REJECT to go to step 1.

The requested record(s), and FSBs are cleared to Zeroes; Program returns to step 1.

Zero a range of files

Step	Prompt	Operator Action
1	First file ID	Enter the first file ID to be cleared and ACCEPT. Enter ESC to return to the sub-menu.
2	Last File ID	Enter the last file ID to be cleared and ACCEPT. Enter ESC to go to step 1.
3	Proceed?	Enter ACCEPT to clear the file(s). Enter REJECT to go to step 1.

The requested File(s) are cleared to zeroes; go to step 1.

Notes.

a) All related disk based FSBs will be cleared.
b) The range of files is considered in strict numerical order, i.e. for a range 021-022, only files 021 and 022 will be cleared.
c) Only Direct Access files may be cleared within a range, though the range may include Direct Access files which are used within an Indexed Sequential structure.
d) Any other file types encountered within the range are ignored.

Please note that it is possible to escape from a range of files by pressing the escape key (ESC). The option will stop after completing a file and warn the operator accordingly.

Zero absolute sectors

Step	Prompt	Operator Action
1	Disk no.	Enter octal disk number and ACCEPT. Enter ESC to return to sub-menu.
2	Start sector(octal)	Enter sector number at which clearing is to start and ACCEPT. Enter ESC to go to step 1.
3	Sectors(decimal)	Enter number of sectors to be cleared and ACCEPT. Enter ESC to go step 2.
4	Proceed?	Enter ACCEPT to start clearing. Enter REJECT to go to step 1.

The requested sector(s) are cleared to Zeroes; go to step 1.

N.B.
This Option calls a Zero Overlay which may be incorporated into user programs if required. The module number is 15 and may be run in an input or print task. It resides on the Utilities Library. It occupies columns 30 and 31 and is called in the following manner:

```
        JSBR    1000
        P1              Offset address of control block.
```

Control Block

Word 1	Disk no. Bit 17 if protection override.
Word 2	Start Sector(Octal)
Word 3	Number of sectors(Decimal)
Word 4	Start Displacement/End Displacement.
Word 5	Memory buffer offset address.

Word 4 — BITS 16-9 contain the word in the range 0-127 at which the first sector will start to be zeroised.
BITS 8-1 contain the word in the range 0-127 which is the last word to be zeroised in the last sector. If zero the whole of the last sector will be zeroised.

Word 5 — This overlay requires a buffer size of 8 sectors.

BASH

OPTION 3 COPY SECTORS

This utility will copy any number of sectors from any part of a disk onto any part of another or the same disk. The receiving sectors are re-read and a software 'hash check' is carried out. Copying is carried out 8 sectors at a time using a 1K memory buffer.

Step	Prompt	Operator Action
1	Sectors(decimal)	Enter the number of sectors to be copied. ACCEPT only to default to the previous number of sectors copied. ESC to terminate copy.
2	Source disk	Enter the octal disk number from which the sectors are to be copied.
3	Start sector(octal)	Enter the first octal sector to be copied.
4	Target disk	Enter the octal disk number to which the sectors are to be copied.
5	Start sector(octal)	Enter the first octal sector to which the sectors are to be copied.
6	Proceed?	ACCEPT only to commence copying. Enter 'P' & ACCEPT to commence copying and override any protected sectors. REJECT to return to step 1.

N.B.
If copying sectors from low to high sector numbers on the same disk and the source and target overlap, NO corruption will occur.

BASH

OPTION 4 COPY OPERATING SYSTEM
OPTION 5 COPY CONFIGURATION TABLE

These options will copy either the operating system or the configuration table from any disk to any disk. No protection override is required.

BASH

OPTION 6 COMPARE SECTORS

This utility will compare any number of sectors on either the same disk or two seperate disks. Where a discrepancy arises the disk and sector numbers will be displayed.

Step	Prompt	Operator Action
1	No. of sectors(nnnnnn)	Enter the decimal number or '0' & the octal number of sectors to be compared. ACCEPT to default to displayed number(nnnnnn).
2	First disk(nnn)	Enter octal disk number of first disk. ACCEPT to default to displayed disk number(nnn).
3	Start sector(octal)(nnnnnn)	Enter octal sector number from which comparison is to start. ACCEPT to default to displayed start(nnnnnn).
4	Second disk(nnn)	Enter octal disk number of second disk. ACCEPT to default to displayed disk(nnn).
5	Start sector(octal)(nnnnnn)	Enter octal sector number at which comparison is to be made. ACCEPT to default to displayed sector(nnnnnn).
6	Proceed?	ACCEPT to start comparison. REJECT to return to step 1.

The sectors will be compared and any discrepancies will be displayed. The space bar may be used to stop and start the display and ESC will terminate the comparison.

BASH

OPTION Z READ DISK BASED FSB

Step	Prompt	Operator Action
1	File ID or A-mend	Enter file ID of required FSB. Enter A to amend displayed record and continue at step 2. Enter ESC to return to menu. Enter ACCEPT to re-read current record.

The required FSB details will be displayed.

2	Offset	Enter octal offset for single word amendment. Enter 'D' & octal offset for double word amendment. Enter 'T' & octal offset for triple word amendment. ESC to return to step 1.
3	Contents	Enter decimal or 'O' & octal value if single word amendment. The value must be decimal if double or triple word amendments. Enter 'N' to unpack from word at offset as a decimal value. Enter 'D' to unpack from word at offset as a date.

The contents of the displayed record will be amended and redisplayed with the amendments.

OPTION 8 SCAN SECTORS

Step	Prompt	Operator Action
1	Scan Disk(777)	Enter required Disk number and ACCEPT. Note the Disk MUST be on line. Enter ACCEPT to use default Disk. Enter ESC to return to menu.

The Disk type, number of sectors (decimal) and maximum sector address (octal) are displayed.

2	Base sector(777777)	Enter sector address at which the scan may commence and ACCEPT. Enter ACCEPT to use default sector address. Enter ESC to go to step 1.
3	Transfer length, 1 to 8 (n)	Enter the number of sectors to be transferred per disk access and ACCEPT. Enter ACCEPT to use default length. Enter ESC to go to step 2.
4	Number of transfers(99999)	Enter the number of actual disk transfers required (each of length stated in step 3.) and ACCEPT. Enter ACCEPT to use default number. Enter ESC to go to step 2.

The number of sectors to be scanned (decimal) and the last transfer start sector address (octal) are displayed.

5	R-ead only or W-rite after read(R)	Enter "R" and ACCEPT to read sector(s) and proceed. Enter "W" and ACCEPT to perform a write operation, after the initial read, followed by a re-read. Enter ACCEPT to use the default transfer type.

With this option the current transfer is Hash-checked through the read-write-read procedure and the operator is alerted if a Hash fail occurs. The transfer is repeated until the hash appears correct.

6	Scan F-orwards, B-ackwards or S-awtooth(F)	Enter "F" and ACCEPT to start the scan from the base sector and work forwards. Enter "B" and ACCEPT to start the scan from the computed end sector and work backwards. Enter "S" and ACCEPT to alternately scan forwards and backwards working inwards. Enter ACCEPT to use the default scan type.
7	O.K.?	Enter ACCEPT to start scan. Enter REJECT to re-input data. Enter ESC to return to menu.

The current transfer start sector address is displayed. The scan process is repeated indefinitely, until the ESC key is pressed; go to step 1.

BOOT = BOOTSTRAP

Step	Prompt	Operator Action
1	Password?	Enter PASSWORD 3, from Passwords File, and ACCEPT. Enter ESC to return to PROGRAM?.
2	Program will report any other active Task; if any then:	
	Continue?	Enter ACCEPT to proceed. Enter REJECT or ESC to return to PROGRAM?.
3	Options?	Enter any or all of the following options, separated by a comma or space, terminated by ACCEPT: C = Stow C-ontrol Record S followed by a decimal value = Bootstrap from the specified S-urface. D followed by 0, 1 or 2 = Bootstrap from specified D-evice. R = Initiate R-ecovery from specified device, surface. T followed by a decimal value = Bootstrap single T-ask; the value being the Task type. Enter ESC to go to PROGRAM?.

All selected options are re-displayed.

Note.
If "Stow Control Record" option was selected, then this is performed immediately, if the relevant File exists.

4	Proceed?	Enter ACCEPT to start the bootstrap. Enter REJECT to go to Step 1. Enter ESC to go to PROGRAM?.

N.B.
The Machine MUST be idle otherwise file corruption may occur.

C = CLEAR LINE

This utility clears the 25th line on an ADM-32 VDU. No further operation is necessary after calling this utility.

CDL = CHECK DIGIT LIST

This utility will produce, for a given range of numbers, a list of those numbers and their corresponding check digit.

Step	Prompt	Operator Action
1	First no.?	Enter the number at which the printout is to start & ACCEPT. Enter ESC to return to PROGRAM?.
2	Last no.?	Enter the number at which the printout is to end & ACCEPT.

The check digit list request is posted to the assigned queue and the Program returns to Step 1.

COMA - CONFIGURATION MAINTENANCE

Each installation has its own configuration table(s), which are referred to by the "initiator" when the system is 'boot-strapped' in. The table(s) are used to convert the Standard Operating System into a working system in memory appropriate to the installation. Access to this program should be carefully controlled on customer sites.

COMA has no effect on the operating system currently running; it only comes into being as a result of bootstrapping the system again.

Step	Prompt	Operator Action
1	Password?	Enter the systems password and ACCEPT.
2	Disk No.	Enter the system disk number and ACCEPT.

The following menu is displayed on completion of any option:

1 Resource Tables Allocation
2 O.S Workspace Allocation
3 Machine Data
4 Backing Storage Allocation
5 Master Task Definitions
6 Shared Disk Buffer Control Blocks
7 Resident Overlays
8 Masterfiles Definition
9 Machine Literals
10 Master Task Enquiry
11 Resource Tables Allocation Print
12 Configuration Map Display
13 Configuration Map Print
14 Device Types Display
15 Application F.C.B. Workspace Estimate
16 Shared Buffer Usage Display
17 Shared Buffer Usage Print

3	Option?	Enter option number and ACCEPT. Enter ESC to go to step 1.

Please note that the options have been set out in a logical sequence. When setting up a new system (or indeed amending an existing one) the menu options should be selected in numerical order.

COMA

OPTION 1 RESOURCE TABLES ALLOCATION

The current Configuration Table layout will be displayed.
e.g.

ORIGIN	ENTRIES		DESCRIPTION
1 1	5V	2	O.S. WORKSPACE
2 12	3F	1	BACKING STORAGE TYPE TABLE
3 16	15V	2	TASK PARTITIONS
4 47	10V	1	I/O STATIONS
5 58	5V	1	PRINTERS
6 0	0V	3	BACKGROUND TASKS
7 64	1F	8	MASTERFILES DEFINITION
8 0	0V	1	SPECIAL APPLICATIONS INTERFACE AREA
9 76	10V	3	SHARED BUFFER CONTROL BLOCKS
10 152	1V	3	RESIDENT OVERLAYS
11 156	1V	1	SYSTEM TABLE

Column 1 is the field number.
Column 2 is the start word within the table
Column 3 is the number of entries and a V-ariable or F-ixed indicator.
Column 4 is the number of words required for each entry.

The total words required for each field are:

(Column 3 X Column 4) + 1

Column 5 is the description of each field.

Step	Prompt	Operator Action
1	FIELD?	Enter U and ACCEPT to update table and return to menu. Enter field no. and ACCEPT and continue at step 1. Enter ACCEPT to redisplay table. Enter ESC to return to menu.
2	ORIGIN: 999 ?	Enter ACCEPT to default to existing origin. Enter origin and ACCEPT. Enter value of zero to delete table, if allowed.
3	ENTRIES nn ?	Enter ACCEPT to default to existing entry. Enter required number of entries and ACCEPT.

The table will be re-configured and redisplayed for confirmation. Return to Step 1.

COMA

OPTION 2 O.S. WORKSPACE ALLOCATION

Step	Prompt	Operator Action
1	Switched memory banks: 9 ?	Enter number of switchable 32K memory banks available or zero if none.

If switchable memory is selected, COMA will force all resources except Task Partitions to be defined below 32K; go to Step 3.

2	Assumed Memory size nnK?	Enter ACCEPT to default to assumed memory size. Enter memory size in 4K blocks and ACCEPT.
3	BASE nn/nn?	Enter ACCEPT to default to displayed location. Enter base & ACCEPT. Enter ESC to display rest of table and continue at step 5. Enter zero to end input and ACCEPT.
4	LENGTH (COLUMNS) 32?	Enter ACCEPT to default to displayed length. Enter number of columns & ACCEPT

Prompts 3 and 4 are repeated until all workspace entries have been allocated or the ESC key is hit.

5	O.K.?	Enter ACCEPT to update and return to menu. Enter REJECT to return to menu.

COMA

OPTION 3 MACHINE DATA

Step	Prompt	Operator Action
1	File Table 0 initiates with System nn ?	Enter ACCEPT to default to the system number displayed (nn). Enter a valid system number & ACCEPT. This prompt will be repeated for all file tables available to the system.
2	PROCESSOR CONFIGURATION TYPES: S = SINGLE PROCESSOR A = DUAL PROCESSOR, MACHINE A B = DUAL PROCESSOR, MACHINE B Configuration type: a?	Enter ACCEPT to default to the type displayed (a). Enter configuration type and ACCEPT.
3	Task 01 Start-up program: aaaa?	Enter ACCEPT to default to the program displayed (aaaa). Enter a valid program name (max. 4 A/N chars) and ACCEPT. Enter NONE and ACCEPT for no start-up program.
4	User Routine @ *NONE* ?	The O.S. may be instructed to call a user-written subroutine whenever any Task is about to be Scheduled. This Routine must be allocated within a Resident Overlay. Enter ACCEPT to default to the current value. Enter the memory start address of the routine. Enter zero to delete the entry.

N.B.
No checking is performed on the relevant routine; seek Systems advice if considering using this feature.

5	User Workspace (per Task) = 999 word(s)?	An area of memory, available for applications use, may be allocated per Task. This memory is cleared to zeroes on Boot-strapping the machine, and then is the responsibilty of the Applications Programmer. See sub-routine descriptions, Section 3. Enter ACCEPT for no change. Enter number of words, 0 to 127, of memory required per Task.
6	O.K.?	Enter ACCEPT to update the system data and return to menu. Enter REJECT to return to menu.

COMA

OPTION 4 BACKING STORAGE ALLOCATION

The current storage allocation will be displayed.
eg.

UNIT	DRIVES	DISKS	SECTORS PER DISK	DEVICE TYPE
70	1	2	12992 (0-31277)	DD1600 FIXED/ EXCHANGEABLE
71	5	10	12992 (0-31277)	D8000 MULTI-PLATTER CARTRIDGE.
72				

Step	Prompt	Operator Action
1	UNIT?	Enter unit number (70, 71, 72) and ACCEPT. Enter U and ACCEPT to update the storage allocation and return to menu.
2		Enter ACCEPT to default to DD1600 disk drive. Enter disk type and ACCEPT.

Currently, if D8000 no further input is required. If DD1600 or DD818, then:

3	DISKS ON-LINE: n?	Enter ACCEPT to default to number displayed (n). Enter number of disks (max. 8) and ACCEPT.

COMA

OPTION 5 MASTER TASK DEFINITIONS

Step	Prompt	Operator Action
1	Task 01 Type aaaaaa?	Enter ACCEPT to default to displayed device type (aaaaaa). Enter a device type (see Option 14) and ACCEPT. NOTE After having entered a printer device type you cannot revert to an input device. Enter F and ACCEPT to terminate definition and go to step 13. Enter ESC to display rest of existing definitions and go to step 13. N.B. Table NOT updated.

- If the device type is a printer the next prompt is ignored.

2	(Flashes? (No Flashes?	Indicates whether flash messages are to be received by this task. Enter ACCEPT to retain displayed flash status. Enter REJECT to change displayed flash status.

If this site only contains 1 file table the next prompt is ignored.

3	F/Table n?	Enter ACCEPT to default to displayed file table number (n). Enter file table number (0 to max. for this machine) and ACCEPT. This is the file table used at initialisation for this task.

If task 01, then it is assumed that the partition base is page 05, bank 0 and its device codes are 50 and 40. Consequently prompts 4, 5, 6 and 7 are ignored.

If Switched memory has not been selected (see Option 2) go to step 5.

4	Bank n ?	Enter ACCEPT for no change. Enter bank number and ACCEPT.
5	Base nn?	Enter ACCEPT to default to displayed partition base (nn). Enter partition base and ACCEPT.

6	I/P nn?	Enter ACCEPT to default to displayed input device. Enter input device code, or zero and ACCEPT.
7	O/P nn?	Enter ACCEPT to default to displayed output device. Enter output device code and ACCEPT.

The above prompts will be repeated for each task entered. All input tasks must be declared first, followed by printer tasks and finally any background tasks.

8	BACKGD?	Enter ACCEPT to define background partition. Enter F and ACCEPT to terminate definitions and go to step 13.
9	Initiate with aaaa	Enter ACCEPT to default to displayed program (aaaa). N.B. no checking performed; the selected program should reside in the relevant PRINT Directory. Enter program name and ACCEPT.

If Switched memory has not been selected (see Option 2) go to step 11.

10	Bank n ?	Enter ACCEPT for no change. Enter bank number and ACCEPT.
11	Base nn?	Enter ACCEPT to default to displayed base (nn). Enter memory base address of background partition and ACCEPT.

If this site only contains 1 file table the next prompt is ignored.

12	F/Table n?	Enter ACCEPT to default to displayed table number (n). Enter file table number (0 to max for this machine) and ACCEPT. This is the file table used at initialisation for this task.

The above prompts 8-12 are repeated for each background task as detailed in the resource table.

The master task definitions are redisplayed for confirmation:

13	O.K.?	Enter ACCEPT to update the definitions and return to menu. Enter REJECT to return to menu.

COMA

OPTION 6 SHARED DISK BUFFER CONTROL BLOCKS

Step	Prompt	Operator Action
1	BASE nn/nn?	Enter ACCEPT to default to displayed page and column address. Enter base address and ACCEPT. Enter ESC to display rest of existing table and go to step 4. Enter 0 and ACCEPT to terminate block and go to step 4.
2	LENGTH (SECTS) n?	Enter ACCEPT to leave unchanged. Enter length in sectors (max. 8) and ACCEPT.
3	FOR UNIT nn?	Enter ACCEPT to default to displayed disk unit. Enter disk unit and ACCEPT.
4	O.K.?	Enter ACCEPT to update control blocks and return to menu. Enter REJECT to return to menu.

COMA

OPTION Z RESIDENT OVERLAYS

Step	Prompt	Operator Action
1	O'LAY NO nnnnnn?	Enter overlay number and ACCEPT. Enter zero to end input and ACCEPT.

The program will check that the overlay is valid and will display:

LENGTH (SECTS) nn LOADS AT nn/nnnn

The above prompt will be repeated until all resident overlays have been declared.

2	O.K.?	Enter ACCEPT to update allocation and return to menu. Enter REJECT to return to menu.

N.B.
All resident overlays must reside on the Utilities Library (Slave Library 3) with which Task 01 initiates. It is also recommended that copies of such overlays are kept on the associated Application System so as to avoid their loss when the Utilities are updated.

COMA

OPTION 8 MASTER FILES - DEFINITION

Step	Prompt	Operator Action
	The current catalogue details are displayed.	
1	RE-DEFINE AS:	Enter ACCEPT to leave catalogue displayed. Enter catalogue data-set name and ACCEPT.
	If a new catalogue data-set name is entered the details will be displayed.	
	The current system file details are displayed.	
2	RE-DEFINE AS:	Enter ACCEPT to leave system file unchanged. Enter system file data-set name and ACCEPT.
	If a new system file data-set name is entered the details will be displayed.	
3	O.K.?	Enter ACCEPT to redefine master files and return to menu. Enter REJECT to return to menu.

COMA

OPTION 9 MACHINE LITERALS

Step	Prompt	Operator Action
1	INSTALLATION NAME: a--------a?	Enter ACCEPT to default to displayed name. Enter name (max. 16 chars) and ACCEPT.
2	SYSTEMS PASSWORD?	Enter ACCEPT to retain existing systems password. Enter password (max. 5 chars) and ACCEPT.
3	ERROR MESSAGE TEXT: a------a?	Enter ACCEPT to default to displayed text. Enter text (max. 15 chars) and ACCEPT.
4	O.K.?	Enter ACCEPT to update password and literals; return to menu. Enter REJECT to return to menu.

