

IN.FL.HEADER (I,S)

P1 is the stream number. This procedure accesses the global information associated with the stream and stores it in a standard area in an area described by P2. If P2 is zero or negative a standard area in the I/O control segment is used.

The user is faulted if the input is not of Flocoder type or the header is missing from the input.

OUT.FL.HEADER (I,S)

Analogous to IN.FL.HEADER. In this case P1 is an output stream. The user is faulted if S or its default does not describe valid global information.

TEXT.TO.FLOC (I,I,S)

This reads text in standard Flocoder form (see Software Tools Manual) and outputs the equivalent information on a Flocoder type stream. P1 is the input stream, P2 the output stream and P3 describes an area containing global information.

The procedure has the following effects:-

- a) The text describes a chart.
- The text is processed using the global information if it exists. If no global information exists the user is faulted.
- b) The text describes the global information.
- If the global information is valid the user is faulted, otherwise the text is transferred to output in Flocoder form and the corresponding global information is created.

Reading from a Flocoder type stream using INSYM takes place in one of two modes. These yield either the original text input (or an equivalent form) or the code of the associated program. These may be thought of as the original TEXT or the result of FLOCODING the file. Consequently, INSYM can operate in TEXT mode or FLOC mode. This mode setting always results from a call of a Flocoder file manipulation procedure.

With the exception of CODECHART all the file manipulation procedures which have an input stream leave that stream in TEXT mode.

SET.TO.CHT (I,S)

This specifies that stream P1 is to be interpreted by INSYM in FLOC mode to yield a 'program' whose master chart is given by the descriptor of P2.

The user is faulted if the stream is not of Flocoder type or if the chart does not exist.

N.B.

The only action of this procedure is to position the input pointer before the required chart and set the mode to FL.

DRAWCHART (I,I,I,S)

This procedure will draw a single chart specified in standard Flocoder form.

Parameters:-

P1 Number of an input stream positioned before a chart description
P2 Number of an output stream for the diagram
P3 Size of diagram required - permitted values are 1 to 9
The boxes will normally be numbered. Numbering is suppressed by adding 10 to the size in P3.
P4 This is a descriptor for the JOBTXT if any. If negative or zero then no JOBTXT is output.

e.g.,

To draw from stream 1, output directives on stream 3. The diagram has no numbers, is SIZE 2 and has no JOBTXT.

DRAWCHART (1,3,12,-1)

FIEDITOR (1,1,1,1)

The parameter allocation is as for EDIT and EDITOR, but streams

P1 and P2 will be Flocoder type streams.

This editor is a Flocoder chart and/or statement editor.

There are two basic positioning commands C (copy) and S (skip).

They take a parameter which is either

a string of text : meaning advance to the TITLE of this chart
an integer : meaning advance over this many charts or

part charts. The pointer will be positioned at the TITLE of the next chart, if any.

These positioning commands may also be used at the statement level.

They then take a parameter of the form:-

- letter integer
- mnemonic
- integer

In the first case the integer is a box number. The result of such commands is to position the pointer immediately before a:-

- declaration (D)
- translation (T)
- column statement (C)
- row statement (R)
- flow statement (F)

concerning the box number specified.

The second case, i.e., mnemonics, permit positioning before statements not directly associated with box numbers. The mnemonics are:-

E	meaning	END
ST	meaning	START declaration
FIN	meaning	FINISH declaration
CJ	meaning	CJUMP statement
JP	meaning	JUMP statement
LT	meaning	LABEL statement
LF	meaning	LABDEF statement
ET	meaning	ENDTEXT statement

The third form advances the pointers over the specified number of statements or part statements.
Positioning within a statement is done by context. There are two commands:-

A <delim> <text> <delim> These position the pointer After
B <delim> <text> <delim> or Before the specified text.

Changes can be made using:-

I <delim> <text> <delim> These Insert or Delete the
D <delim> <text> <delim> specified text.

The Insert has its effect at the current position. The Delete has its effect on the first occurrence of the text in the statement.

Note

It is proposed that the specification of exit captions on boxes should occur in FLOW statements and not in box declarations as at present. This will make editing easier.