COMA

OPTION 10 MASTER TASK DEFINITIONS ENQUIRY

Displays all relevant details regarding the Task partitions.

COMA

OPTION 11 RESOURCE TABLES ALLOCATION PRINT

A request for a print of the resource table allocations as displayed in Option 1 is posted to the currently assigned queue.

COMA

OPTION 12 CONFIGURATION MAP DISPLAY

Displays a detailed layout of memory as defined by the currently chosen Configuration Table. This display does not use any other information than that contained within this Configuration Table and so any Configuration can be checked before it is initiated.

As well as producing a Memory Map, information on the Configuration Table itself is displayed, together with an estimate of OS Workspace required by the Configuration EXCLUDING Application files (x11-x77)

At the end of the display the program will optionally calculate the maximum amount of workspace required to support the configured number of File Tables. This option should only be used when the configuration in question is also the current running configuration.

This allows the installer to completely check a Configuration before attempting to use it.

See Option 15 for estimation of Application file control blocks workspace requirements.

COMA

OPTION 13 CONFIGURATION MAP PRINT

Requests a hard copy version of Option 12.

COMA

OPTION 14 DEVICE TYPES DISPLAY

Displays the associated mnemonics for peripherals supported by this OS.

COMA

OPTION 15 APPLICATION F.C.B. WORKSPACE ESTIMATE

Allows the user to estimate the extra OS workspace requirements for a given System over and above that calculated as the 'machine' requirements.

Also, by entering the character "R" instead of a valid system number, the maximum amount of workspace required to support the number of File Tables configured can be calculated.

Used in conjunction with Option 12, a complete workspace estimate for a given installation is possible.

COMA

OPTION 16 - SHARED BUFFER USAGE DISPLAY

For each selected Shared Buffer, a display of all files, all Systems, which use it is produced.

N.B.
This Option should only be used when the Configuration Table in question is also the currently running Configuration.

COMA

OPTION 17 SHARED BUFFER USAGE PRINT

Requests a hard copy version of Option 16, all configured Shared buffers.

DATE = ADVANCE SYSTEM DATE

Step	Prompt	Operator Action
1	DD MMM YY DDD?	System date is displayed eg. 20 FEB 70 TUE. Enter ACCEPT only to set the System date to the date displayed and return to PROGRAM? Enter REJECT to display the following day's date eg. 21 FEB 70 WED, Return to step 2. Enter ESC to return PROGRAM? without affecting the System Date.

DAY - DAY OF WEEK

Step	Prompt	Operator Action
1	DATE	Enter a date in the form dd mm yy and ACCEPT. Enter ESC to return to PROGRAM?
2	DDD	The day of the week (eg. TUE) for the given date in the 20th century is displayed Return to Step 1.

DLU - DISK LABEL UTILITY

Step	Prompt	Operator Action
1	PASSWORD	Enter the system password and ACCEPT.
2	DEVICE (000070)	Enter the device code in the range 70-72 and ACCEPT or ACCEPT only to default to the displayed device.
3	SURFACE (00)	Enter the surface number and ACCEPT or ACCEPT only to default to the displayed surface.
4	The utility will attempt to read the label from the selected device/surface and will display either DISK PREVIOUSLY LABELLED or DISK NOT PREVIOUSLY LABELLED	
5	LABEL DISK?	If the disk has been labelled then the label may be displayed by use of the REJECT key. Enter ACCEPT to label the disk. If the disk was previously labelled then the previous contents will be displayed in parenthesis against each prompt and the operator may default to the original contents by use of the ACCEPT key only.
6	DISK NO.	Enter disk number in the range 1-277 and ACCEPT.
7	MASTER?) SECURITY?)	Enter ACCEPT to retain status or REJECT to change.
8	DAILY SECURITY NO OF COPIES	Enter the number of daily security copies in the range 1-7 and ACCEPT.
9	SECURITY DISK NO./n	This prompt will be repeated n times where n is the number of copies. Enter the security disk number in the range 1-277 and ACCEPT.
10	SPECIAL SECURITY NO OF COPIES	Enter the number of special copies in the range 1-7 and ACCEPT.

11	SECURITY DISK NO./n	This prompt will be repeated n times where n is the number of copies. Enter the security disk number in the range 1-277 and ACCEPT.
12	CONTINUE?	Enter ACCEPT to label disk and redisplay the label details. Enter REJECT to cancel the input. Return to step 2.

N.B.
If in a recovery situation the copy sequence number may need to be changed to correctly maintain the copying sequence. This is acheived by entering a 'R' and the number of copies. The sequence number will now be prompted for immediately after the number of copies.

FE = FILE ENQUIRY

This utility is available to show the file usage in a system. All files previously selected to be monitored (see SCAM utility) will be displayed on entry to the utility.

After the first screen has been displayed the utility will prompt MONITOR?

Entry of 'REJECT' will redisplay all the files to be monitored. If all the files cannot be contained on one screen then the display will be scrolled.

Entry of 'ACCEPT' will continuously repeat the display of all the files to be monitored. If all the files cannot be contained on one screen then the display will be truncated after the first screenful.

Enter 'ESC' to return to PROGRAM?

It is also possible to select individual files, a range of files or a mixture of both at the MONITOR prompt.

Enter individual file ID's separated by commas or a range separated by a dash (-). The numbers may be in any order except that the end of a range must not be less than the start:

i.e. 11,211,212,13,21-25,131-135,314-337

The display may be stopped and restarted by use of the space bar. The escape key (ESC) will stop the display and return the utility to the MONITOR? prompt.

FLSH - REMOVE FLASH INHIBIT

If an I/O Station has FLASH messages from the Operating System suppressed then this utility may be used to re-instate them from another screen.

Step	Prompt	Operator Action
1	I/O Station Task no.?	Enter the task number for which FLASHes are to be re-instated and ACCEPT. Enter ESC to return to PROGRAM?

FMOD - FILE MODIFICATION

If a file organisation is deemed incorrect from the point of view of record length them FMOD may be used to duplicate a file record by record to a new file with either a smaller or larger record length.

A temporary file, with the new definition is 'catalogued' and identified to the current File Table by SCAM and MAPS. FMOD will transfer the file contents. Then, delete the old file and rename what was the temporary file.

If the Target record length is less than the Source, the record is truncated. If the Target record is larger, it is zero filled.

A message is flashed to all screens at the start and end of the transfer; no checking is done and validation is the responsibility of the programmer.

Step	Prompt	Operator Action
1	SOURCE FILE ID	Enter file identifier and ACCEPT.
2	TARGET FILE ID	Enter file identifier and ACCEPT.

KICK - KICK/RESET DEVICES

From time to time an I/O Station or printer may not respond to usual commands, this may be caused by a build up of static electricity. This utility attempts to reset hardware status of all devices on the system.

Step	Prompt	Operator Action
1	PROCESS?	Enter ACCEPT only to attempt to reset devices. Enter REJECT only to return to PROGRAM?.

Any devices found to be in an unorthodox state are reported to the operator.

MADP - MACHINE DATA PRINTOUT

Allows a complete list of all the Configuration(s) and Application(s) for a given site.

This enables the Installer/Programmer to have detailed information regarding all Customer sites available at Branch offices.

These listings should be updated whenever changes are made at Customer sites.

Step	Prompt	Operator Action
1	Configuration from Disk nnn?	Enter ACCEPT to produce both a listing of the possible Configuration from that C.T. and also the layout of the C.T. itself. Enter REJECT to omit this disk's C.T. Enter ALL and ACCEPT to request listings for all disks on-line.
2	Start System Number?	Enter ACCEPT to start printouts from beginning of Systems Tables File. Enter System Number from which print is tc begin.
3	End System Number?	Enter ACCEPT only to print to end of Systems Tables File. Enter System Number at which print will stop.

For each Application System a list of its associated Data-Set definitions together with both an alphabetic and numeric Program Map will be produced.

Following the System printouts, the Catalogue will be printed alphabetically and by disk.

4	O.K.?	Enter ACCEPT to request above prints. Enter REJECT to go to Step 2.

MAPS - MAINTAIN APPLICATIONS SYSTEM(S)

This suite of programs enables the user to tailor a system in terms.of program, catalogue and data files. Having set up several systems on disk it is possible to load a system onto a file table held in memory to be accessed by one or several task partitions.

Password 2, from the Passwords file, is required to gain entry to this Program.

The following menu is displayed on entry to the utility and on completion of all options:

1	Files Definition
2	System(s) Data
3	System Definition Print
4	Active System(s) Enquiry
5	System Delete
6	Reload System
7	Defined Systems
8	Inhibit Class Names amend
9	Switch Inhibit Class
10	Passwords Maintenance
11	Serial Number Maintenance
12	Set Print Queue Definition
13	Delete Print Queue Definition
14	Duplicate System Definition

Step	Prompt	Operator Action
1	Option?	Enter option number and ACCEPT. Enter ESC to return to PROGRAM?.

MAPS

OPTION 1 FILE DEFINITION

Allocation of file ID numbers to catalogue entries for the specified system. This affects a system on disk only and will come into effect when a system is reloaded.

Step	Prompt	Operator Action
1	System Number?	Enter system number and ACCEPT. Enter ESC to return to menu.

If the system exists the name will be displayed for confirmation.

2	Major File I/D: 00?	Enter ACCEPT to default to displayed file ID number. Enter a major file ID (range 00-77) and ACCEPT.

If the major file ID has already been allocated then the details of both major and minor, if applicable, file details will be displayed for confirmation.

3	Required Entry?	Enter ACCEPT to retain existing allocation. Enter a catalogue name and ACCEPT. Enter "NONE" and ACCEPT to remove any allocation.

Details of the allocation will be displayed for confirmation. If no changes were made the next prompt will be ignored.

4	O.K.?	Enter ACCEPT to update the system. Enter REJECT to go to the next file ID.

The major file ID will be incremented and the above prompts repeated from step 2.

MAFS

OPTION 2 SYSTEM DATA

Step	Prompt	Operator Action
1	SYSTEM NUMBER?	Enter system number and ACCEPT.

The utility will decide if this is a new system or an existing one and display LOAD and AMEND accordingly.

2	NAME: a------a?	Enter ACCEPT to default to displayed name. Enter the system name (max. 16 chars) and ACCEPT.
3	NUMBER OF SPOOL QUEUES ALLOWED: n?	Enter ACCEPT to default to displayed number (n). Enter number of queues and ACCEPT.
4	NUMBER OF "PLAIN PAPER" QUEUES: n?	Enter ACCEPT to default to displayed number (n). Enter number of queues and ACCEPT.
5	NUMBER OF EXTENSION QUEUES: n?	Enter ACCEPT to default to displayed number (n). Enter number of queues and ACCEPT.
6	PASSWORD n	Enter ACCEPT to retain existing password. Enter a password (max. 9 chars) and ACCEPT.

The above prompt is repeated for four system passwords.

7	NUMBER OF PRINTERS CONFIGURED: n PRINTERS CURRENTLY AVAILABLE TO THIS SYSTEM: a, b, . . . REQUIRED PRINTERS?	Enter ACCEPT to retain displayed printers (a, b, . . .). Enter printers A, B etc. and ACCEPT. Enter NONE and ACCEPT to remove all printers. Enter ALL and ACCEPT to include all printers.
8	TASK 01 DEFAULT PRINTER IS: a	Enter ACCEPT to retain displayed printer (a). Enter printer identity and ACCEPT. Enter NONE and ACCEPT to remove printer default from this task.

The above prompt is repeated for each input task in the installation.

9 FILE TABLES CONFIGURED: n
FILE TABLES CURRENTLY AVAILABLE TO THIS SYSTEM: n,n,..
FILE TABLES REQUIRED?

This is to indicate the file tables onto which this sytem may be loaded.
Enter ACCEPT to retain displayed file tables.
Enter file tables n,n,n,... etc and ACCEPT.
Enter NONE and ACCEPT to inhibit all tables.
Enter ALL and ACCEPT to include all tables.

The above prompt is repeated for file tables available from this system, i.e. to restrict the user from moving from the existing file table to certain other file tables by using the T command.

10 ("LOAD SYSTEM" OPTION AVAILABLE
("LOAD SYSTEM" OPTION NOT AVAILABLE

Enter ACCEPT to retain option.
Enter REJECT to change option.
This option is to allow/inhibit a system to be replaced in a file table.

MAPS

OPTION 3 DEFINITION PRINT

Step	Prompt	Operator Action
1	Print from?	Enter ACCEPT to print range from start. Enter a system number and ACCEPT.
2	to?	Enter ACCEPT to print range to end. Enter a system number and ACCEPT.
3	O.K.?	Enter ACCEPT to post the print request to assigned queue. Enter REJECT to return to step 1.

MAPS

OPTION 4 ACTIVE SYSTEM ENQUIRY

The following information is displayed for each system currently running on the machine.

File table
System number
System name
Tasks using system.

MAPS

OPTION 5 SYSTEM DELETE

This option allows the user to remove a defined system from disk only.

MAPS

OPTION 6 RELOAD SYSTEM

This option enables a new file system to be loaded on the users' file table.

The option checks that the current file table is not in use elsewhere and that no entries in the Print Queues originated on this system. If either condition is true the operator is advised.

The option will stow the current system control record, replace the current file table by the new system, establish the new file control blocks by referring to the new system catalogue, fetch the new system control record and flash to all screens that a new system exists on a certain file table.

N.B. The use of ONE Control Record only is strongly advised.

The Control Record will be phased out. This is because the Operating System use of this file is no longer required. FSB details, Spool maintenance data and Serial Numbers are now disk based in separate reserved files.

Consequently all Application orientated information should always use specific files as defined by the user/programmer. Therefore on new projects the Control Record must NOT be used and, wherever possible, existing sites should be amended where applicable.

MAPS

OPTION Z DEFINED SYSTEMS

The number and name of each system loaded on disk is displayed on the screen.

MAPS

OPTION 8 INHIBIT CLASS NAMES - AMEND

Step	Prompt	Operator Action
The existing inhibit class names will be displayed.		
1	Switch No.	Enter switch number (range 1-16) and ACCEPT.
The class name will be displayed if present.		
2	a----------a	Enter ACCEPT to retain displayed name. Enter name (max. 14 chars) and ACCEPT. Enter DELETE and ACCEPT to remove switch name.

MAPS

OPTION 9 SWITCH INHIBIT CLASS

This utility will display the current switches and names and the message INHIBITED if the inhibit flag is set. Entry of a valid switch number will reverse the state of its inhibit flag and redisplay all switches.

MAPS

OPTION 10 PASSWORDS MAINTENANCE

Step	Prompt	Operator Action
1	Password?	Enter Password 1 from Passwords file. Enter ESC to return to menu.
2	Password No.?	Enter required Password number.

The current value of the selected Password, if any, will be displayed.

3	a----------a ?	Enter ACCEPT to leave unchanged. Enter up to 15 alpha-numeric characters and ACCEPT to change the Password. Enter the word DELETE to remove the Password. Enter ESC to go to step 2.

MAPS

OPTION 11 SERIAL NUMBER MAINTENANCE

Step	Prompt	Operator Action
1	Serial no.	Enter required Serial Number number. Enter ESC to return to menu.

The current values for the selected Serial Number are displayed along with dates for its last use and last reset.

2	Reset no.	Enter ACCEPT to leave unchanged. Enter decimal number to which the Serial Number will reset when it reaches the pre-defined maximum. Enter the word DELETE to clear the current values for this Serial number and ACCEPT.
3	Maximum no.	Enter ACCEPT to leave unchanged. Enter decimal number at which the Serial Number will reset and ACCEPT. Enter ESC to go to step 2.
4	Next available no.	Enter ACCEPT to leave unchanged. Enter decimal number to replace the current value of the Serial Number and ACCEPT. Enter ESC to go to step 3.

The selected Serial number will now be updated; Program returns to step 1.

MAPS

OPTION 12 SET PRINT QUEUE DEFINITION

This option is used to define Form size for electronic forms control printers and also the line-up procedures for all printers; a queue definition set up by this option may then be referenced by the command Vn (where n is the queue number) so that stationery alignment may be achieved.

Step	Prompt	Operator Action
1	Queue no.	Enter queue number (1 to max allowed) and ACCEPT.
2	Code for characters per inch(n)	Enter the code (0-7) defining the number of characters per inch (ie. the pitch) and ACCEPT.
3	Code for lines per inch(n)	Enter the code (0-3) defining the number of lines per inch and ACCEPT.
4	Lines per page (nnn)	Enter the length of form to be used and ACCEPT (default is 66 lines).
5	Next VT @ line(nnn)	Enter the line number at which the next vertical tab is situated and ACCEPT.

Return to Step 5 until zero is entered or a maximum of 10 Tabs have been set.

The above has now defined the form layout; below defines the line-up procedures.

6	Line nnn - Prints @(01/1600.-)	The line number displayed will denote the line at which the print head is currently situated. Enter the start address for printing X's and ACCEPT.
7	No. of X's(nnn)	Enter the number of X's to print on this line and ACCEPT.
8	F-orm feed, V-ertical tab or no. of line feeds(xxx)	Enter the type of paper throw required and ACCEPT.

F for form feed; used to terminate the definition.

V for vertical tab; see notes on device types.

nn number of line feeds; 1 denotes print on current line

Return to step 6 until either form feed is entered or a maximum of 16 entries.

NOTES (see Appendix for full list of available options)

a) Characters per inch.

The DRE 8840 allows 5 character sizes as follows (any printers used in the future will use the same codes as far as is possible).

Code	Size (inches).
0	10 (default)
1	12
2	13.3
3	15
4	17.1

b) Lines per inch

The DRE 8840 allows 6 and 8 lines per inch as follows:

Code	Size
0	6 (default)
1	8

The Lear Seigler Ballistic printer also allows 6 and 8 lines per inch but the selection method is different in that one can only 'swap' states.

Code	
0	Leave unchanged.
1	Swap L/P/I State

This serves as an example regarding compatability problems; as far as possible the System will give compatability but the ultimate responsibility is with the Application design and implementation.

c) Vertical Tabs

Historically, with the DRI paper tape format, a Tab set on line n meant that when issuing a Tab instruction on any line 1 to n-1, the next print would occur on line n. The DRE 8040, when programmed with a Tab on line 1 to n, is defined such that a Tab instruction issued on any line 1 to n, prints next on line n+1; again this serves to highlight the problems of compatability and with reference to the above utility, means that although the concept is to allow device independance, the operator may still need to have detailed information about individual devices.

d) File Requirements

A direct access file is required by this utility (File I/D 100) containing 1 to n records (where n is the maximum number of definitions required) each of 64 words.

MAPS

OPTION 13 DELETE PRINT QUEUE DEFINITION

This option allows the removal of a previously created print layout definition. See Option 12.

MAPS

OPTION 14 DUPLICATE SYSTEM DEFINITION

This option allows a previously defined System to be duplicated under a different System number, with a different System name. The original System is unchanged.

Step	Prompt	Operator Action
1	Source System Number?	Enter required System number and ACCEPT.

The source system must be a previously loaded system. The System name is displayed for sight checking.

2	Target System Number?	Enter required System number and ACCEPT.

The target system must NOT be loaded.

3	New System Name?	Enter the required name (max. 16 chars.) and ACCEPT.
4	O.K.?	Enter ACCEPT to proceed. Enter REJECT to go to step 1. Enter ESC to return to the menu.

OP - ONLINE PROGRAM AMENDMENT

Interrogates and amends the contents of any part of memory. Operations are usually performed in octal. When in octal mode, the program attempts to decode machine instructions. Facilities are included to read overlay modules into memory and to write the amended module back to the overlay library. A 'breakpoint' handler is included; up to 20 breakpoint intercepts may be inserted into a program under test in another task partition.

OP differs from other utilities in that many non-octal characters input at the keyboard act as an (ENTER) key. Octal characters include "/" which may be used to enter an address in "pp/ccss" format (a 0 or 1 must follow "/").

N.B.
When giving instructions by telephone it is advisable to warn an unaccustomed operator not to press the (ENTER) key.