Format Of Floccoder Files

DIRECTORY

LIB	GLOBAL	HEADER	1st
USAGE	POINTERS	TITLES	INFORMATION
- 12 - < > - 1536 - < > - 3072 - < > - 572 - < > - 2 - < > - 2			
A			
START OF TEXT (5192)			

All sizes are in bytes.

A Pointer Entry Has The Following Format

START ADDRESS OF	CODED	START ADDR OF CODE
CHART (IN BYTES)	ORDER	REL TO START OF
CHART (IN BYTES)		CHART (IN BYTES)
1 < - - - 23 - - - < > - 8 - - - < > - 16 - - - < >		

All sizes are in bits.

Format of the Global Information

TEXT FOR	GLOBAL	EDIT SYMS	GENERATOR	JUMPS ETC
< - - - 8 - - - < > - 2 - - - < > - 4 - - - < > - 4 - - - < >				

The language data has 2 bytes per language element i.e., Label, Jump Cjump, Labelled. These contain the length of text appearing in the text area for the leading and trailing part of each.

ID contains the length of text associated with ENDTEXT. V is verification byte. It is 255 for a valid header.

Communication Area Format

This is set up by IN.FL.HEADER or TEXT.TO.FLOC.

LABEL	JUMP	CJUMP	LABDEF	ENDTEXT	GEN	TEXT
< - - - - - >						
LANGUAGE DATA - - - - - >						
DESCRIPTORS						
1 PER WORD						
< EDIT SYMS >						

8 DESCRIPTORS

The verify word is a 64 bit word (48 bit on 1900) containing verification information. The descriptors all describe text in the following text area. The label generator is a 4 byte object, and occupies the bottom 4 bytes of the 64 bit (48 bit) word.

I/O FAULT NUMBERS

1	FAULTY PARAMETER
2	STREAM NOT DEFINED
3	INPUT ENDED
4	INPUT STREAM IS NOT A 'SET' INPUT
5	HEADER NOT DEFINED
6	OUTPUT STREAM NOT DEFINED
7	WRONG TYPE OF INPUT STREAM
8	UNABLE TO PASS SEGMENT
9	STREAM ALREADY DEFINED
10	ILLEGAL SYMBOL STARTING NUMBER
11	FILE STORE LIMIT EXCEEDED
12	MESSAGE FAULT
13	PROCESS FAULT
14	SEGMENT FAULT
15	FILE FAULT
16	FAULT IN PROGRAM CONTROL SEQUENCE
17	FAULT DETECTED BY PROCEDURE ENTERED FROM PROGRAM
18	CONTROL SEQUENCE
19	STREAM NOT DEFINED
20	OUTPUT EXCEEDED
21	ATTEMPT TO BACKSPACE POINTER FROM START OF BUFFER
22	INPUT STREAM NOT DEFINED
23	STREAM ALREADY DEFINED
26	CANNOT BEND UNBUFFERED I/O
27	REPLY TO UNDEFINED INPUT
28	MESSAGE FAULT
29	PROCESS FAULT
30	SEGMENT FAULT
31	FILE FAULT
32	DISC FAULT
65	SYSTEM PERFORMANCE MONITOR FAULT

ALGOL-LIKE INPUT/OUTPUT

- 33 STRING PARAMETER IS ZERO LENGTH
- 34 ILLEGAL CHAR BETWEEN EXPONENT SIGN AND EXPONENT
- 35 LEGAL NON-NUMERIC CHAR OCCURS MORE THAN ONCE IN NUMBER
- 36 NUMBER NOT COMPLETE OR IN WRONG FORMAT
- 37 ILLEGAL NUMBER TERMINATOR OR NUMBER IN WRONG FORMAT
- 38 STRING INVALID AS PARAMETER OF PROCEDURE CODE
- 39 INVALID LAYOUT-EDITING COMMAND IN STRING

FLOC INPUT/OUTPUT

- 50 CHART NOT FOUND
- 51 INVALID FLOC HEADER
- 52 INVALID COMMUNICATION AREA
- 53 FLOC DIRECTORY OVERFLOW - O/P STREAM CONTAINS USELESS INFORMATION

- 1 Segment Already Exists
- 2 Segment Limit exceeded
- 3 Illegal segment number specified
- 4 Segment shared (in a context where a shared segment is not allowed)
- 5 Illegal access requested
- 6 Specified segment does not exist
- 7 System has insufficient resources to create segment at this time
- 8 Illegal size
- 9 User store limit exceeded

Segment Faults

Executive Procedure Faults

organisational commands is given below.

fault indications in PWO. A list of the faults signaled by the as 64-bit variables. By convention all library procedures return when used as 32-bit variables and PWO, PWW2, PWW4, . . . when used to the library procedures. These have the names: PWO, PW1, PW2, . . . locations set aside at the beginning of segment 0 which are available In addition to the formal parameters there are a number of of the 1900 conventions see the Appendix.

of the 1900 system is different. For a more detailed description though the detail of the calling and parameter passing mechanism environment. The descriptions are for both the M5 and 1900 systems, mode and will permit a process to organise and manipulate its This chapter describes the procedures which run in executive

Introduction

Chapter 5 The Organisational Commands

- store used
- 39 User has sub-users, jobs in the system, or file
 - 38 Authorised User List full
 - 37 Illegal parameter in user management
 - 36 User overdrawn on output
 - 35 User overcommitted
 - 34 Priority out of range
 - 33 Process name does not exist
 - 32 Process name already exists
 - 31 Index invalid (Read parameter)
 - 30 New supervisor invalid
 - 29 Attempt to free process which is not suspended
 - 28 Attempt to suspend suspended process
 - 27 Attempt to activate active process
 - 26 Attempt to deactivate inactive process
 - 25 Invalid password
 - 24 Invalid user name
 - 23 process at this time
 - 22 System has insufficient resources to create
 - 21 SPN empty or terminating
 - 20 not supervisor
 - 21 Process Identifier Incorrect, or current process is
 - 20 SPN Invalid

Process Faults

- 14 No message buffers
- 13 No messages on this channel (Read Message)
- 12 Illegal reply channel
- 11 Channel number out of range/channel closed
- 10 Faulty Parameters Descriptor in send/read message

Message Faults

File Command Faults

-40	Illegal user identifier or password
-41	File store limit exceeded (on Open/Obtain file)
-42	File name already exists
-43	File name does not exist
-44	File Open/Shared by someone else
-45	Unauthorized access requested
-46	This segment is already a file
-47	No write permission
-48	Segment access incompatible with requested file access
-49	Unable to open file into segment specified
-50	Number of files held limit exceeded
-51	Insufficient wait time specified (giver not owner)
-52	File has been deleted
-53	System file table full
-54	Recipients file store/number of files held limit exceeded
-55	No segment (on file)
-56	Directory full
-57	Store allowance exceeded
-58	

System Performance Monitor Faults

-60	System Performance Monitor not reserved
-61	Faulty Parameter

MUS Disc Command Faults

-64	Bands Isolated
-65	Invalid Disc Addr
-66	Process not authorised to update Common Segment
-67	Segment too large
-68	Common Segment not defined in SS
-69	Invalid Initial block for SS

- 5.8.74/3

1) PROC SPEC CREATE, PROCESS (u64, u64, u64, i32, i32, i32, i32, i32)

This command creates a process on behalf of the specified user, providing the password check is satisfied and he is not overdrawn on output. However, if the user is overcommitted (that is, the sum of the COST.LIMIT's for processes which he currently has in the system is less than his balance) the process will only be created providing the priority of the process is 15. Note: only privileged processes can create processes at priorities 14 and 15. Names and passwords are right justified and zero filled.

Parameters:-

P1 The PROCESS.NAME for the created process
 P2 The NAME of the user, on whose behalf the process is created

P3 The PASSWORD

P4 The SEGMENT LIMIT, specifying the maximum number

of segments which the process may have.