Alternative "load" keys to (cursor down) are (ENTER+) or (GS), (ENTER-) or (FS) acts as a 'load' key which additionally sets bit 17 of the word being loaded (useful for loading an offset address), whilst "." immediately preceding a 'load' key sets bit 16 of the word (useful for loading an odd byte address). The address being loaded is always the current value of the "memory Address", which is set to the task partition base upon entry of OP and increments by one each time a word is loaded.

The program will display a string giving the current value of the memory address in absolute address pp/ccss b format and will attempt to decode the contents in the following manner:

```
41/0001 1 = 047456, 23/1456 0, INSZ IZ 1456
<-----> |   <----> <-------> <--Instruction decode-->
   |    |     |          | - Address decode
   |    |     | - Octal decode
   |    | - Current memory bank no.
   | - Current memory address
```

This display can be further extended to decode the value as a single word decimal number, as two ASCII characters and as three Metacode characters. See the "C" command below.

COMMANDS

Note
The following abbreviations are used below:
a) 777 = any octal number and/or address.
b) 999 = any decimal number, usually to a maximum of 65535.
c) 99 = a decimal number with a defined limit.
d) 77 = an octal number with a defined limit.
e) If the 'number' is in brackets, then it is optional.

1 777R
Program checks that the Breakpoint Handler is not active then reads the Overlay Module into memory and sets memory address to point to the first word of the current Task partition.

2 W
Program checks that a module was first read from disk. If so, the module is written back to disk and its new "hash total" inserted into the Overlay Index.

3 (777)Enter+, LF, GS
Program loads the value at the current memory address, re-displays the current memory address, increments the memory address and then displays the new contents.

4 L
Each character entered at the keyboard is loaded into the next byte until the (ESCAPE) key is pressed. The memory address is incremented by one after every second character. N.B. (cursor left) does not correct a typing error, but loads ASCII code "BS" into the current byte.

5 M
Characters input are converted to Metacode. For every three characters entered the generated Metacode word is loaded into memory and the memory address incremented. Entry of ESC terminates Metacode mode.

6 SPACE
From the memory address pointer, the contents of memory are displayed as ASCII until a NUL is encountered. The memory address remains constant.

7 *
From the memory address pointer, the contents of memory are displayed as Metacode until a Metacode NUL is encountered. The memory address remains constant.

8 C
The display type is reversed. i.e. a long display replaces a short display and vice versa. It should be noted that this command works independently from the "C" command in the Breakpoint Handler (see below).

9 B
The Breakpoint Handler is either loaded and entered or re-entered if previously loaded.

10 %
The current memory contents are decoded and displayed in date format.

11 ESC
Program checks if the Breakpoint Handler is loaded and if not exits to PROGRAM?. Otherwise if no Breakpoints are currently set or active (see Breakpoint Handler below) then OP will remove the Breakpoint Handler and allow normal OP commands.

12 ETX
Acknowledges an ERROR condition.

13 (777)Enter- or FS
Enters the octal value into the current memory location after automatically appending Bit 17.

14 777A
Sets the memory address to the address input, current bank.

15 (777)O
Sets the memory address to the address input PLUS the current base, current bank (see "P" command).

16 (777)I
Sets the memory address to the contents of the location input. If the value obtained has Bit 17 set, then the current base is added to it.

17 (777)G
Sets the memory address to the contents of the location addressed by the octal input PLUS the current base. If the value obtained has Bit 17 set, then the current base is added to it.

18 (77)P
Sets the current base to the input value. If the input value is zero, then the current Partition base is used. It should be noted that this command works independently of the "P" command in the Breakpoint Handler (see below).

19 (9)@
Sets the current memory bank number to the input value. If the input value is zero, then the current Partition's bank is used. It should be noted that this command works independently to the "@" command in the Breakpoint Handler (see below).

20 (999)S
Enters a single word decimal value at the current memory address, re-displays the current memory address, increments the memory address and then displays the new contents.

21 (999)D
Enters a double word decimal value at the current memory address, re-displays the current memory address, adds two to the memory address and then displays the new contents.

22 (999)T
Enters a treble word decimal value at the current memory address, re-displays the current memory address, adds three to the memory address and then displays the new contents.

23 (999)Cursor Right or Control-U
The memory address is incremented and the new contents displayed. This operation is repeated for the input number of steps. At any point, entry of ESC will stop the display at the current step; entry of SPACE will stop/start the display.

24 (999)Cursor Left or Control-H
The memory address is decremented and the new contents displayed. This operation is repeated for the input number of steps. At any point, entry of ESC will stop the display at the current step; entry of SPACE will stop/start the display.

25 (999)Control-Y
The current memory address contents are displayed. This operation is repeated for the input number of times. At any point, entry of ESC will stop the display; entry of SPACE will stop/start the display.

26 (777)F
The octal input is interpreted as the control word for an 'Extract from File Control Block', e.g. 001411F will display the address and contents of word 3 of the FCB of file 011 (usually the Customer file), i.e. the minimum record number. This command must be used with care as no checking of the input is performed.

27 Command or Control- V
Allows entry to the Command processor.

BREAKPOINT HANDLER

When entering the Breakpoint Handler using the "B" command in OP the following is requested:

Step	Prompt	Operator Action
1	Target Task number?	Enter Octal Task number running the program under test and ACCEPT. Enter ESC to return to OP.

The Program now responds with the prompt "??". Entry of ESC now will simply allow re-entry of OP but retain the Breakpoint Handler. The user can now freely move between the two programs by using ESC when in the Breakpoint Handler and the "B" command when in OP.

The Breakpoint Handler offers the facility to control, and dynamically amend the execution of another program running in another Task partition.

This is accomplished by placing within the target program, in memory, 'Breakpoint intercepts'. Eventually the target program will 'hit' a Breakpoint upon which the Handler will suspend the target program and display relevant information to the user as follows:

```
                                        <State of Processor flags>
Task 04 Breakpoint at:-                  (NOT G/THAN, NOT CARRY)
41/0034 1 = 037634, 17/1634 0, JSBR IZ 1634                    ,
 <------> |   <----> <--------> <--Instruction decode-->
    |     |     |          | - Address decode
    |     |     | - Octal decode
    |     | - Current memory bank no.
    | - Current memory address

 (line continues:)                    A= 234561, B= 033642
                                      <--------> <-------->
                                       Values of A and B
                                      registers respectively
```

This display can be expanded to give more detailed information; see "C" command below.

The user may now allow the target program to proceed from the current position or eslewhere. By moving to OP other areas of the program, such as dynamic data or program code may be examined and amended.

It should be noted that certain areas of 'volatile memory' are also retained by the Handler and re-instated when the currently active Breakpoint is removed. These areas are:

a) Zero Page 00/0045 (No. characters input)
b) Zero Page 00/0140 to 00/0177 (Transient workspace)

Also the Handler allows the user to Breakpoint through a Fetch & Lock followed by a Rewrite/Overwrite effectively without breaking the 'Lock', though it is the user's responsibilty to ascertain the validity of the relevant transfer.

COMMANDS

Note
The following abbreviations are used below:
a) 777 = any octal number and/or address.
b) 999 = any decimal number, usually to a maximum of 65535.
c) 99 = a decimal number with a defined limit.
d) 77 = an octal number with a defined limit.
e) If the 'number' is in brackets, then it is optional.

1 777A
Sets a Breakpoint at the input address, current bank.

2 7770
Sets a Breakpoint at the input address PLUS the current base, current bank.

3 (77)P
Sets the current base to the input value. If the input value is zero, then the Breakpointed partition base is used. It should be noted that the offset base maintained by the Breakpoint Handler is different to that used by OP.

4 (99)R
Allows the removal of the previously set Breakpoint number 99. If no Breakpoint number is input, then the removal of all Breakpoints is assumed.

5 ESC
Allows the user to move to program OP. If there is a currently active Breakpoint, OP will be entered with the memory address set to the current Breakpoint position.

6 Enter+ or LF or SPACE
Indicates the restart of the currently active Breakpoint. The user is prompted for the Restart address and the values of the A-register, B-register, Carry flag and Greater Than flag. Enter + to the above prompts will use the values recorded when the Breakpoint was intercepted.

7 *
The Handler will re-display the currently active Breakpoint or the last active Breakpoint and or the current memory base and bank number.

8 (99)S
The currently active Breakpoint may be automatically restarted and a new Breakpoint inserted an input number steps in advance of the current Breakpoint address. Input of zero defaults to the next step. Care must be taken as Breakpoints should only be inserted in EXECUTABLE steps. i.e. Breakpoints should not be set within parameter blocks, data areas etc.

9 (99)D
Breakpoint number 99 will be displayed, giving the memory address at which the intercept was placed. Input of zero will display all Breakpoints.

10 (99)L
This command allows the target program to pass through, or Loop on the currently active Breakpoint an input number of times. A zero input defaults to a loop of one. This command works by automatically setting a Breakpoint at the step after the currently active Breakpoint, then restarts the Breakpoint. A new Breakpoint is then set at the original step and the intermediate Breakpoint restarted. If the step following the active Breakpoint is not excecutable code, then by using the Loop Offset variable the intermediate Breakpoint can be placed successfully.

11 (777)M
Sometimes the target program fails at some point within an overlay called dynamically and if the Fetch & Link facility is used then a standard Breakpoint cannot be placed. Using this command allows the Handler to intercept the nominated overlay as it is loaded into memory, producing a standard Breakpoint at the first step of the target overlay. The input intercept overlay number is active until overidden or removed. This facility can be used to trap a Print Program as the OS loads it into memory. Care must be taken in this instance as the Breakpoint produced is within the OS. The user must Step the Program through to the actual applications entry point.(see command "S")

12 C
Allows the Breakpoint display to be changed from the long display to the short and vice versa. It should be noted that this command acts independently of the "C" command in OP.

13 ETX
Acknowledges an ERROR condition.

14 (9)@
Allows the current memory bank number to be changed. It should be noted that this command acts independently of the "@" command in OP.

15 (99)J

When the currently active Breakpoint consists of a JUMP or JSBR instruction this command will set a Breakpoint at the target address and restart the current Breakpoint.

For example:

```
JUMP    1524        (The current Breakpoint consists
                    (of this instruction.
```

Entry of "J" will cause a new Breakpoint at current page step 1524.
Entry of "99J" will cause a new Breakpoint at current page step 1524(octal) + 99.

Alternatively:

```
JSBR    1524
```

Entry of "J" will cause a new Breakpoint at current page step 1525 - i.e. the first executable step.

JUMPs or JSBRs via indirect addresses will produce the same results.

N.B.
The above command will also attempt to process instructions involving references to Zero Page. i.e. JSBR IZ 1733. As other programs, including the OS use these subroutines attempting to use the "J" command will invariably result in a complete machine failure forcing the system to be re-bootstrapped. DO NOT therefore use this command except where the references are within the bounds of the application.

Two special cases exist:

a) JSBR/JUMP IZ A
b) JSBR/JUMP IZ B

These instructions will perform correctly.

16 H

Allows a previously Halted Task to be restarted. The restart address will be requested.

OP/BREAKPOINT HANDLER COMMAND SUMMARY

COMMANDS		FUNCTION	
OP	B/PT	OP	BREAKPOINT HANDLER
777A	777A	Examine abs. loc .	Set B/pt @ abs. loc.
(777)O	(777)O	Examine offset loc.	Set B/pt @ offset
(777)I	---	Examine via abs. loc.	
(777)G	---	Examine via offset	
(77)P	(77)P	Change Base	Change Base
(777)R	(99)R	Read Program Overlay	Remove B/pt(s)
ESC	ESC	Escape to PROGRAM?	Escape to OP
SPACE	SPACE	Display ASCII	Restart B/pt
*	*	Display Metacode	Re-display B/pt
(999)S	(999)S	Decimal input-Single	Step/Skip B/pt
(999)D	(99)D	" " " -Double	Dispaly B/pt(s)
(999)T	---	" " " -Triple	
(999)cU	---	Examine next loc(s).	
(999)cY	---	Re-examine loc.	
(999)cH	---	Examine previous loc	
(777)LF	LF	Enter Value	Restart B/pt
(777)E+	E+	" "	" "
W	---	Write Program O'lay	
L	(99)L	ASCII input	Loop B/pt
M	(777)M	Metacode input	Module Intercept
C	C	Switch Display	Switch Display
B	---	Enter B/PT Handler	
%	---	Display as Date	
(777)F	---	Examine F.C.B.	
ETX	ETX	Cancel ERROR	Cancel ERROR
CMND	---	Enter Command Mode	
cV	---	" " " "	
E-	---	Enter Value + B17	
(9)@	(9)@	Change memory bank	Change memory bank
---	(99)J		B/pt thru JUMP/JSBR
---	H		Restart from HALT

POMM - PROGRAM & OVERLAY MODULE MAINTENANCE

This program provides all the facilities for creating Programs and maintaining Program Libraries.

The following menu is displayed on entry and on completion of any option.

1 Module Library Maintenance
2 Input Directory Maintenance
3 Print Directory Maintenance
4 Input Directory Security Override
5 Delete Modules
6 Module Listing
7 Module Library Compression Request
8 Transfer Modules
9 Modify Module
10 Scan Modules
11 Program & Overlay Module Map
12 Search Modules
13 Unused Module Numbers
14 Transfer Library

Step	Prompt	Operator Action
1	Option?	Enter option number and ACCEPT. Enter ESC to return to PROGRAM?.

POMM

OPTION 1 MODULE LIBRARY MAINTENANCE

This option enables the programmer to create new overlay module areas on disk by establishing parameters relating to size, disk location and relative location within a 2K partition within the Overlay Library Module Index. It also allows the programmer to define in which module area a program may be found and to establish the entry point within that module for the program.

Step	Prompt	Operator Action
1	Module no.	Enter the octal module number and ACCEPT. Enter ESC to return to the menu.

If the requested Module already exists the current load location, hash total, length etc. will be displayed; go to step 9.

2	No. of sectors	Enter the length required, from 1 to 13 and ACCEPT.
3	Load location	Enter the address at which the module is to be loaded & ACCEPT. This is normally a relative address within a 2K partition ie. an Offset Address.

The program will force an entry in either the Print or Input directory to give an indication, on a Program Library listing of what the Overlay is for.

4	I/O Station or Print module? (I/P)	Enter "I" or "P" to indicate in which directory the entry should appear and ACCEPT.
5	Input directory name) Print directory name)	Enter up to 4 ASCII chracters and ACCEPT.

If the new module is to be used as an overlay, no entry point is required.

6	Is this module an overlay?	Enter ACCEPT if no entry point required; go to step 8. Enter REJECT if an entry point is required.

7	Entry location	Enter the program start point and ACCEPT.
8	The new module is added to the library (with all sectors clear) in a 'free' area of the Overlay Library. Return to step 1.	
9	Load location	Enter ACCEPT to leave unchanged. Enter a revised load location and ACCEPT. Enter "DELETE" and ACCEPT to remove the module from the library.

If a delete is requested then both the directories are searched for any program names relating to this overlay. If any entries are found they are displayed.

10	Proceed?	Enter ACCEPT to delete the module, but NOT the names. Enter REJECT to go to step 1.

FOMM

OPTION 2 INPUT DIRECTORY MAINTENANCE

Step	Prompt	Operator Action
1	Input directory name	Enter up to 4 characters and ACCEPT. Enter ESC to return to menu.

If the name already exists the module number and the current entry location will be displayed.

2	Module no.	Enter ACCEPT to leave unchanged. Enter the word DELETE to remove the name from the directory. Enter the octal module number required and ACCEPT.

If the module is to be used as an overlay, no entry point is required.

3	Is this module an overlay?	Enter ACCEPT if no entry point required; go to step 5. Enter REJECT if an entry point is required.
4	Entry location	Enter ACCEPT to leave unchanged. Enter the program start point and ACCEPT.
5	The name is either loaded or amended; go to step 1.	

POMM

OPTION 3 PRINT DIRECTORY MAINTENANCE

This Option operates in exactly the same way as Option 2 except it maintains the Print program directory.

FOMM

OPTION 4 INPUT DIRECTORY SECURITY OVERRIDE

When a disk Security is in progress, entry to programs is not allowed unless that program has previously been given a Security Overide.

This Option should be used sparingly, allowing for example only enquiry type programs to run.

Step	Prompt	Operator Action
1	Input directory name	Enter up to 4 characters and ACCEPT.

The module number and entry location for the entered name are displayed.

2	Override allowed? Override not allowed?	Enter ACCEPT to retain current state. Enter REJECT to change state.

The directory element is updated; go to step 1.

POMM

OPTION 5 DELETE A RANGE OF MODULES & ASSOCIATED DIRECTORY ELEMENTS

Step	Prompt	Operator Action
1	Start module no.	Enter octal module number at which deletion is to begin and ACCEPT.
2	End module no.	Enter module number afterwhich deletion will stop.
3	Proceed?	Enter ACCEPT to start the deletions. Enter REJECT to go to step 1.

Each entry found in the Overlay index file is cleared and any associated names in either the input or print directories are displayed and then removed. Go to step 1.

FOMM

OPTION 6 LIST MODULE/MEMORY

Step	Prompt	Operator Action
1	List Module?	Enter ACCEPT to list an overlay module; go to step 4. Enter REJECT to list from the machine memory.
2	From?	Enter the start memory address and ACCEPT.
3	To?	Enter the memory address afterwhich the list is to stop and ACCEPT; go to 5.
4	Module Number?	Enter octal module number and ACCEPT.
5	Title?	Enter up to 33 characters and ACCEPT.
6	O.K.?	Enter ACCEPT proceed. Enter REJECT to go to step 1.

The required print request is posted to the currently assigned queue; program returns to step 1.

FOMM

OPTION 7 MODULE LIBRARY COMPRESSION REQUEST

A request is posted to the currently assigned queue to compress the current master program library. This process releases areas within the Overlay library for re-use.

POMM

OPTION 8 TRANSFER MODULES

This Option allows the transfer of a module or range of modules within a library or from one library to another. The source and target libraries may reside on different system disks, provided the Catalogue, Systems Tables file, FSB file and actual program files are available from both systems. This in effect means that all libraries within a system should reside on the same disk with the above files. The program also requires valid Configuration Table(s) to gain access to the above files.

Step	Prompt	Operator Action
1	Source Disk: nnn?	Enter ACCEPT to leave unchanged. Enter the octal Disk number which contains a Configuration Table, Catalogue and Systems Tables file and ACCEPT. Enter ESC to return to the menu.

The program will extract the required information. Any irregularities found will be reported and the program will not continue. The information gathered may refer to a different disk number from that specified. For example the input disk number may be that of a security disk. The definitions of the above files will then refer to a different disk (in reality the master disk from which the security was taken). Therefore, any variations in disk number found by the program will require verification by the operator.

2	Source System Number: 999 ?	Enter ACCEPT to leave unchanged. Enter the System numbe r which defines as its master library the required library and ACCEPT.

The selected System name will be displayed for verification.

The above procedure is repeated for the target file details.

3	Source Start Module Number?	Enter the octal number from which the transfer is to start and ACCEPT.
4	End Module?	Enter the module number after-which the transfer is to stop and ACCEPT.
5	Target Start Module?	Enter the module number to which the transfer is to start and ACCEPT.

N.B.
It should be noted that when transferring a range of modules, the non-existent modules are included in the transfer. For example if the range of modules 1 to 3 are transferred to the range starting at 11, then:

module	1	goes to module		11)	This is true even if
""	2	""	""	12)	module number 2
""	3	""	""	13)	does not exist.

6 Transfer Directory Element(s)?
Enter ACCEPT to transfer any program names.
Enter REJECT to transfer only the modules.

7 O.K.?
Enter ACCEPT to start the transfer.
Enter REJECT to go to step 3.

A display is produced showing for each actual module transferred the target module number, the character "O" if the overlay existed before the transfer and the character "D" if the directory element existed before the transfer.

The program returns to step 3.

POMM

OPTION 9 MODIFY MODULE

This Option allows a program module to be either extended or truncated in size, or a section from another module to be 'merged' into the module.

Step	Prompt	Operator Action
1	Module Number?	Enter the octal module number and ACCEPT. Enter ESC to return to the menu.
	The load location and length are displayed for checking.	
2	Extend/Truncate?	Enter ACCEPT to change the module size; go to step 7. Enter REJECT to merge from another module.
3	Merge from Module?	Enter octal module number from which the merge is to take place and ACCEPT.
4	Source Start Address?	Enter the offset address from which the data is to be taken and ACCEPT.
5	Length(words) 999 ?	Enter ACCEPT to leave unchanged. Enter the number of words to be moved and ACCEPT.
6	Target Start Address?	Enter the offset address at which the data is to be moved and and ACCEPT; go to step 10.
7	Revised length?	Enter the new module length and ACCEPT.

If the revised length is less than the current length go to step 9.

The program will only request how the module is to be extended if there is a choice.

8	Extend at end?	Enter ACCEPT to add the required length to the end of the overlay. Enter REJECT to add the required length to the start of the module; go to step 10.

9	Truncate at end?	Enter ACCEPT to remove from the end of the module. Enter REJECT to remove from the start of the module.
10	O.K.?	Enter ACCEPT to perform the required change. Enter REJECT to go to step 2.

The relevant function is performed; program goes to step 1.

POMM

OPTION 10 SCAN MODULES

Step	Prompt	Operator Action
1	START MODULE?	Enter ACCEPT to scan from start. Enter octal module number at which the scan is to start and ACCEPT.
2	END MODULE?	Enter ACCEPT to scan to end. Enter the end module number and ACCEPT.
3	O.K.?	Enter ACCEPT to perform the scan. Enter REJECT to go to step 1.

The program reads all overlays and performs a hash check, reporting any failures. Program returns to step 1.

POMM

OPTION 11 PRINT PROGRAM MAP

Produces a printout of the structure of the user's overlay module library and program directories as generated by the POMM utility. The printout can be in directory name or module number order for the whole file or a selected range.

For each module selected the module number, relative start sector within the library, size, memory load location and software hash total will be printed. For each directory name selected the name, entry location and directory type will be printed.

Step	Prompt	Operator Action
1	Required System no.(99)	Enter ACCEPT to print from the current master library. Enter required System number and ACCEPT.
	The System name is displayed for sight checking.	
2	Alphabetic or Numeric Sequence. Enter 'A' or 'N'	Enter A and ACCEPT for directory name order. Enter N and ACCEPT for module number order.
3	Print a range?	Enter ACCEPT for a range and continue at step 4. Enter REJECT for the whole file and continue at step 6.
4	Start	Enter first module number or directory name and ACCEPT.
5	End	Enter last module number or directory name and ACCEPT.
6	Process?	Enter REJECT to re-enter details. Enter ACCEPT to request the print and return to the step 1.

FOMM

OPTION 12 SEARCH MODULE/MEMORY

Step	Prompt	Operator Action
1	Search Type: Ascii String, Word, Mandatory Bits, Optional Bits Search word nn - Type	Enter A, W, M or O for required type of search for the displayed word or words. Enter ACCEPT if search details are complete, continue at step 7.

If an Ascii search is required then this type may only be selected at word 1 and no further search types will be allowed. Continue at step 2.

All other search types may be used to set up a block of upto 10 words of varying types.

If a Word search is required continue at step 3.

If a Mandatory bit search is required the routine will check that all the selected bits are present with no other bits in each compared word. An optional bit search may also be performed on the same word if required. The optional bit search will check that there are no other bits present other than those declared by the mandatory bits and any of those specified as optional. Continue at step 4.

If an Optional bit search is required the routine will check that no bits are present outside the requested optional bits but that any of the optional bits are present. Continue at step 6.

2	Ascii String	Enter up to 20 characters and ACCEPT, continue at step 7.
3	Octal Code	Enter an octal value and ACCEPT. The program will attempt to decode this as an instruction before returning to step 1.
4	Mandatory Bits(Octal)	Enter the required bit(s) in octal format and ACCEPT.

Step	Prompt	Operator Action
5	Optional Bits(Octal)	Enter the required bit(s) in octal format and ACCEPT. Enter ACCEPT to ignore optional bits. The program will check that the same bits were not chosen in Options 4 and 5. Continue at step 1.
6	Optional Bits(Octal)	Enter the required bit(s) in octal format and ACCEPT. Return to step 1.
7	Search Medium, M-emory or D-isk based modules	Enter "M" and ACCEPT for a memory search; go to step 8. Enter "D" and ACCEPT for a disk search; go to step 10. ACCEPT to default to Disk.
8	Start memory address	Enter search start location and ACCEPT.
9	End memory location	Enter last search location and continue at step 12.
10	Start Module No.	Enter octal start module number and ACCEPT. Enter ACCEPT to search from the beginning of the module library.
11	End Module No.	Enter the last octal module number to be searched and ACCEPT. Enter ACCEPT to search to the end of the module library.
12	Display/Print? Enter: ACCEPT to Display only D to Display & Print P to Print only	Enter the above required input.
13	Proceed?	Enter ACCEPT to begin the search. Enter REJECT to return to step 1.

The display may be stopped and restarted by use of the SPACE bar. ESC will terminate the search immediately and return to step 1.

POMM

OPTION 13 UNALLOCATED MODULE NUMBERS

This Option simply lists all module numbers within the current master library which are not in use.

POMM

OPTION 14 TRANSFER LIBRARY

This Option allows the transfer of libraries between systems. The source and target libraries may reside on different system disks, provided the Catalogue, Systems Tables file, FSB file and actual program files are available from both systems. This in effect means that all libraries within a system should reside on the same disk with the above files. The program also requires valid Configuration Table(s) to gain access to the above files.

Step	Prompt	Operator Action
1	Target Disk: nnn?	Enter ACCEPT to leave unchanged. Enter the octal Disk number which contains a Configuration Table, Catalogue and Systems Tables file and ACCEPT. Enter ESC to return to the menu.

The program will extract the required information. Any irregularities found will be reported and the program will not continue. The information gathered may refer to a different disk number from that specified. For example the input disk number may be that of a security disk. The definitions of the above files will then refer to a different disk (in reality the master disk from which the security was taken). Therefore, any variations in disk number found by the program will require verification by the operator.

2	Target System Number: 999 ?	Enter ACCEPT to leave unchanged. Enter the System number which defines as its master library the required library and ACCEPT.

The selected System name will be displayed for verification.

The above procedure is repeated for the source file details.

The source and target file lengths are compared and any differences reported to the operator.

3	O.K.?	Enter ACCEPT to proceed with the transfer. Enter REJECT to go to step 1.

The relevant program files are updated. Program returns to step 1.

Q - SPOOL QUEUE ENQUIRY

This program allows the interrogation of the Spool File by queue number and/or by Program name. The current Spool usage is calculated, being displayed as a percentage, along with a list of all active queues.

Step	Prompt	Operator Action
1	POSTED PROGRAM NAME?	Enter the required Print program name for interrogation. Enter ACCEPT for a display of all Postings.
2	SEARCH QUEUE?	Enter the Print queue number for a specific search. Enter ACCEPT for all queues.

The following will be displayed:

999 XXXX (999) tn

This consists of the Queue Number followed by the Posted Program name and a count of consecutive postings of this Program. The originating File Table number is given for sites where the spool is shared between File tables.

R - RESTART TASK

This utility may be used after a TASK HALTED or TASK AWAITS DISK message has been issued by the System; it enables programmers to restart a program under test at any point.

Step	Prompt	Operator Action
1	Password?	Enter Password 3 from the Passwords file and ACCEPT.
2	Restart Task no.?	Enter the number of the task to be restarted in octal and ACCEPT.
3	Restart Address?	Enter the memory address (absolute or offset) at which the task is to be restarted and ACCEPT. Enter ACCEPT only to restart at the default address (see SPECIFY DEFAULT RESTART ADDRESS)
4	O.K.?	Enter ACCEPT only to continue. Enter REJECT only to return to Step 2.

The task, if it was halted or disk queued, will now restart at the indicated address.

RECY - DISK RECOVERY UTILITY

After the system has been bootstrapped and the system disk successfully recovered it is possible to recover each data disk by simply reversing the security procedure. Each security copy will remember how it was last secured and this utility will attempt to re-instate the master disk in use at that time.

The user/programmer must ensure that the destination master disk has been labelled at least with the disk number before entering the copy disk number of the disk to be recovered. The utility will display the recovery procedure it is about to perform for confirmation.

Please note that this utility should be run for every copy disk in the current sequence before the system can be deemed totally recovered.

All master disk labels will be correctly re-instated ready to copy to the next security disk in the sequence after the disk from which the master was recovered.

The operator simply enters the SECURITY Disk number to be recovered next.
The parameters required by the recovery sequence will be automatically located, by the program, from the nominated disk.

RPT - REPRINT

This utility is available to allow documents to be reprinted provided the originating programs satisfy certain elementary conditions. Each printer recognised within the system will have its own reprint queue and re-printable work processed by the printer will be placed into that reprint queue regardless of the queue of origin of that work.

Step	Prompt	Operator Action
1	PRINTER A?	This prompt only appears if more than 1 printer is available to the System. Enter the printer identification letter A, B, C etc. and ACCEPT. Enter ESC to return to PROGRAM?
2	PRINT Q nn	None; nn is the reprint queue of the selected printer.
3	SERIAL NO	Enter the serial number of the document to be reprinted and ACCEPT. If more than 1 document is to be reprinted then enter the first serial number. Enter ACCEPT only if this is an enquiry on the contents of the reprint queue. Enter ESC to return to PROGRAM?
4	The utility will search the reprint queue for a document with the specified serial number. As the search proceeds, the VDU will display the 4 character program name and the serial number of each item in the queue. If the serial number is found then continue at step 5 otherwise return to step 3.	
5	REPRINT?	Enter ACCEPT only to request the document with this serial number to be reprinted. Enter REJECT only if this document is not to be reprinted. Enter ESC to return to PROGRAM?
6	Next Item in Queue	None; the utility will return to step 5 unless the end of the queue has been reached whereupon a return is made to step 1.

If ACCEPT only was entered at step 5 for any item, the utility places a marker at the current end of the reprint queue. A printer commanded to process the reprint queue examines each item in the queue in chronological order. Items with reprints pending will not only be reprinted but will be re-posted to the end of the same reprint queue in case a further reprint is required. Items with reprints not pending will be dropped and it is now too late to request a reprint for those. When the marker placed by RPT is encountered in the queue, the printer switches itself onto queue 0 and becomes IDLE, thus allowing the Operator the opportunity to request further reprints.

S - SEND MESSAGE

Step	Prompt	Operator Action
1	Target I/O Station Task no.?	Enter the task number of the destination.in octal and ACCEPT. Enter ACCEPT only if the message is to be sent to all I/O Stations Enter ESC to return to PROGRAM?
2	Message :	Enter the message to be displayed and ACCEPT. The message will be displayed (with BEL tone) at the required destination and the routine returns to step 1.

SCAM - SYSTEM CATALOGUE MAINTENANCE

The System Catalogue

The System Catalogue is used to map out the available backing storage, thus allowing easy maintenance of disk-based data, by means of defining the basic layout and usage of the 'Data-Sets'. These Data Sets may be used to define Files (see later).
Associated with each Data Set Definition and included on the appropriate catalogue record will be a 12 character name.

The System Tables File

File accessing is always done via an octal file identifier; that identifier is then used to access a pointer to the details for that file, known as the File Control Block (FCB). All FCB pointers are in fact held as a 'File Table' within the System Table file, and each pointer is simply the Catalogue Record Number which holds the FCB details for that file identifier. The initiator will, upon boot-strapping, obtain the FCB details of all files from the Catalogue and establish them in free memory areas.

File identifiers are associated with the correct FCB details by use of a Programmer Utility known as MAPS i.e. Maintain Application System(s). Within this utility it is possible to set-up/amend the file table so that any slot reserved for a file identifier may have inserted into it the catalogue record number containing the FCB details. This is done by simply entering the file identifier and the Data-Set name with which it is to be associated. A search of the System Catalogue will then take place until a catalogue record is found containing the entered name. The number of that record within the catalogue is then inserted into the slot reserved for that file identifier within the file table.

When the system is bootstrapped, one of the functions performed by the Initiator when 'moulding' the Operating System is the transfer of the file table of FCB pointers from the System table file to a location in free memory. At this point all the catalogue records specified within the file table are read into various free memory areas. The memory address of each Catalogue Record (or FCB) is then inserted into the file table replacing the initial catalogue record number. Reference to any file via a file identifier will then enable the address of the FCB details for that file to be accessed. With the FCB details located, any part of the file may be accessed.

Catalogue maintenance does not effect the current system running, only the catalogue file on disk. In order to include any changes to the catalogue the system must be bootstrapped. Due to the powerful nature of this program, access to it should be carefully controlled on customer sites.

The following menu is displayed at the start of the utility and on the completion of any option.

```
 1    Disk Definition
 2    Data Set Definition
 3    Indexed File Definition
 4    File Table Extensions Definition
 5    Re-name Catalogue entry
 6    Re-number Disk extent
 7    Request Catalogue re-organisation
 8    Compress Disk extent
 9    Print Catalogue entries
10    Display Disk layout
11    Copy Data Sets
12    Extended Direct Access Definition
13    Delete Unused Catalogue Entries
```

Step	Prompt	Operator Action
1	Option?	Enter option number and ACCEPT. Enter ESC to return to PROGRAM?.

SCAM

OPTION 1 DISK DEFINITIONS

Step	Prompt	Operator Action
1	DISK NUMBER	Enter disk number (range 1-277) and ACCEPT.
2	DISK TYPE DD1600, 12856 SECTORS?	Enter ACCEPT to default to displayed type. Enter disk type and ACCEPT. Enter DELETE and ACCEPT to remove the selected disk from the catalogue.
3	n SECTORS UNUSED, n SECTORS IN LARGEST BLOCK PROCEED?	Enter ACCEPT to update catalogue and return to menu. Enter REJECT to return to menu.

Current Disk types are DD1600, DD9600, D8000, D818.

SCAM

OPTION 2 DATA-SET DEFINITION

Step	Prompt	Operator Action
1	DATA-SET NAME?	Enter data-set name (max 12 chars) and ACCEPT.

(Entry of less than 12 characters will initiate an alpha-match search of the Catalogue index. Matching keys are displayed, the operator may choose from these or elect to use the actual input key.)

If the entry exists on the catalogue the details will be displayed.

2	DATA-SET TYPE: DA?	Enter ACCEPT to default to direct access. Enter OL and ACCEPT for an overlay library definition. Enter PD and ACCEPT for a program directory definition.

If a program directory or overlay library type definition was selected go to step 18.

3	MINIMUM RECORD NUMBER: n?	Enter ACCEPT to default to displayed minimum (n). Enter minimum (range 1-65535) and ACCEPT. Enter DELETE and ACCEPT to remove catalogue entry and go to step 17.
4	MAXIMUM RECORD NUMBER: n?	Enter ACCEPT to default to displayed maximum (n). Enter maximum (range min-65535) and ACCEPT.
5	RECORD LENGTH: n?	Enter ACCEPT to default to displayed size (n). Enter size (range 1-999) and ACCEPT.
6	(x SHARED BUFFERS AVAILABLE (NO SHARED BUFFER REQUIRED (USE SHARED BUFFER n?	The total number of buffers available is displayed x. Enter D and ACCEPT to display a breakdown of all available shared buffers. Enter ACCEPT to retain displayed buffer status.

Enter 0 and ACCEPT for no shared buffer.
Enter a shared buffer number and ACCEPT.

If no shared buffer is required the next prompt is ignored.

7 The selected buffer is displayed for confirmation.
n SECTORS/TRANSFER?
Enter ACCEPT to retain displayed size (n)
Enter number of sectors required to use within buffer and ACCEPT.

8 DATA-SET REQUIRES x SECTORS
ON DISK: n?
Enter ACCEPT to default to displayed disk number.
Enter disk number (range 1-277) and ACCEPT. (x = total sectors required by file).

9

DISK	FREE SECTORS	LARGEST BLOCK	DISK SIZE
n	n	n	n

START SECTOR: n?
Enter ACCEPT to default to optimum start (n) as recommended.
Enter start sector (within limits displayed) and ACCEPT.

10 (PROTECTED?
(UNPROTECTED
Enter ACCEPT to retain existing protection status.
Enter REJECT to change protection status.

11 (CONTROL RECORD F.S.B. @ pp/ccss?
(DISK BASED F.S.B.?
(NO F.S.B.?
Enter ACCEPT to default to displayed file status block.
Enter a memory address (format pp/ccss) and ACCEPT.
Enter D and ACCEPT for disk based F.S.B.
Enter N and ACCEPT for no F.S.B.

If no file status block (F.S.B.) is requested the next prompt is ignored.

12 (MONITOR?
(NO MONITOR?
Enter ACCEPT to retain monitor status.
Enter REJECT to change monitor status.
(The file status block is monitored through the FE utility).

All file details will be redisplayed for confirmation.

13	O.K.?	Enter ACCEPT to update catalogue. Enter REJECT to return to step 1.
14	MOVE CONTENTS?	Enter ACCEPT to copy the file from its original location to the new location. Enter REJECT to leave contents.

If the contents are moved the next two prompts are ignored.

15	CLEAR DATA?	Enter ACCEPT to clear the new definition. Enter REJECT to leave the definition unchanged.

If the definition has no F.S.B. the next prompt is ignored.

16	CLEAR F.S.B.?	Enter ACCEPT to clear the File Status Block. Enter REJECT to leave the F.S.B. unchanged. The program returns to step 1.
17	PROCEED?	Enter ACCEPT to delete the definition. Enter REJECT to return to menu.
18	FILE LENGTH = n SECTORS?	Enter ACCEPT to retain displayed definition length. Enter length in sectors and ACCEPT.

If file type was PD go to step 6. If file type was OL go to step 8.

SCAM

OPTION 3 INDEXED FILE DEFINITION

Step	Prompt	Operator Action
1	CATALOGUE ENTRY NAME?	Enter indexed file name (max 12 chars) and ACCEPT.

(Entry of less than 12 characters will initiate an alpha-match search of the Catalogue index. Matching keys are displayed, the operator may choose from these or elect to use the actual input key.)

If the name exists on the catalogue the details of the index will be displayed.

2	(NO DATA FILE REQUIRED? (DATA FILE IDENTIFIER n?	Enter ACCEPT to default to displayed file identifier (n). Enter a file ID (range 0-377) and ACCEPT. Enter DELETE and ACCEPT to remove name from catalogue and go to step 7. Enter NONE and ACCEPT for no data file.
3	SECONDARY INDEX FILE IDENTIFIER n?	Enter ACCEPT to default to displayed file ID (n). Enter file ID (range 0-377) and ACCEPT.
4	PRIMARY INDEX FILE IDENTIFIER n?	Enter ACCEPT to default to displayed file ID (n). Enter file ID (range 0-377) and ACCEPT.
5	LOGICAL KEY LENGTH (WORDS) n?	Enter ACCEPT to default to displayed key length (n). Enter key length and ACCEPT.

The definition will be redisplayed for confirmation.

6	O.K.?	Enter ACCEPT to update the catalogue and return to step 1. Enter REJECT to return to step 1.

The utility will ascertain that the delete is valid and if so will prompt:

7	PROCEED?	Enter ACCEPT to delete and go to step 1. Enter REJECT to go to step 1.

SCAM

OPTION 4 FILE TABLE EXTENSIONS DEFINITION

Step	Prompt	Operator Action
1	CATALOGUE ENTRY NAME?	Enter file table extension name (max 12 chars) and ACCEPT.

(Entry of less than 12 characters will initiate an alpha-match search of the Catalogue index. Matching keys are displayed, the operator may choose from these or elect to use the actual input key.)

If the name exists on the catalogue the details of the extension will be displayed.

2	FILE 0xx ENTRY REQUIRED?	Enter name and ACCEPT. Enter DELETE and ACCEPT and go to step 4. Enter NONE and ACCEPT for no entry.

If a catalogue name is entered the details will be displayed.

The above prompt is repeated for 3 further levels of extension before the next prompt.

3	O.K.?	Enter ACCEPT to update the catalogue and go to step 1. Enter REJECT to go to step 1.

4	The utility will ascertain that the delete is valid and if so will prompt	
	PROCEED?	Enter ACCEPT to remove the name from the catalogue and go to step 1. Enter REJECT to go to step 1.

OPTION 5 RE-NAME ENTRY

Step	Prompt	Operator Action
1	Catalogue Entry Name?	Enter existing name and ACCEPT.
2	Re-name as?	Enter new name (max 12 chars) and ACCEPT.

The names changed will be displayed.

SCAM

OPTION 6 RE-NUMBER DISK

Step	Prompt	Operator Action
1	Disk Number?	Enter existing disk number and ACCEPT.
2	Re-number as?	Enter new disk number (1-277) and ACCEPT.

All catalogue entries relating to the original disk will be changed to the new one.

SCAM

OPTION Z CATALOGUE REORGANISATION

Since the System Catalogue is an Indexed Sequential file it may require reorganisation. Reorganisation operates only on the indexes not the catalogue itself.

SCAM

OPTION 8 COMPRESS DISK EXTENT

This routine will on a selected disk remove the free sectors dispersed amongst the data-sets and make a contiguous block of free space at the end of the disk. This utility will check each data-set to see if it begins where the previous data-set ended and if not will move it and display the name on the screen.

N.B. If COMPRESSing the System disk, the definitions of the Catalogue Data and the Systems Tables Data-sets MUST be re-defined to all relevant Configuration Table(s) using utility COMA immediately after the compression ends.

SCAM

OPTION 9 PRINT CATALOGUE

Step	Prompt	Operator Action
1	By Disk?	Enter ACCEPT for all disk catalogues in disk order. Enter A and ACCEPT to print the Access Catalogue entries. Enter a disk number and ACCEPT for the catalogue to the entered disk only. Enter REJECT for an alphabetical print by catalogue name.
2	From?	Enter ACCEPT to default to the start of the catalogue. Enter start catalogue name and ACCEPT.
3	To?	Enter ACCEPT to default to the end of the catalogue. Enter end catalogue name and ACCEPT.

SCAM

OPTION 10 DISPLAY DISK LAYOUT

This utility provides the same information as the catalogue print displayed on the screen. The space bar may be used to stop and restart the display. The ESC key will abort the display. Entry of G preceeding the disk number will display the gaps i.e. free space entry.

SCAM

OPTION 11 COPY DATA-SETS

Step	Prompt	Operator Action
1	SOURCE DATA-SET NAME?	Enter data-set name and ACCEPT.

The file details of the source will be displayed for confirmation.

2	TARGET DATA-SET NAME?	Enter data-set name and ACCEPT.

The file details of the target will be displayed for confirmation.

3	O.K.?	Enter ACCEPT to copy file. Enter REJECT to go to step 1.

SCAM

OPTION 12 EXTENDED DIRECT ACCESS DEFINITION

The standard Direct Access file is capable of holding up to 65,535 records occupying a single disk extent. This is satisfactory for most applications, but occasionally two problems arise.

Firstly with the current range of disk media, the number of sectors on one disk extent is appreciably less than that required. For example a typical Customer record is one sector in length. This immediately forces a maximum of about 12,000 Customers within one Direct Access file when using the standard DD1600 disk drive.

Occasionally a particular file is required to have more than 65,535 records available. On some larger sites for example, the Sales Transaction file may be of this type. This file may or may not require more than a standard disk extent.

To overcome these problems, it is possible to create a number of Direct Access Data-Sets extending over different disks and then link them together to form a logical file of the desired proportions.

When these Direct Access Data-Sets have been created (see Option 2) the following is used to link them together. The Catalogue entry produced here is the one used to define the file (see MAFS).

Step	Prompt	Operator Action
1	Catalogue entry name?	Enter the name (up to 12 chars.) and ACCEPT. Enter ESC to return to menu.

(Entry of less than 12 characters will initiate an alpha-match search of the Catalogue index. Matching keys are displayed, the operator may choose from these or elect to use the actual input key.)

If the entry already exists, a full display of the constituent Direct Access Data-Sets will be produced.

2	Minimum Record Number 999999 ?	Enter ACCEPT to leave unchanged. Enter the required min. record no. and ACCEPT. Enter DELETE to remove the entry from the Catalogue; go to step 8.

3 Maximum Record Number 999999 ?
Enter ACCEPT to leave unchanged.
Enter the required max. record no. and ACCEPT.

N.B.
Overall record numbers for EDAs can be specified as standard single word (1-65,535), 17 bit single word (1-131,070) or double word. In the latter case all FSB processing must be performed by the applications programmer and the EDA cannot be used within an Indexed Sequential file structure.

If the maximum record number specified is less than double word, the Key field may still be specified as double word.

4 Use Single Word Key Field?
Enter ACCEPT to use single word.
Enter REJECT to force double word.

All EDAs are automatically specified as having a disk-based FSB, if the Key length is single word only.

5 Monitor?)
No Monitor?)
Enter ACCEPT to leave unchanged.
Enter REJECT to change status.

If the current entry already exists, each constituent Direct Access Data-Set is displayed.

6 Entry required?
Enter ACCEPT to leave unchanged.
Enter the data-set name required and ACCEPT.
Enter ESC to cancel the current input; go to step 1.

As each Direct Access Data-Set is displayed, the EDA's current minimum, maximum and required maximum record numbers are displayed for reference.

Step 6 is repeated until all required Data-sets have been specified.

7 O.K.?
Enter ACCEPT to load/amend the entry.
Enter REJECT to go to step 1.

8 If the entry is available for deletion:
Proceed?
Enter ACCEPT to start the delete.
Enter REJECT to return to the menu.

9 Delete Data Sets?
Enter ACCEPT to delete the Direct Access Data-Sets which make up the EDA as well as the EDA definition.
Enter REJECT to delete only the EDA definition.
Enter ESC to go to step 8.

The required deletion is performed; go to step 1.

SCAM

Option 13 DELETE UNUSED CATALOGUE ENTRIES

Step	Prompt	Operator Action
1	START CATALOGUE ENTRY NAME	Enter the name (max 12 chars) and ACCEPT. ACCEPT to default to start of catalogue. ESC to return to menu.
2	END CATALOGUE ENTRY NAME	Enter the name (max 12 chars) and ACCEPT. ACCEPT to default to end of catalogue.
3	SELECTIVE DELETION?	ACCEPT to allow an operator decision for every unused entry. REJECT to allow deletion without operator intervention.
4	PROCEED?	ACCEPT to continue.

If selective deletion was requested the utility will ask the operator to confirm or reject the deletion of every unused entry. ESC will terminate the selected range.

If selective deletion was not requested the utility will delete all unused entries within the selected range without further operator intervention. The screen display may be stopped and restarted by use of the space bar. ESC will terminate the the selected range.

SE - SYSTEM ENQUIRY

This utility is available to show the status of the system at any point in time in terms of which tasks are engaged on which routines and what stage they are at. The system information is displayed, for all Tasks, immediately upon entry to the utility.

If at the prompt Monitor? the reply is ACCEPT, or a list of Task numbers i.e. 1,3,4,5,7 and ACCEPT then the display is continuously repeated until ESCAPE is pressed, either for all Tasks or those specified. For the best effect certain calling I/O Stations should be in PAGE Mode. If the reply REJECT only is entered then the System Status information will simply be redisplayed once. The display can be stopped/started at any time using the SPACE bar.

Various static machine details are given:

a) current OS Version and Release numbers
b) memory size
c) current system date
d) site name
e) system name
f) backing storage configuration and current disks on-line

I/O Stations precede printers in the display and immediately adjacent to any I/O Station task number will be the identification letter of the default printer COMMANDed by that task. If this position is blank then that I/O Station cannot use any Printer. The letter 'I' immediately following the default printer indicates that the I/O Station has the 'Flash' function turned off. No 'Flash' Messages will be received by this screen. All printer tasks will have their printer identification letter adjacent but separated from the task number by a space.

The AT Column contains the page number of the first page of the 2K memory partition assigned to each task and the assigned memory bank number.

The name of the program currently being run (the literal IDLE appears if no program is being run for I/O Stations, or simply blank for Printers) on each task appears in the RUN column. A Program name may be displayed for a Printer task even though the task is known to be idle. The name displayed refers to the last Program to use that partition and indicates that if the next posting to that Printer is for the same Program, the Program itself will NOT be re-loaded.

The P/Q column gives the number of the Spool queue into which the task is posting, suffixed with the letter R if the queue was requested with Reprint.

For I/O Stations the U/Q column denotes the latest Plain Paper Queue selected, by the operator, via the 'U' Command; for Printer and Background partitions this column denotes the Spool queue currently being Unspooled from.

The POST column indicates the Program name used in requests for background processing. This name is usually the same as the RUN Program name, but for more complex procedures may well be different. One RUN Program can POST to various other Programs in one cycle.

The T and SYS columns relate which SYStem the routine being run on this task has been extracted from and on which file Table that System is currently residing.

The PHASE column indicates the basic function that task requested the OS to perform last.
PROGRAM AT is an attempt, by SE, to give detail relating to the PHASE for each task.
The actual memory address at which the PHASE was requested is displayed in the ADDRESS column.

Lastly the actual hardware Device Code(s) and also the software Device Type are given to facilitate easy location of ports and screen/printer types.

SMS - SWAP "FILE 90% FULL" MESSAGE STATUS

If a Data file is maintained by using a Disk based F.S.B. and the File Status Block Processor subroutine, then the user is automatically informed when the file is over 90% full each time new entries are added.

This facility may be suppressed if required, though it is stressed that the message should NOT be ignored.

Step	Prompt	Operator Action
1	File I/D?	Enter required octal file identifier and ACCEPT or ESC to return to PROGRAM? prompt.
	The File Identifier, Data Set name and the current message status are displayed.	
2	Swap?	Enter ACCEPT to change the message status or REJECT to leave unchanged; return to step 1. Enter ESC to return to PROGRAM?.

N.B.
The selected message status (active or inactive) will remain in force until the current System is reloaded or the machine is Bootstrapped. The message is active as a default on these occasions.

SUDS - SPECIAL UTILITY FOR DISK SECURITY

This utility will allow the operator to perform a security on any correctly labelled master disk in the system. Each disk number will be shown to the operator for confirmation before performing the normal security procedure as detailed in the relevant disk labels.

SUFR - SPECIAL UTILITY FOR FILE RE-ORGANISATION(S)

This utility will allow the operator to re-organise selected Indexed-Sequential file structures.

SUM - ARITHMETIC CALCULATOR

This utility will perform arithmetic calculations of any length based upon the operators

\+ add
\- subtract
/ divide
* multiply
( ) priority brackets.

Arithmetic operators follow the mathematical rules of priority as follows

1 contents of brackets evaluated
2 divisions evaluated
3 multiplications evaluated
4 additions evaluated
5 subtractions evaluated.

for example

7 * (144 + 8) / 2 = 532

The utility will cater for single, double, or triple word entries in decimal, octal or hexadecimal. Octal entries will be preceded by the letter O for example:

```
O 1          ..... Single Word
O 1,1        ..... Double Word
O 1,1,1      ..... Triple Word
```

Hexadecimal entries will be preceded by the character @ for example.

```
@ A             ..... Single Word
@ A,1001        ..... Double Word
@ A,1001,09AC   ..... Triple Word
```

Decimal entries of the form 999,999 may be entered.

The utility will display the result of a calculation in decimal, octal and hexadecimal and will also keep a running total of individual calculations again in decimal, octal and hexadecimal.

The ACCUMULATED result of computations and the IMMEDIATE result of the last expression may be re-input as part of the next expression by using 'A' and 'I' within that expression.

Any divisions are taken to 4 decimal places.

If decimal places exist in a preceding expression then any whole number input thereafter will be displayed with decimal places.

Step	Prompt	Operator Action
1	SUM?	Enter the calculation to be performed and ACCEPT. Enter ESC to return to PROGRAM?
2		Calculation result and accumulated result are displayed.
3	SUM?	Enter next calculation to be performed and ACCEPT and return to step 2. Enter ESC to clear accumulated result and return to step 1.

TACK = TEXT PRINT ON LABELS

This utility is designed to reproduce in name and address tacky label format an entered text utilising up to three column stationery.

Step	Prompt	Operator Action
1	LABELS/LINE?	Enter the number of labels across the page, 1, 2 or 3 Enter ACCEPT only for single label stationery
2	COPIES	Enter the number of labels to be printed containing the required text in the range 1-9999 and ACCEPT.
3	---TEXT--- (up to 7 lines)	Enter up 120 chs including up to 6 carriage returns and ACCEPT.

The labels print request is posted to the assigned queue and the program returns to step 1.

SECTION 8

OPERATING SYSTEM COMMANDS

CONTENTS

	PAGE
Introduction	8.1
I/O Station Commands	8.1
Printer Commands	8.2

OPERATING SYSTEM COMMANDS

Introduction

Any I/O Station may enter instructions to the Operating System by switching to 'Command' mode regardless of the program currently running at that I/O Station; the program will resume at the point of interruption upon completion of, or escape from, Command Mode.

Command Mode is obtained by pressing the CLEAR key or COMMAND key on a VDU whereupon the COMMAND? prompt will be displayed. Please note that the key that invokes the Command function on a VDU may differ according to the type of VDU being used. Described below are the facilities available within Command Mode and the Operator entries required in order to use those facilities.

I/O Station Commands

1. Check Digits Enquiry

When in Command Mode it is possible to find out the check digit associated with any number by simply entering that number, whereupon the corresponding check digit will be displayed.

2. Set the System Date

When in command mode the Operator may set up the machine date by entering the command

/dd mm yy

Where dd mm yy represents the day, month and year. Leading zeros are not required to be entered but the day, month and year must be separated by a single space.

4. Allocate Reports to Plain Paper Print Queue

For any program which allocates details for printing to any plain paper queue, the Operator may specify which print queue that shall be prior to entry of that routine. This may enable printer output to be segregated and printed out in convenient sections at some later stage.

Some programs which allocate details to a plain paper print queue will do so to a specific plain paper queue, and in this case it will not be possible to override the program by using this command.

Where programs do spool details to any general plain paper queue then the command Un may be used; this will result in the details being spooled to print queue n where n is defined using Utility MAPS.

5. Change File Table

It is possible for different screens to operate on different systems. In order to achieve this it is necessary that the file information relating to those different systems be accessible. To this end, it is possible to establish up to 8 areas of memory through which file information for up to 8 different systems may be accessed. Having reserved 8 areas of memory it is then possible to associate an I/O Station with one of those areas by use of the command Tn where n is in the range 0-7. Having associated an I/O Station with one of these reserved areas it is then possible by use of the MAPS utility described in section 7 to establish access to all files for a particular system within that reserved area ie. establish all FCB's for that system.

6. Change 'FLASH' status of I/O Station

System messages will normally be flashed to all I/O Stations unless the COMA utility has been used to suppress 'flashed' messages to particular I/O Stations. By use of the command '*' the flash status of the I/O Station may be reversed.

7. Enquire on System

Entry of the command '?' will display the current Application System name.

Printer Commands

Every I/O Station within the Installation's configuration will have the ability to control a printer ie. spool details which will only be printed on a specified printer. This link between an I/O Station and a printer is established by the MAPS utility described in section 7.

The printer command functions described below will normally activate that printer controlled by the I/O Station from which they are entered. At installations with more than one printer, the I/O Station may activate a different printer from that which it normally controls by simply prefixing the command with the printer's identification letter eg. A, B, C etc.

1. Suspend a printer

By entering the command S the printer program will be suspended when one further line has been printed. This is particularly useful when making minor adjustments to paper alignment.

2. Restart a Suspended Printer

By entering the command R the suspended printer program will resume normal operation.

3. Restart a Suspended Printer with Restrictions.

By entering the command Rn the suspended printer program will resume printing but will suspend again after n lines have been printed.

4. Cancel or Quit a Printout

By entering the command Q the suspended printer program will be cancelled and any details awaiting printing associated with it will be removed from the Spool File.

If the printer program itself disallows cancellation then this command will be rejected by the Operating System.

5. Print Work from a Print Queue

By entering the command Pn the printer will print any details awaiting printing in the print queue n.

Immediately before changing stationery, a P0 command should be issued to ensure that the printer remains idle whilst changing stationery (print queue 0 is always empty). Having loaded the correct stationery the command Pn should then be entered. If print queue n is, or becomes empty, then the printer will remain inactive until further work is allocated to that print queue by the I/O Station, or until the Operator commands the printer to print details from a different print queue by a subsequent Pn command.

6. Verify Stationery Line Up

Normally, one print queue is reserved for every different customised stationery format.

By entering the command Vn where n is the print queue number required a stationery alignment facility will be invoked for the customised stationery details associated with that print queue.

See Utility MAPS for description of the queue definition.

SECTION 9

MISCELLANEOUS PROGRAMMER INFORMATION

CONTENTS

	PAGE
USEFUL PROGRAMMER INFORMATION	9.1
Program Access During Security	9.1
Printout Suppression	9.1
Disk Status Problems	9.1
Mains Power Problems	9.2
Suspended Device Problems	9.2
I/O Station Program Halts	9.3
MANUAL BOOTSTRAP PROCEDURES	9.4
Normal Start up	9.4
Modify Configuration Prior to Start up	9.5
Recover from security	9.5
Disk Label Facility	9.6
CONTROL PANEL	9.7

USEFUL PROGRAMMER INFORMATION

Program Access During Security

If a program is needed during a security copy operation and has not been set up by the utility POMM, then it can be entered by setting the CPU Switch Register to the appropriate I/O Station Task Number.

Printout Suppression

It may sometimes be convenient to be able to suppress at will the print-out from any program(s) as distinct from merely suspending the print operation. This is achieved by use of the Control Panel switches as follows:

Bit 17 set	=	Suppression Request
Bit 16 set	=	Suppress all printer tasks except that specified in bits 8-1
Bit 16 clear	=	Suppress only the printer task specified in bits 8-1.
Bit 8-1	=	printer task number (or zero)

Disk Status Problems

Standard operating system action upon encountering hardware status on a disk transfer is to re-try continually until the status clears, flashing a "STATUS" message after 3 successive failures.

If, however, control panel switch 13 only is set after the first "STATUS" message has occurred processing will continue AS IF THE TRANSFER HAD BEEN SUCCESSFUL.

It is not necessary to put the AUTO/MANUAL switch to the MANUAL position. Switch 13 should be returned to the DOWN position immediately after use. This facility is obviously only to be used with extreme caution!

The status message will be of the form:

UNIT nn WRITE Status nnnnnn, Sector nnnnnn, Disc nnn
READ

where the unit field is a 2 digit octal number identifying the Disk Unit (Controller).

The sector identifies the base sector of the transfer (not necessarily the faulty sector as up to 8 sectors may be involved in the transfer) and the status field identifies the status register contents as follows:

For DD1600/DD818/D8000 Drives

Bit 10	-	Monitor
Bit 9	-	Seek in progress
Bit 8	-	Seek error
Bit 7	-	Data late
Bit 6	-	Address Incorrect
Bit 5	-	Check word Incorrect
Bit 4	-	Temperature Check
Bit 3	-	Off-Line
Bit 2	-	Logic or Write Check
Bit 1	-	Fault Exists

For DD9600 Drives

Bit 7	-	Programme error - e.g. non valid head or track address
Bit 6	-	Address error
Bit 5	-	CRC error
Bit 4	-	Write protected (and attempted)
Bit 3	-	Offline - not up to speed, not connected
Bit 2	-	Logic Fault (hardware or drive fault)
Bit 1	-	Common Fault Bit

The Disc or surface is identified by its three digit octal label number. If this field appears as asterisks then the System has failed to read even the label.

Mains Power Problems

Occasionally, mains electricity voltage fluctuations can occur which cause all or part of the machine configuration to lock out. This is an engineering problem, though in the short term following the procedure outlined in 'Suspended Device Problems' may help the user to continue working.

Suspended Device Problems

Experience has shown that program KICK does not always work on suspended devices. A better method is to force a mains fail condition as follows:

1. Set the AUTO/MANUAL switch to MANUAL.
2. LOWER, and then RAISE back to NORMAL, the "CONT INT" Switch.
3. Processor has halted with numbered lights 5 and 1 only lit.
4. Press continue.
5. Return the AUTO/MANUAL to auto.

I/O Station Program Halts

If an I/O Station program stops with a HALT message, there are facilities available to restart the program from that screen:

Keyboard input	Action
! (Shift-1)	Restart at Default Restart Address
% (Shift-5)	Restart at step following the HALT
*	Restart at PROGRAM? prompt

N.B.
This facility should ONLY be used by authorised personnel.

MANUAL BOOTSTRAP PROCEDURES

Normal Start-Up

To initiate the Operating System proceed as follows:

1. Place appropriate Systems Disk on Device 7x Drive n.
2. Set the AUTO/MANUAL switch to MANUAL and raise STOP.
3. Press RESET.
4. Load the following bootstrap at 000100, if not already present:

```
000100    01067x    where x denotes the unit 0-2
   101    01447x
   102    210107
   103    01557x
   104    01277x
   105    020104
   106    020200
   107    000200
```

This bootstrap is permanently memory resident whilst the Operating System is running.

5. Set A-register as follows:

DD1600	DD9600
Zero (All switches down)	Octal 100 (Switch 7 only up)

6. Set B-register as follows:

DD1600

B17 = Exchangeable/Fixed Surface (0/1).
B16-15 = Drive code (0-3).
B14-01 = MUST be zero.

DD9600

B17-16 = Drive code (0-3).
B15-11 = Surface code (currently 0-3).
B10-01 = MUST be zero.

7. Clear all switches.
8. If the disk is READY, press LOAD.
9. Return the AUTO/MANUAL switch to AUTO.

Initiation occurs almost immediately; each I/O Station will bell and display "PROGRAM?" provided the disk(s) holding the required System Files are on-line.
Otherwise, each I/O Station will flash a message indicating which disk is required.

Modify Configuration Prior to Start-up

This facility is provided to overcome start-up problems caused when the physical configuration does not match the Configuration Table as set up. For example, a memory stack may have gone down or a VDU become inoperable.

Follow the normal start-up as far as step 7.

7. RAISE SWITCHES 6 and 1 to 3; see below.
8. If the disk is READY, press LOAD.
9. Return the AUTO/MANUAL switch to auto.

The Configuration Table on the selected surface will be amended to cater for one I/O Station, a single File Table, up to 64K, no Resident Overlays.

The I/O Station Device Type will be represented by switches 1 to 3 as follows:

Device	Device No.(Octal)
CIFER	1
CDC	2
CT800	3
CT760	5
CT820	6

Initiation then proceeds as indicated by the status of switches 16-7 details of which follow.

When the "PROGRAM?" prompt is obtained the utility "COMA" may be called to amend the Configuration Table to the current physical configuration.

Recover From Security Disk

A feature is provided to enable recovery to be made from a security disk or disks. The disk containing the operating system etc. will always be recovered as part of the bootstrapping procedure. At single drive sites it is probable that both exchangeable and fixed disks would be recovered as part of the bootstrap procedure. All other data disks must be recovered by use of the utility RECY (see UTILITIES section). When copying is complete, all I/O Stations will Bell and display "PROGRAM?".

To initiate recovery from a security follow the normal start up procedure as far as step 7.

7. RAISE SWITCH 9.
8. If the disk is READY, press LOAD.
9. Return the AUTO/MANUAL switch to AUTO.

Initiation will occur and recovery copying will proceed automatically. The Configuration can be modified at the same time; see above.

Disk Label Facility

A facility is also provided when bootstrapping the OS to label disks as follows:

Follow the normal start-up procedure as far as step 7.

7. RAISE SWITCH 7. N.B. See note below.
8. If the disk is ready, press LOAD.
9. Return the AUTO/MANUAL switch to auto.

One of the I/O Stations will activate:

I/O Station Display	Operator Action
	Enter 70U and ACCEPT (71U for device 71 etc)
SURFACE?	Enter surface number (0-n) followed by Drive type (H/L for Hawk/Lark) and accept eg. 0L
DISK?	Enter octal disk number followed by Master/Security indicator (M/S) and ACCEPT e.g. 27M Enter R and ACCEPT to read current label and display

N.B.
This disk label is only to be used in emergency procedures and does not produce the full disk label required by a working system for security procedures.

Step 7 above will usually activate Task 01. This facility can be extended to activate a different screen by setting bit 17, instead of bit 7, and setting bits 16-9 and 8-1 to correspond to the required input and output device codes respectively.

THE CONTROL PANEL

The MOLECULAR 18 is provided with a Control Panel through which the programmer may conveniently exercise total manual control of any application.

The Control panel includes the following set of control elements:

1	Power on/off pushbutton
1	Key operated security lock
31	Indicator lamps/lights
17	Data switches
14	Control switches

The power on/off button should be pushed in to switch on the CPU - it will then lock in and light up. To switch off the button should be pushed again.

The security lock is a two-position, key operated locking switch which can disable all switches on the panel except the 17 data switches, which can then only be accessed from program by specific instructions (ESRA and ESRB - enter switch register into Accumulator A or B). This lock eliminates any accidental manual input to the system.

With the AUTO/MANUAL switch in the auto position when the computer is switched on, the OS starts from the address set on PC and MA, just as if the CONTINUE switch has been pressed. When in 'manual' mode, all control panels switches are enabled. If the computer is switched on with the switch in the manual position it will not automatically start up but will have to be started by following a start up procedure ie. set up a small program whose function is to access a larger program which in turn calls the OS and hands over control to it. This is known as Bootstrapping.

The 3 voltage lamps relate the correct functioning of devices attached to the computer ie. disk drives, VDUs etc. Depending upon the configuration a particular combination of these lamps will always be lit when the computer is running. Subsequent failure of one of these lamps will usually indicate a fault with the device or interface (unless of course the lamp itself has failed).

Four lamps are available to display control conditions; very few of these display useful information while the computer is running because they change too rapidly, and so they are described in terms of the information they display when the processor has stopped.

The FETCH lamp indicates that the next program cycle will be used to 'fetch' an instruction from memory.
The EXECUTE lamp indicates that the processor cycle will be used to reference memory for an operand in a move data or modify memory instructions.

The INDIRECT lamp shows that the next cycle will be used to fetch an address word in an indirectly addressed memory reference instruction.
The RUN lamp indicates that the processor is in normal operation.

Every time data is written or read a parity check takes place; this helps the CPU to discover when erroneous data has been recorded due to a fault. The PARITY lamp is used to indicate such a condition. The INT. ON lamp indicates that an interrupt is waiting to be actioned. The GREATER THAN and CARRY lamps are used to show the state of the respective flags.

There are seventeen other indicator lamps. These are 'bit indicator' lights numbered 17 to 1 from left to right, forming a register with Bit 17 being the most significant and Bit 1 the least significant. This register can represent and display either the address or the contents of a word, depending on which switches are in use at the time.

Below the bit indicator lights is a set of Data switches used for direct communication with the central processor. This is a set of 17 toggle switches used to specify binary numbers, which are then loaded into the appropriate registers when other console switches are operated or loaded into the Accumulator(s) at Program command (ESRA or ESRB). The 17 positions represent a 17-bit binary word.

Some of the control switches are actually a three positional spring-loaded switch with a common off position in the centre. Lifting a switch lever up loads the contents of the data switches into the specified position. Pressing a switch down displays the contents in the bit-indicator lights, to enable the user either to examine what has just been loaded, or simply to examine the contents of whichever register is specified.

This means that the Operator can manually load switch register details into, or display the current contents of Accumulator A, Accumulator B, the Program Counter (PC) which represents the address of the next instruction to be fetched out of memory, or the Memory Address Register (MA) which represents the address of the memory word being examined or loaded.

Having loaded the Memory Address with the required position, the CURRENT switch may be used to load this location with the contents of the data switches and/or examine the contents of said location. With the 'current' location loaded with data from the switches the NEXT switch may be used to load the next consecutive word by resetting the data switches to the required pattern and pressing the 'NEXT' switch up. This operation may be repeated as often as necessary because use of the NEXT switch increments the Memory Address by 1.

Eight of the control switches have dual functions; raising the switch initiates the condition printed above it and lowering the switch initiates the condition printed below.

AUTO Allows the processor to run without intervention from any of the other switches.

MANUAL Allows the control panel switches to exercise control over the processor and registers.

RESET When the switch is pressed up all Interrupts are dismissed and the PC register and MA register are reset to the start of memory.

LOAD Performs a reset before attempting to bootstrap.

AUTO-STEP This switch provides slow speed operation of the Instruction Step (an average of 5 instructions per second)

I/O RESET Clears the busy and done status of all devices.

INSTRUCTION STEP When the switch is depressed, the processor will execute one complete instruction from wherever the PC and MA have been set. In other words, it will operate from the beginning of a FETCH, completing the current instruction. The address displayed on the bit-indicator lights after each INSTRUCTION STEP is the address of the next instruction word (consecutively).

STOP When the switch is pressed up program execution will be halted at the end of the instructions in progress, when the run light will go off.

CONTINUE When the switch is depressed, the run indicator light will be turned on, and the processor will begin normal operation, executing the instruction at the location specified by the PC and MA.

CONTINUOUS INTERRUPT An interrupt is called after every step, assuming Interrupt is on. Note that NO Interrupt can occur between a Jump Indirect Instruction and the next step. This does not apply to any other instruction whatsoever, only an Indirect jump.

INTERRUPT OFF This switches off all Input/Output external interrupts, but does not switch off internal processor interrupts such as Memory Boundary Interrupt, Mains Failure Interrupt, Mains On Interrupt, Memory Parity Interrupt, Memory Protect Interrupt, Memory Address = Switch Register Interrupt and Continuous Interrupt Switch Interrupt.

INTERRUPT IF MA=SW Having first set an address on the Data switches, this switch may be depressed; the program is then set to Continue until the Memory Address equals the address previously set on the switches, when an Interrupt is called.

SECTION 10

APPENDICES

CONTENTS

	PAGE
SUBROUTINE CHECK LIST (ALPHABETIC)	10.1
SUBROUTINE CHECK LIST (NUMERIC)	10.4
PROGRAM INSTRUCTIONS SUMMARY	10.7
REGISTER INSTRUCTIONS LOOK UP TABLE	10.8
COMBINATIONS OF REGISTER INSTRUCTIONS PERMITTED	10.9
BIT NUMBER AND DECIMAL CHART	10.10
SYSTEM HALTS	10.11
PROCESSOR HALTS	10.12
DISC STATUS CHART	10.13
TRANSLATION TABLE	10.14
MASKS AND CONSTANTS	10.17
STANDARD MEMORY/REGISTER VALUES	10.20
GET - PARAMETERS AND CONTROL WORD FORMATS	10.22
- Character Mode	10.22
- Binary Mode	10.24
- Fetch Mode	10.26
- Split	10.28
MISCELLANEOUS CONTROL WORDS	10.29
- Extract from File Control Block	10.29
- Binary to ASCII	10.30
- File Status Block processor	10.31
OP/BREAKPOINT HANDLER COMMAND SUMMARY	10.32
AVAILABLE HARDWARE PRINT OPTIONS	10.33
AVAILABLE HARDWARE I/O STATION DISPLAY OPTIONS	10.35

SUBROUTINE CHECK LIST

(Alphabetic sequence)

DATA HANDLING

OCTAL	SUBROUTINE CALL	FUNCTION
037711	JSBR IZ 1711	Add Words
037713	JSBR IZ 1713	Add to Double Word
037740	JSBR IZ 1740	Calculate Check Digit
037710	JSBR IZ 1710	Clear Block of memory
037723	JSBR IZ 1723	Compare Blocks of memory
037750	JSBR IZ 1750	Complement Double Word
037747	JSBR IZ 1747	Compute
037765	JSBR IZ 1765	Convert to ASCII
037762	JSBR IZ 1762	Convert to Binary
037606	JSBR IZ 1606	Convert to Binary and Test
037637	JSBR IZ 1637	Convert to/from Metacode
037702	JSBR IZ 1702	Convert to Negative if positive
037605	JSBR IZ 1605	Convert to octal Address
037612	JSBR IZ 1612	Convert to octal digits
037701	JSBR IZ 1701	Convert to positive if negative
037772	JSBR IZ 1772	Convert to Upper Case
037732	JSBR IZ 1732	Divide with remainder
037745	JSBR IZ 1745	Divide by 10 with remainder
037726	JSBR IZ 1726	Divide and Round
037744	JSBR IZ 1744	Divide by 10 rounded
037707	JSBR IZ 1707	Duplicate block of memory
037743 *	JSBR IZ 1743	Extract System Serial No.
037733	JSBR IZ 1733	Jump to Subroutine at Offset
037632	JSBR IZ 1632	Jump to Subroutine in Resident Overlay
037662	JSBR IZ 1662	Load A from Application Workspace
037721	JSBR IZ 1721	Load A from Offset
037771	JSBR IZ 1771	Load Byte
037741	JSBR IZ 1741	Move and Pad Character string
037734	JSBR IZ 1734	Multiply Words
037761	JSBR IZ 1761	Multiply by 10
037655	JSBR IZ 1655	Name and Address Processor
037607	JSBR IZ 1607	Pack Date
033630	JSBR IZ 1627	Resolve Block of Offset Addresses
037630	JSBR Z 1630	Resolve Offset Address
037627	JSBR IZ 1627	Resolve Offset Address Block
037731	JSBR IZ 1731	Space Fill Block of memory
037663	JSBR IZ 1663	Store A in Applications Workspace

OCTAL	SUBROUTINE CALL	FUNCTION
037725	JSBR IZ 1725	Store A at Offset
037775	JSBR IZ 1775	Store Byte
037712	JSBR IZ 1712	Subtract Words
037714	JSBR IZ 1714	Subtract from Double Word
037706	JSBR IZ 1706	Swap Blocks of memory
037751	JSBR IZ 1751	Unpack Date
037724	JSBR IZ 1724	Zero Test Block of memory

INPUT/OUTPUT HANDLING

OCTAL	SUBROUTINE CALL	FUNCTION
027641 *	JUMP IZ 1641	Error Handler
037641 *	JSBR IZ 1641	Error Handler
037653	JSBR IZ 1653	Flash All I/O Stations
037654	JSBR IZ 1654	Flash Single I/O Station
037640 *	JSBR IZ 1640	Get - binary/character/ fetch/split
037635 *	JSBR IZ 1635	Get Password
037777 *	JSBR IZ 1777	Halt Program
037636 *	JSBR IZ 1636	Test Inhibit
037742	JSBR IZ 1742	Load Overrun
037644 *	JSBR IZ 1644	Print Line
037652 *	JSBR IZ 1652	Put characters
037656 *	JSBR IZ 1655	Put Spaces
023400 *	JUMP Z 1400	PROGRAM? (Clear screen)
023402 *	JUMP Z 1402	PROGRAM? (without clearing)
023402 *	JUMP Z 1402	Exit to Printer Control Program
037643	JSBR IZ 1643	Specify Default Restart Address
037634	JSBR IZ 1634	Specify I/O Station Escape Point
037617	JSBR IZ 1617	Specify Printer termination Options
037610	JSBR IZ 1610	Specify Re-Entry Point
037611	JSBR IZ 1611	Specify Termination Routine
037625 *	JSBR IZ 1625	Suspend
037601 *	JSBR IZ 1601	Suspend Display
037722 *	JSBR IZ 1722	Suspend for number of cycles

SPOOL FILE HANDLING

OCTAL	SUBROUTINE CALL	FUNCTION
037614 *	JSBR IZ 1614	Assign Spool Queue
037602	JSBR IZ 1602	Assign Unspool Queue
037650 *	JSBR IZ 1650	De-queue Posting
037616 *	JSBR IZ 1616	Get Next Spool Record No.
037647 *	JSBR IZ 1647	Post to Print Queue
037776	JSBR IZ 1776	Skip if Original

OCTAL	SUBROUTINE CALL	FUNCTION
037613 *	JSBR IZ 1613	Specify I/O Station Print Queue
037413 *	JSBR IZ 1413	Specify I/O Station Print Queue with reprint
037645 *	JSBR IZ 1645	Spool
037764 *	JSBR IZ 1764	Spool to specified Program name
037651 *	JSBR IZ 1651	Spool and Post to Print Queue
037763 *	JSBR IZ 1763	Spool and Post to specified Program name
037600 *	JSBR IZ 1600	Stow Specified Spool Buffer
037657 *	JSBR IZ 1657	Stow Spool Buffer
037646 *	JSBR IZ 1646	Unspool

DISC FILE HANDLING

OCTAL	SUBROUTINE CALL	FUNCTION
037661 *	JSBR IZ 1661	Delete Index Sequential Record
037700	JSBR IZ 1700	Extract from File Control Block
037665 *	JSBR IZ 1665	File Status Block Processor
037670 *	JSBR IZ 1670	Fetch
037661 *	JSBR IZ 1661	Load/Delete Indexed-Sequential Record
037672 *	JSBR IZ 1672	Overwrite Record
037671 *	JSBR IZ 1671	Rewrite Record

* These subroutines will release any LOCK.

SUBROUTINE CHECK LIST

(Numeric sequence)

OCTAL		SUBROUTINE CALL	FUNCTION
023400	*	JUMP Z 1400	PROGRAM? (clears screen)
023402	*	JUMP Z 1402	PROGRAM? (without clearing)
023402	*	JUMP Z 1402	Exit to Printer Control Program
027641	*	JUMP IZ 1641	Error Handler
033630		JSBR Z 1630	Resolve Offset Address
037413	*	JSBR IZ 1413	Specify I/O Station Print Queue with Reprint
037600	*	JSBR IZ 1600	Stow Specified Spool Buffer
037601	*	JSBR IZ 1601	Suspend Display
037602	*	JSBR IZ 1602	Assign Unspool Queue
037605		JSBR IZ 1605	Convert to Octal Address
037606		JSBR IZ 1606	Convert to Binary and test
037607		JSBR IZ 1607	Pack Date
037610		JSBR IZ 1610	Specify Re-Entry Point
037611		JSBR IZ 1611	Specify Termination Routine
037612		JSBR IZ 1612	Convert to Octal Digits
037613	*	JSBR IZ 1613	Specify I/O Station Print Queue
037614	*	JSBR IZ 1614	Assign Print Queue
037616	*	JSBR IZ 1616	Get Next Spool Record No.
037617		JSBR IZ 1617	Specify Printer termination Options
037625	*	JSBR IZ 1625	Suspend
037627		JSBR IZ 1627	Resolve Block of Offset Addresses
037630		JSBR IZ 1630	Resolve Offset Address
037632		JSBR IZ 1632	Jump to Subroutine in Resident Overlay
037634		JSBR IZ 1634	Specify I/O Station Escape Point
037635	*	JSBR IZ 1635	Get Password
037636		JSBR IZ 1636	Inhibit
037637		JSBR IZ 1637	Convert to/from Metacode
037640	*	JSBR IZ 1640	Get-binary/character/fetch/ split
037641	*	JSBR IZ 1641	Error Handler
037643		JSBR IZ 1643	Specify Default Restart Address
037644	*	JSBR IZ 1644	Print line
037645	*	JSBR IZ 1645	Spool
037646	*	JSBR IZ 1646	Unspool
037647	*	JSBR IZ 1647	Post to Print Queue
037650	*	JSBR IZ 1650	De-queue Posting
037651	*	JSBR IZ 1651	Spool & Post to Print Queue
037652	*	JSBR IZ 1652	Put characters
037653		JSBR IZ 1653	Flash All I/O Stations
037654		JSBR IZ 1654	Flash Single I/O Station.
037655		JSBR IZ 1655	Name and Address Processor
037656	*	JSBR IZ 1656	Put Spaces
037657	*	JSBR IZ 1657	Stow Spool Buffer

OCTAL	SUBROUTINE CALL	FUNCTION
037661 *	JSBR IZ 1661	Load/Delete Indexed-Sequential Record
037662	JSBR IZ 1662	Load A from Applications Workspace
037663	JSBR IZ 1663	Store A into Applications Workspace
037665 *	JSBR IZ 1665	File Status Block Processor
037670 *	JSBR IZ 1670	Fetch
037671 *	JSBR IZ 1671	Rewrite Record
037672 *	JSBR IZ 1672	Overwrite Record
037700	JSBR IZ 1700	Extract from File Control Block
037701	JSBR IZ 1701	Convert to Positive if negative
037702	JSBR IZ 1702	Convert to Negative if positive
037706	JSBR IZ 1706	Swap Blocks of memory
037707	JSBR IZ 1707	Duplicate block of memory
037710	JSBR IZ 1710	Clear block of memory
037711	JSBR IZ 1711	Add words
037712	JSBR IZ 1712	Subtract Words
037713	JSBR IZ 1713	Add to Double Word
037714	JSBR IZ 1714	Subtract from double word
037721	JSBR IZ 1721	Load A from Offset
037722 *	JSBR IZ 1722	Suspend for a number of Cycles
037723	JSBR IZ 1723	Compare Blocks of memory
037724	JSBR IZ 1724	Zero Test Block of memory
037725	JSBR IZ 1725	Store A at Offset
037726	JSBR IZ 1726	Divide and Round
037731	JSBR IZ 1731	Space Fill Block of memory
037732	JSBR IZ 1732	Divide with Remainder
037733	JSBR IZ 1733	Jump to Subroutine at Offset
037734	JSBR IZ 1734	Multiply Words
037740	JSBR IZ 1740	Calculate Check Digit
037741	JSBR IZ 1741	Move and Pad Character String
037742	JSBR IZ 1742	Load Overrun
037743 *	JSBR IZ 1743	Extract System Serial Number
037744	JSBR IZ 1744	Divide by 10 rounded
037745	JSBR IZ 1745	Divide by 10 with remainder
037747	JSBR IZ 1747	Compute
037750	JSBR IZ 1750	Complement Double Word
037751	JSBR IZ 1751	Unpack Date
037761	JSBR IZ 1761	Multiply by 10
037762	JSBR IZ 1762	Convert to Binary
037763 *	JSBR IZ 1763	Spool and Post to Specified Program name
037764 *	JSBR IZ 1764	Spool to Specified Program name
037765	JSBR IZ 1765	Convert to ASCII
037771	JSBR IZ 1771	Load Byte
037772	JSBR IZ 1772	Convert to Upper Case

OCTAL	SUBROUTINE CALL	FUNCTION
037775	JSBR IZ 1775	Store Byte
037776	JSBR IZ 1776	Skip if Original
037777 *	JSBR IZ 1777	Halt Program

PROGRAM INSTRUCTIONS

Memory Reference Instructions

```
 17 16   15 14 13   12   11 10   9  8  7   6  5  4   3  2  1
 <---OP. Code--->  I/D  Z/C  <--------Memory Address--------->
```

Instruction	Mnemonic	Octal OP Code
Jump	JUMP	02
Jump to Subroutine	JSBR	03
Increment, Skip if Zero	INSZ	04
Decrement, Skip if Zero	DESZ	05
'And' to A	ANDA	06
'Inclusive OR' to A	IORA	07
'Exclusive OR' to A	XORA	10
Add to A	ADA	11
Add to B	ADB	12
Subtract from A	SFA	13
Subtract from B	SFB	14
Add to A with Carry	ADAC	15
Add to B with Carry	ADBC	16
Subtract from A with Carry	SFAC	17
Subtract from B with Carry	SFBC	20
Load A	LDA	21
Load B	LDB	22
Compare A, Skip if not Equal	CMPA	23
Compare B, Skip if not Equal	CMPB	24
Store A	STA	25
Store B	STB	26

Register Instructions

MODE	MICRO INSTRUCTIONS									
Bits: 12	11	10	9	8	7	6	5	4	3	2	1
0	1	A/B 1/0	Clear Carry	Left/ Right	Shift	Rotate	With Carry	Dec.	Inc.	Skip B16=0	Skip b1=0
1	0	A/B 1/0	Clear Accum	One's Comp.	Clear Carry	Comp. Carry	Skip	Swap	Clear Sign	Comp. Sign	Enter S/reg
1	1	A/B 1/0	True/ False	Skip Neg.	Skip Not 0	Skip Carry	Clear Carry	Skip If >0	Clear >Flag	Clear Accum	One's Comp.

Multiple Shift/Rotate

MODE	MICRO INSTRUCTIONS									
12	11	10	9	8	7	6	5	4	3	2	1
0	0	A/B 1/0	1	Left/ Right	Shift Rotate	0	0	No. of Shifts or rotates			

Register Instruction Look-Up Table

Octal Code/Mnemonic A Reg.		Octal Code/Mnemonic B Reg.		Description
003001	ALSB	002001	BLSB	Skip if Bit 1=0
003002	AMSB	002002	BMSB	Skip if Bit 16=0
007600	ANEG	006600	BNEG	Skip if Negative
007500	AN=0	006500	BN=0	Skip if Not Zero
007200	APOS	006200	BPOS	Skip if Positive
007100	A=0	006100	B=0	Skip if Zero
005400	CLA	004400	CLB	Clear Register
007002	CLA	006002	CLB	Clear Register
002400	CLC			Clear Carry Flag
004100	CLC			Clear Carry Flag
006020	CLC			Clear Carry Flag
006004	CLGT			Clear Greater Than Flag
005004	CLSA	004004	CLSB	Clear Sign Bit
005200	CPLA	004200	CPLB	One's Complement of Reg.
007001	CPLA	006001	CPLB	One's Complement of Reg.
004040	CMPC			Complement Carry Flag
005002	CPSA	004002	CPSB	Complement Sign Bit
003010	DECA	002010	DECB	Decrement Register by 1
005001	ESRA	004001	ESRB	Enter Switch Register
003004	INCA	002004	INCB	Increment Register by 1
003240	LRA	002240	LRB	Left Rotate by 1 excl B17
003260	LRAC	002260	LRBC	Left Rotate with Carry
0016nn	RLMAnn	0006nn	RLMBnn	Left Rotate by n
003300	LSA	002300	LSB	Left Shift by 1
0017nn	SLLAnn	0007nn	SLLBnn	Left Shift by n
000000	NO-OP			No Operation
003040	RRA	002040	RRB	Right Rotate by 1
003060	RRAC	002060	RRBC	Right Rotate with Carry
0014nn	RRMAnn	0004nn	RRMBnn	Right Rotate by n
003100	RSA	002100	RSB	Right Shift by 1
0015nn	RSAnn	0005nn	RSBnn	Right Shift by n
001001	STGT			Set Greater Than Flag
006440	SK=C			Skip if Carry Flag Set
006410	S=GT			Skip if Gt. Than Flag set
004020	SKIP			Unconditional Skip
006040	SKNC			Skip if Carry Not Set
006010	SNGT			Skip if Gt. Than Not Set
005010	SWPA	004010	SWPB	Swap Bits 16-9 with 8-1

COMBINATIONS OF REGISTER INSTRUCTIONS PERMITTED

```
ANEG,  ANO,   SKC,   CLC,   S=GT,  CLA,   CPLA;
APOS,  A=0,   SKNC,  CLC,   SNGT,  CLGT,  CLA,   CPLA;
BNEG,  BNO,   SKC,   CLC,   S=GT,  CLGT,  CLB,   CPLB;
BPOS,  B=0,   SKNC,  CLC,   SNGT,  CLGT,  CLB,   CPLB;
CLA,   CPLA,  CLC,   CMPC,  SKIP,  SWPA,  CLSA,  CPSA,  ESRA;
CLB,   CPLB,  CLC,   CMPC,  SKIP,  SWPB,  CLSB,  CPSB,  ESRB;
CLC,   LSA,   DECA,  AMSB,  ALSB;
CLC,   LSA,   INCA,  AMSB,  ALSB;
CLC,   LRAC,  DECA,  AMSB,  ALSB;
CLC,   LRAC,  INCA,  AMSB,  ALSB;
CLC,   RSA,   DECA,  AMSB,  ALSB;
CLC,   RSA,   INCA,  AMSB,  ALSB;
CLC,   RRAC,  DECA,  AMSB,  ALSB;
CLC,   RRAC,  INCA,  AMSB,  ALSB;
CLC,   LSB,   DECB,  BMSB,  BLSB;
CLC,   LSB,   INCB,  BMSB,  BLSB;
CLC,   LRBC,  DECB,  BMSB,  BLSB;
CLC,   LRBC,  INCB,  BMSB,  BLSB;
CLC,   RSB,   DECB,  BMSB,  BLSB;
CLC,   RSB,   INCB,  BMSB,  BLSB;
CLC,   RRBC,  DECB,  BMSB,  BLSB;
CLC,   RRBC,  INCB,  BMSB,  BLSB;
```

RULES FOR USE

1. No micro-instructions may be used which combine instructions from different sets.

2. Instructions selected from the above sets must be written in the order shown and will be executed in that order.

3. If two (or more) skip functions are combined, the skip will occur only if all conditions are fulfilled. One exception exists: if AMSB and ALSB (or BMSB and BLSB) are both included in the same statement, the skip will occur if either of both conditions are met.

4. Shift and Rotate or Increment and Decrement must not be mixed in the same micro-instruction. The effect of doing so is undefined.

BIT NUMBER AND DECIMAL CHART

Bit no	Decimal value of Bit	Decimal Capacity of all Bits up to & including Bit Shown
1	1	1
2	2	3
3	4	7
4	8	15
5	16	31
6	32	63
7	64	127
8	128	255
9	256	511
10	512	1,023
11	1,024	2,047
12	2,048	4,095
13	4,096	8,191
14	8,192	16,383
15	16,384	32,767
16	32,768	65,535
17	65,536	131,071
18	131,072	262,143
19	262,144	524,287
20	524,288	1,048,575
21	1,048,576	2,097,151
22	2,097,152	4,194,303
23	4,194,304	8,388,607
24	8,388,608	16,777,215
25	16,777,216	33,554,431
26	33,554,432	67,108,863
27	67,108,864	134,217,727
28	134,217,728	268,435,455
29	268,435,456	536,870,911
30	536,870,912	1,073,741,823
31	1,073,741,824	2,147,483,647
32	2,147,483,648	4,294,967,295
33	4,294,967,296	8,589,934,591
34	8,589,934,592	17,179,869,183
35	17,179,869,184	34,359,738,367
36	34,359,738,368	68,719,476,735
37	68,719,476,736	137,438,953,471
38	137,438,953,472	274,877,906,943
39	274,877,906,944	549,755,813,887
40	549,755,813,888	1,099,511,627,775
41	1,099,511,627,776	2,199,023,255,551
42	2,199,023,255,552	4,398,046,511,103
43	4,398,046,511,104	8,796,093,022,207
44	8,796,093,022,208	17,592,186,044,415
45	17,592,186,044,416	35,184,372,088,831
46	35,184,372,088,832	70,368,744,177,663
47	70,368,744,177,664	140,737,488,355,327
48	140,737,488,355,328	281,474,976,710,655

SYSTEM HALTS

The system may halt a Task (by calling the HALT subroutine) for the reasons given below.

ADDRESS

001377 Record Key Error. "A" register contains program back address. In the case of a Direct Access file, this means that the logical record number is outside the file limits.

001376 Subroutine called out of context "A" register contain program back address.

001375 GET AND FETCH file type not supported. "A" register contains program back address (check that the file identifier refers to a Direct Access file).

001374 File Not Found. "A" register contains program back address (File identifier is invalid)

001373 Hashfail on reading overlay module. "A" register contains program back address.

001372 Spool queue not assigned. "A" register contains the back address. The program must be corrected to assign a non-zero spool queue to the route before using the Spooling System.

001371 Print program not found. The printer control program has been presented with a "posting" is cannot process because the program (named in the HALT message) is not entered in the Print Program Directory.

001370 Non-priveleged WRITE to Protected Sector.

001367 Breakpoint out of context. "A" register contains address at which the task has encountered a "breakpoint" but there is no controlling task (See description of Breakpoint handler).

001366 Parameter Validation Error. "A" register contains the back address of the subroutine whose parameters do not conform to the required format.

001365 Data file full. "B" register contains the file identifier.

001364 Index full. "B" register contains the file identifier.

PROCESSOR HALTS/LOOPS

SWITCH REGISTER	TYPE	DESCRIPTION
000021	HALT	CONT. INT if switch is down; MA = SW if switch is down; Mains- fail Raise switch if appropriate, Press CONTINUE to dismiss interrupt. NB the CONT. INT switch is programmed to call the mains-fail routine.
0000104	LOOP	Disk status problem or disk not ready whilst attempting bootstrap; to obtain status in A-Reg STOP and the LOAD and INST. STEP an 01437x instruction.
000146) 000537)	HALT	Hashfail on bootstrapping O.S. into memory; press CONTINUE to retry.
003333	HALT	PARITY if "Parity" light is lit otherwise NO REASON FOR INTERRUPT EXAMINE A-Reg to obtain value of PC-Reg at interrupt. Press CONTINUE to dismiss interrupt. NB system will attempt to continue but not generally possible.
007567	HALT	Insufficient Free-Memory available EXAMINE A-reg to obtain GETMAIN subroutine back-address. If halt occurs during bootstrapping, the Configuration Table must be modified to obtain a successful bootstrap; otherwise press CONTINUE to resume (issuing Task will be halted).

DISK STATUS PROBLEMS

Disk Status Register Contents

1. DD1600/D8000/DD818 drives:

BITS

17	Surface number
16-15	Drive Code
14	Drive Code extension for D8000 Disk Drive
13-11	Auto Sector Count
10	Monitor
9	Seek in progress
8	Seek Error
7	Data Late
6	Address Incorrect
5	Checkword Incorrect
4	Temperature Check
3	Off Line
2	Logic or Write Check
1	Fault Exists

2. DD9600 drive:

BITS

17-16	Drive Code
15-11	Surface number
10-8	Auto Sector Count
7	Program or Command error
6	Address error
5	CRC error
4	Write protected (and attempted)
3	Off line (not up to speed, not connected)
2	Logic fault (hardware or drive fault)
1	Fault Exists

Translation Table

NUMERIC VALUES: DEC	OCT	TOP BYTE	EQUIV ASCII CHAR	AT MEMORY LOC'N	EQUIV PRINT POS'N
0	000	000000	NUL	00/0200	
1	001	000400	SOH	/0201	3600-
2	002	001000	STX	/0202	3600.-
3	003	001400	ETX	/0203	3601-
4	004	002000	EOT	/0204	3601.-
5	005	002400	ENQ	/0205	3602-
6	006	003000	ACK	/0206	3602.-
7	007	003400	BEL	/0207	3603-
8	010	004000	BS	/0210	3603.-
9	011	004400	HT	/0211	3604-
10	012	005000	LF	/0212	3604.-
11	013	005400	VT	/0213	3605-
12	014	006000	FF	/0214	3605.-
13	015	006400	CR	/0215	3606-
14	016	007000	SO	/0216	3606.-
15	017	007400	SI	/0217	3607-
16	020	010000	DLE	/0220	3607.-
17	021	010400	DC1	/0221	3610-
18	022	011000	DC2	/0222	3610.-
19	023	011400	DC3	/0223	3611-
20	024	012000	DC4	/0224	3611.-
21	025	012400	NAK	/0225	3612-
22	026	013000	SYN	/0226	3612.-
23	027	013400	ETB	/0227	3613-
24	030	014000	CAN	/0230	3613.-
25	031	014400	EM	/0231	3614-
26	032	015000	SUB	/0232	3614.-
27	033	015400	ESC	/0233	3615-
28	034	016000	FS	/0234	3615.-
29	035	016400	GS	/0235	3616-
30	036	017000	RS	/0236	3616.-
31	037	017400	US	/0237	3617-
32	040	020000	SP	/0240	3617.-
33	041	020400	!	/0241	3620-
34	042	021000	"	/0242	3620.-
35	043	021400	£	/0243	3621-
36	044	022000	$	/0244	3621.-
37	045	022400	%	/0245	3622-
38	046	023000	&	/0246	3622.-
39	047	023400	'	/0247	3623-
40	050	024000	(	/0250	3623.-
41	051	024400	)	/0251	3624-
42	052	025000	*	/0252	3624.-
43	053	025400	+	/0253	3625-
44	054	026000	,	/0254	3625.-
45	055	026400	-	/0255	3626-
46	056	027000	.	/0256	3626.-
47	057	027400	/	/0257	3627-
48	060	030000	0	/0260	3627.-
49	061	030400	1	/0261	3630-
50	062	031000	2	/0262	3630.-

Translation Table

NUMERIC VALUES: DEC	OCT	TOP BYTE	EQUIV ASCII CHAR	AT MEMORY LOC'N	EQUIV PRINT POS'N
51	063	031400	3	/0263	3631-
52	064	032000	4	/0264	3631.-
53	065	032400	5	/0265	3632-
54	066	033000	6	/0266	3632.-
55	067	033400	7	/0267	3633-
56	070	034000	8	/0270	3633.-
57	071	034400	9	/0271	3634-
58	072	035000	:	/0272	3634.-
59	073	035400	;	/0273	3635-
60	074	036000	<	/0274	3635.-
61	075	036400	=	/0275	3636-
62	076	037000	>	/0276	3636.-
63	077	037400	?	/0277	3637-
64	100	040000	@	00/0302	3637.-
65	101	040400	A	00/1200	3640-
66	102	041000	B	/1201	3640.-
67	103	041400	C	/1202	3641-
68	104	042000	D	/1203	3641.-
69	105	042400	E	/1204	3642-
70	106	043000	F	/1205	3642.-
71	107	043400	G	/1206	3643-
72	110	044000	H	/1207	3643.-
73	111	044400	I	/1210	3644-
74	112	045000	J	/1211	3644.-
75	113	045400	K	/1212	3645-
76	114	046000	L	/1213	3645.-
77	115	046400	M	/1214	3646-
78	116	047000	N	/1215	3646.-
79	117	047400	O	/1216	3647-
80	120	050000	P	/1217	3647.-
81	121	050400	Q	/1220	3650-
82	122	051000	R	/1221	3650.-
83	123	051400	S	/1222	3651-
84	124	052000	T	/1223	3651.-
85	125	052400	U	/1224	3652-
86	126	053000	V	/1225	3652.-
87	127	053400	W	/1226	3653-
88	130	054000	X	/1227	3653.-
89	131	054400	Y	/1230	3654-
90	132	055000	Z	/1231	3654.-
91	133	055400	[		3655-
92	134	056000	\		3655.-
93	135	056400	]		3656-
94	136	057000	^		3656.-
95	137	057400	_	00/0311	3657-
96	140	060000	`		3657.-
97	141	060400	a		3660-
98	142	061000	b		3660.-
99	143	061400	c	00/0312	3661-
100	144	062000	d	/0313	3661.-
101	145	062400	e		3662-

Translation Table

NUMERIC VALUES: DEC	OCT	TOP BYTE	EQUIV ASCII CHAR	AT MEMORY LOC'N	EQUIV PRINT POS'N
102	146	063000	f		3662.-1
103	147	063400	g		3663-
104	150	064000	h		3663.-1
105	151	064400	i		3664-
106	152	065000	j		3664.-1
107	153	065400	k		3665-
108	154	066000	l		3665.-1
109	155	066400	m		3666-
110	156	067000	n		3666.-1
111	157	067400	o		3667-
112	160	070000	p		3667.-1
113	161	070400	q		3670-
114	162	071000	r		3670.-1
115	163	071400	s		3671-
116	164	072000	t		3671.-1
117	165	072400	u		3672-
118	166	073000	v		3672.-1
119	167	073400	w		3673-
120	170	074000	x		3673.-1
121	171	074400	y		3674-
122	172	075000	z		3674.-1
123	173	075400	{		3675-
124	174	076000	|		3675.-1
125	175	076400	}		3676-
126	176	077000	~		3676.-1
127	177	077400	DEL	00/0375	3677-

Masks and Constants

ADDRESS	VALUE
00/0200	Single word Zero
/0201	Bit 1
/0202	Bit 2
/0204	Bit 3
/0210	Bit 4
/0220	Bit 5
/0240	Bit 6
/0302	Bit 7
/0316	Bit 8
/0323	Bit 9
/0332	Bit 10
/0341	Bit 11
/0344	Bit 12
/0347	Bit 13
/0352	Bit 14
/0355	Bit 15
/0356	Bit 16
/0377	Bit 17

Masks and Constants

ADDRESS	VALUE
00/1752	000377
/1753	177400
00/0375	000177
00/0376	377777
00/0374	'SP SP'
00/0200-277	0-63
00/0300,301	Double word Zero
/0302	64
/0303	65
/0304	70
/0305	75
/0306	80
/0307	85
/0310	90
/0311	95
/0312	99
/0313	100
/0314	120
/0315	125
/0316	128
/0317	150
/0320	192
/0321	200
/0322	250
/0323	256
/0324	300
/0325	320
/0326	384
/0327	400
/0330	448
/0331	500
/0332	512
/0333	600
/0334	700
/0335	800
/0336	900
00/0337	999
/0340	1000
/0341	1024
/0342	1500
/0343	2000
/0344	2048
/0345	3000
/0346	4000
/0347	4096
/0350	5000
/0351	6000
/0352	8192
/0353	9999
/0354	10000
/0355	16384
/0356	32768

Masks and Constants

ADDRESS	VALUE
/0357	50000
/0360	60000
/0361	65535
/0362, 363	100000
/0364, 365	1000000
/0366, 367	10000000
/0370, 371	99999999

Standard Memory/Register Values

```
00/0040     = Current Task Number
00/0077    -> Current System Date
00/0072    -> Master Buffer (3200- resolved), current Task
00/0067    -> Spool Buffer  (3400- resolved), current Task
00/0066    -> Input or Print Buffer (3600-)
00/0073    -> Current File Table
00/0074    -> Task Control Area
00/0060     = Max. Plain Paper Queue, current System
00/0057     = Max.  (Deletions) Print Queue  Number
00/1300     = System Password 1
00/1305     =    ""   Password 2
00/1312     =    ""   Password 3
00/1317     =    ""   Password 4
00/0045     = Number of characters input(after GET)

             After FETCH:
00/0151    ->Record (1st word) WITHIN buffer
00/0152     =Logical record length (words)

             After ISAM Fetch:
00/0153     =Direct Access record no.  from
             relevant Index or Data file.

             Before UNSPOOL:
3577-      = Next Spool record no. in chain

             After Fetch Overlay:
00/0144    ->1st word of overlay

             After Fetch & Link:
00/0155    ->word following P2 of original Fetch

             After Load ISAM:
00/0153     =new record no. within Data file
00/0154     =Direct Access record no.  from
             relevant Index file.

             After GET - Split:
A-reg= -1    when NO  (minus)
A-reg=  0    when YES (plus)

             After Convert ASCII to Binary:
A-reg= B17   Number to large
B-reg      ->next character after detected end of field
A-reg       =1st  non-numeric character if  Variable  Length
             Field option

             After Convert to Metacode:
A-reg not 0 Unconvertable character found

             After Convert -VE to +VE:
A-reg=0      Number already negative

             After Convert +VE to -VE:
A-reg=0      Number already positive
```

Standard Memory/Register Values

After Move & Pad:
A-reg not 0 Source string truncated

After Multiply by 10:
A-reg =Overflow (if any)

During Name & Address Processor:
A-reg =Call number of exit routine

After Convert Input to Binary:
00/0043 =Non numeric field terminator

After Specify Escape Point:
A-reg =Address of previous Escape point

After Assign Unspool Queue:
A-reg =Previous Unspool queue number

GET - Parameters and Control Word Formats

GET - CHARACTER MODE

Parameter Block

P0	Address of ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0.
P1	CONTROL WORD	
P2	Byte Address of Prompt Message	Variable length ASCII character string terminated by NUL.
P3	(Byte) Address of TARGET AREA	Optional; required only when 'Convert to Metacode' or 'Move and Pad' requested.

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7  6  5  4  3  2  1
  1  0    |  |  |    |  |  0   |  0  <No. of chars. input>
            |  |  |    |  |      1 if escape reqd.
            |  |  |    |  1 if special ETX reqd.
            |  |  |    1 if lower case retained
            |  |  1 if no flashback
            |  1 if move and pad     ) If both set, then input
            1 if convert to metacode) is space filled and
                                      converted to M/code
```

15 Set if Input is to be converted to Metacode. The Metacode is stored at the address specified by P3. Byte addressing is not allowed in this case.

14 Set if input is to be moved to the byte address specified by P3 and if short, space filled to the maximum length specified in bits 7-1. Only the target area is space filled, not the input buffer.

13 Set if input is not to be flashed back (echoed) to the Task's output display. (Used to preserve password security).

12 Set if any lower case alphabetic input is not to be converted to upper case when stored in the input buffer.

11 If set then where the Operator's input produces an error, NO error text is displayed and the Bell only is sounded; upon receipt of an ETX the Operator's input so far is removed leaving the cursor in its original start position. This allows complex screen layouts to remain intact. Note that the use of the CARRIAGE RETURN key will destroy the effect.

9 Set if the ESCAPE address specified in P0 is to override that most recently specified by the SPECIFY ESCAPE POINT subroutine.

7-1 The terminator (accept key) is not counted when determining the length.

NB To prohibit Escape P0 = 001405

GET = BINARY MODE

Parameter Block

P0 Address of ESCAPE POINT - Optional; control word bit 9 set indicates presence of P0

P1 CONTROL WORD

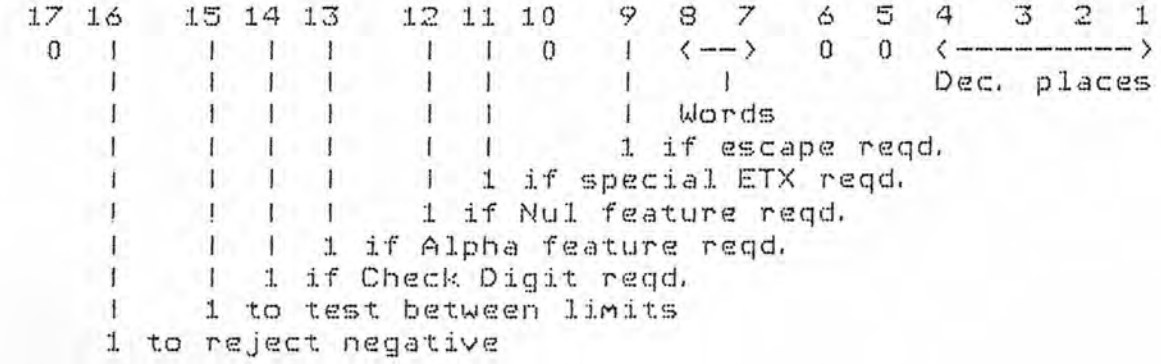
P2 Byte Address of Prompt - Variable Length ASCII character string terminated by NUL

P3 Address for Binary

P4 Address of MINIMUM VALUE)- Required only when testing between limits (for positive or negative numbers).

P5 Address of MAXIMUM VALUE)

Control Word



16 Set if Negative Input to be rejected

15 Set if Input is to be tested between limits. P4 and P5 must point to the minimum and maximum values respectively. These values must have the Word Length specified in Control Word Bits 7-8; P5 must not be less than P4.

14 Set if the Input must include the correct check digit.

13 ALPHA FEATURE. Set if input which starts with a non numeric character is not to result in ERROR. The calling program may detect this occurance by testing the A register on return; if zero then the input was numeric (with correct check digit if applicable), otherwise the A register contains the first input character in ASCII in the bottom byte (the top byte is NUL) and the Binary Target will be unchanged. If there was no input (other than ACCEPT) this is interpreted as non-numeric and the A register sign bit 17 is set, unless the NUL feature is also specified in which case the A register will be zero. If the input was purely numeric when a check digit was required then this is interpreted as non-numeric and the A register bit 16 is set.

12 NUL FEATURE. Set if input consisting of ACCEPT only is to leave the Binary Target unchanged. The Target will, however, still be subjected to any tests specified by Control Word bits 15 and 16. If bit 12 is not set, input of ACCEPT only will be interpreted as input of the value zero unless the Alpha feature is active (see above).

11 If set then where the Operator's input produces an error, no error text is displayed and the Bell only will be sounded; upon receipt of an ETX the Operator's input so far is removed leaving the cursor in its original start position. This allows complex screen layouts to remain intact. Note that the use of the CARRIAGE RETURN key will destroy this effect.

8-7 Word length; if zero a default of 1 is assumed.

4-1 Free Format is allowed on input; any necessary adjustment to the precision is carried out automatically (excess digits are truncated).

GET = FETCH MODE

Parameter Block

P0	Address at ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	Byte Address of Prompt Message	Variable Length ASCII string terminated by NUL
P3	Address of KEY AREA	Direct access; single word binary field. ISAM; ASCII key string preceded by single word work-space.
P4	Address of RECORD EXTRACT AREA	Set zero if extraction is not required.

Control Word

```
 17 16   15 14 13   12 11 10   9  8  7   6  5  4   3  2  1
  0  |    |  |  |    |  |  1   |  <-------File ID.-------->
     |    |  |  |    |  |      1 if escape reqd.
     |    |  |  |    |  1 if special ETX reqd./GET next
     |    |  |  |    |                    record (ISAM)
     |    |  |  |    1 if Nul feature reqd./special ETX
     |    |  |  |                         reqd. (ISAM)
     |    |  | 1 if Alpha feature reqd.
     |    |  1 if Check Digit reqd./Fetch Index (ISAM)
     |    1 if Test option not reqd.
     1 to reject loaded records
```

16 Normally set 0 to reject records not loaded. This bit allows file loading programs to handle the converse situation; it is ignored if bit 15 is set.

15 Normally set 0 to indicate that direct-access records without the record number in the first word are to be regarded as "not loaded". This bit allows files that do not follow this convention to be handled by "GET".

14 For Direct access files, set if input must be suffixed by the correct check digit.
For ISAM files, allow the Secondary Index record to be FETCHED instead of Data record.

13 Operates for Direct Access files as described under "GET" Binary Mode.
For Indexed Sequential files, an input of a single character will be taken as "Alpha" input.

12 For Direct Access files, operates as described under "GET" Binary Mode.
For ISAM files, if input is restarted by using the ETX key then the previous input is suppressed and the previous prompt is not re-displayed.

11 For Direct Access files, if input is restarted (by using ETX key) previous input is suppressed, prompt is NOT re-output.
For ISAM files input of ACCEPT only will "Fetch Next" from the file. If no Indexed Sequential Fetch preceeds this option, the single word work-space allocated with the key area MUST be set to zero.

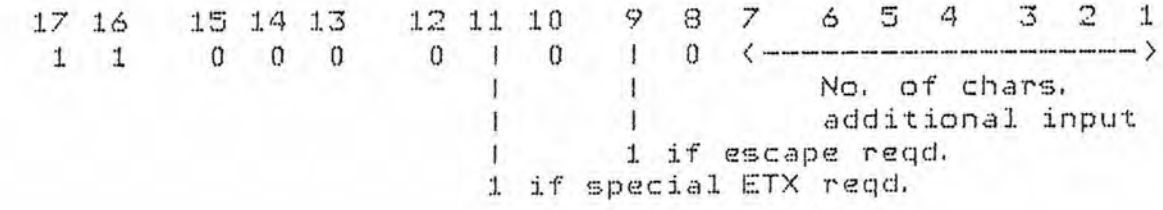
SPLIT

JSBR IZ 1640 037640
P1 = Address of Control Word in Parameter Block

Parameter BLock

P0	Address of ESCAPE POINT	Optional; control word bit 9 set indicates presence of P0
P1	CONTROL WORD	
P2	Byte Address of Prompt Message	Variable Length ASCII character string terminated by NUL.

Control Word



Miscellaneous Control Words

EXTRACT FROM FILE CONTROL BLOCK

JSBR IZ 1700 037700
P1 = Word Number, File Identifier

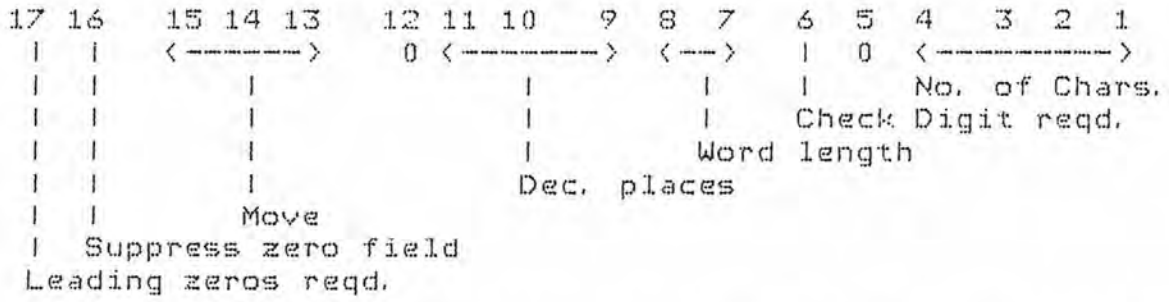
Parameter Format

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	<-Word No. in FCB->				<-------File ID.------->							

CONVERT TO ASCII

JSBR IZ 1765 037765
P1 = Control Word
P2 = Address of Binary Source
P3 = Byte Address of ASCII target

Control Word



17 If 0, preceding zeros are replaced by spaces in the target string.

16 If set, leave target unchanged if binary source is zero.

15-13 Number of figures after the decimal point in the Binary source.

11-9 Number of figures after the decimal point in the Target (if 0, the decimal point is itself omitted from the string).

8-7 Word Length of the Binary Number.

6 If set, the routine will calculate the check digit corresponding to the number and insert it in the last character position.

4-1 Total length of the ASCII string, in characters, including the space, hyphen or check digit and (if non-integer) the decimal point.

FILE STATUS BLOCK PROCESSOR

JSBR IZ 1665 037665
P1 = Address of Control block

This routine deals with processing of the standard FSB, which is:

Word 1 - Count of live records
Word 2 - Count of chained records
Word 3 - First free chained record I/D
Word 4 - Update flag

Control block layout:

P0 = 'File Full' return address or Zero
P1 = Options, File I/D
P2 = Address of Key Area
P3 = Address of 'Extract' Area

Options:

```
17 16  15 14 13  12 11 10  9  8  7   6  5  4   3  2  1
 |  |   |  |  |   | <---> |  <-------File I/D------->
 |  |   |  |  |   |   |    Protection Overide
 |  |   |  |  |   |  Reserved
 |  |   |  |  |  Free chain is maintained via the FIRST
 |  |   |  |  |  word of the specified file (default is
 |  |   |  |  |  using the LAST word)
 |  |   |  |  Insert into the First word of the Extract
 |  |   |  |  area the Record I/D of the Next Free Record
 |  |   |  |  before writing new record (used with Bit 14)
 |  |   |  Release for use Next Free Record, overwriting
 |  |   |  with contents of Extract area (P3). Record I/D
 |  |   |  will be returned in the Key area (P2)
 |  |   Return to Free Chain record nominated by the Key
 |  |   area. Record will be cleared except for Free Chain
 |  |   Pointer (First or Last word)
 |  Leave File Update Flag set on return to Calling Program
 Clear Update Flag only. Required if Bit 15 option has been
 previously used
```

OP/BREAKPOINT HANDLER COMMAND SUMMARY

COMMANDS		FUNCTION	
OP	B/PT	OP	BREAKPOINT HANDLER
777A	777A	Examine abs. loc .	Set B/pt @ abs. loc.
(777)O	(777)O	Examine offset loc.	Set B/pt @ offset
(777)I	---	Examine via abs. loc.	
(777)G	---	Examine via offset	
(77)P	(77)P	Change Base	Change Base
(777)R	(99)R	Read Program Overlay	Remove B/pt(s)
ESC	ESC	Escape to PROGRAM?	Escape to OP
SPACE	SPACE	Display ASCII	Restart B/pt
*	*	Display Metacode	Re-display B/pt
(999)S	(999)S	Decimal input-Single	Step/Skip B/pt
(999)D	(99)D	"" "" -Double	Dispaly B/pt(s)
(999)T	---	"" "" -Triple	
(999)cU	---	Examine next loc(s).	
(999)cY	---	Re-examine loc.	
(999)cH	---	Examine previous loc	
(777)LF	LF	Enter Value	Restart B/pt
(777)E+	E+	"" ""	"" ""
W	---	Write Program O'lay	
L	(99)L	ASCII input	Loop B/pt
M	(777)M	Metacode input	Module Intercept
C	C	Switch Display	Switch Display
B	---	Enter B/PT Handler	
%	---	Display as Date	
(777)F	---	Examine F.C.B.	
ETX	ETX	Cancel ERROR	Cancel ERROR
CMND	---	Enter Command Mode	
cV	---	"" "" ""	
E-	---	Enter Value + B17	
(9)@	(9)@	Change memory bank	Change memory bank
---	(99)J		B/pt thru JUMP/JSBR
---	H		Restart from HALT

AVAILABLE HARDWARE PRINT OPTIONS

The following printer types allow variable form length, by either Paper-tape loops or via the computer interface (i.e. using the "set queue" and "line up" facilities available; see the Utilities and Commands sections), and allow Vertical Tabs:

1. DRI, DRIR, CENT, TALLY2
2. HSLINE
3. TALLY1
4. MT240, MT480, MT930, MICROL, L/S310

In addition the following printers allow various other options:

1. MT240
   a) 6 or 8 lines per inch
   b) Characters per inch: 10, 12, 13.3, 15, 17.1
* c) Double width characters: (SO) = on, (SI) = off
* d) Audible alarm: (BEL)
* e) Backspace: (BS)

2. MT480
   a) Lines per inch: 6, 8, 12, 3
   b) Characters per inch: 10, 12, 13.3, 15, 17.1
* c) Double width characters: (SO) = on, (SI) = off
* d) Audible alarm: (BEL)
* e) Micro line feeds: (Q) followed by parameters
* f) Underline: (numeric 1) = on, (numeric 0) = off

3. MICROL
   a) 6 or 8 lines per inch
   b) 10 or 16.5 characters per inch
* c) Double width characters: (SO) = on, (SI) = off
* d) Graphics mode: (numeric 1) = on, (numeric 2) = off

4. MT930
   a) 6 or 8 lines per inch
   b) Characters per inch: 10, 12, 15, 16.6
* c) Double width characters: (SO) = on,(SI) = off
* d) Audible alarm: (BEL)
* e) Backspace: (BS)
* f) One half forward line feed: (U)
* g) One reverse line feed: (LF)
* h) One half reverse line feed: (D)
* i) Near-letter-quality print: (numeric 1)
* j) Draft quality print: (numeric 0)
*+ k) Print in secondary colour: (A)
*+ l) Print in primary colour: (B)
* m) Underscore: (E) = on, (R) = off
* n) Underline: (_) = on, (^) = off
* o) Bold print: (O) = on, (&) or end of line = off
* p) Shadow print: (W) = on, (&) or end of line = off
*+ q) Shadow dual colour: (F) = on,
(&) or end of line = off

Notes

* denotes that the specified code is placed within the print buffer. It should be noted that these codes will only be acted upon if the relevent byte containing the code has Bit 8 set. Also care should be taken when designing the print line as although these codes take space within the buffer, no print position is used when acted upon.

+ denotes that a hardware printer option is required before the function is available.

Interested parties should obtain the relevant printer manual before attempting use of the above options.

AVAILABLE HARDWARE I/O STATION DISPLAY OPTIONS

DUMP PRINTERS

This facility allows the connection of a printer directly to an I/O Station and allow printing of data, with or without simultaneous display, under the control of the applications program.

Three functions are available:

Octal code	Result
372	Disable Auxiliary (Printer) port.
373	Enable port WITHOUT display.
374	Enable port WITH display.

This facility is available to the CT760 and CT800 range of I/O Stations, using the MICROL printer (80 or 132 column).

CASH TILL

An unintelligent cash till is available with the CT760 and CT800 range of I/O Stations, allowing the cash draw to be opened under application program control. The code required to do this is switch selectable on the till, the standard code being (DEL). This option can be used in conjunction with dump printers (see above) if required.

AVAILABLE VIDEO ATTRIBUTES

As far as is possible, the following attributes are available throughout the I/O Station range.

	Code	Result
+	(SO)	Flashing Text ON
+	(SI)	Flashing Text OFF
	(CAN)	Clear Screen
	(CR)	Newline
	(SYN)	Clear line
	(BS)	Backspace one character position
	(EM)	Home cursor
	(SUB)	Cursor up one line
	(BEL)	Sound audible alarm

The following attributes are available to the CT800 range of I/O stations only.

Code	Result
(Octal 375)	Inverse Video ON
(Octal 376)	Underline ON
(Octal 377)	Send to 25th line
(SI)	Inverse/Underline OFF

Note

\+ Some screens have Inverse Video instead of Flashing text.