P5 The CPU.LIMIT, specifying the maximum CPU time

allotted to the process in instruction count

interrupts, i.e., every 64K instructions or

1/200 seconds in MVS. The units in the 1900 are

seconds. MVS treats this number as modulo 2^{24} ,

the 1900 treats it modulo 2^{15} .

P6 The COST.LIMIT for the process during its existence

in the system. This quantity is added into the

commitments of the user on whose behalf the process

is created. It is treated as modulo 2^{24} by MVS.

At P15 1 Cost Unit = 1 sec 1900 time

= 1/50 sec MVS time.

P7 The WAIT.LIMIT, specifying the time in seconds

for which the process is allowed to be in a waiting

state. This will be treated as modulo 2^{15} .

p8 The SEGMENT NUMBER of the segment to be given to the

process as its segment zero. This segment must be a scratch segment (i.e., not a file).

After successful execution of the command, this

segment will have been released from the issuing

process. The newly created process will have read

and write access to the segment.

p9 The maximum PRIORITY level at which P1 is to run.

This is an integer in the range 1 - 15, with 15 as

the lowest priority.

p10 The supervisor may arrange to be informed when the

process terminates by setting this parameter to the

TERMINATE CHANNEL number to which a message will be

sent. P10 is set to -1 if the supervisor does not

wish to be informed.

Output parameters:-

In PW1 is stored the SPN for the created process. In PW2

the process identifier allocated to this process.

The new process will have the creating process as its

supervisor. The supervisor may now perform the other process

organisational commands on it using the SPN, and may send

messages to it using the SPN and process identifier parameters

given by the create process command. When first created, the

process will be inactive and will have only one segment, i.e.,

segment zero. The process's channel are initially open to the

supervisor.

Any command which may be executed on a process by its

supervisor may also be issued by a privileged process, but the

privileged process must also supply the Process Identifier of

the process on which the command is to operate.

The initial value of control for the created process

will be set to a common initialization procedure. See the

appropriate library specification for a specification of this

procedure.

2) PROC SPEC INITIALISE.CHANNEL (I32,I32,I32,I32,I32,I32)

This command may be used by a supervisor or privileged process to initialise the channel status of a channel in one of its created processes. The operation may only be performed on an inactive process.

Parameters:-

P1 The SPN of the process whose channel is to be initialised.
 P2 The Process Identifier of the process.
 P3, P4, P5, P6 The channel number, type, SPN and count as for SET.CHANNEL.STATUS.

3) PROC SPMC SUSPEND.PROCESS (I32,I32)

This command may be used by a supervisor or privileged process to suspend one of its processes until the supervisor issues a FREE.PROCESS command.

Parameter:-

P1

The SPM of the process to be suspended.

P2

The Process Identifier of the process.

4) PROC SPEC FREE.PROCESS (I32,I32)

This command allows a supervisor or privileged process to free one of its suspended processes.

Parameters:-

P1	The SPN of the process to be freed.
P2	The Process Identifier of the process.

5) PROC SPEC TERMINATE.PROCESS (I32,I32,I32)

This command permits a process to terminate itself, or a supervisor to terminate one of its processes. A job record

containing the job statistics will be sent to the accounting

process.

Parameters:-

P1

The SPN of the process to be terminated. If this parameter is negative then the process wishes to

terminate itself.

P2

The Process Identifier for the process.

P3

This is a 32-bit word which indicates a reason for termination (REASON.CODE). If the supervisor wished to be informed when the process terminated then the 'Reason Code' will be sent to the supervisor on the 'Terminate Channel'. Negative values of the Reason Code will indicate that the process will have been terminated involuntarily.

6) PROC SPEC SWOP.SUPERVISOR (I32,I32)

This command is generated by the supervisor of a process which requires a supervisor change.

Parameters:-

- P1 The SPN of a process whose supervisor is to be changed.
- P2 The SPN of the new supervisor.

7) PROC SPEC READ.STATUS (132,132)

This command allows a supervisor or privileged process to read the status word in the SPT of one of its processes.

Parameters:-

P1 The SPN of the process to be examined.
P2 The PID of the process to be examined.

Output parameter:-

In PWT is stored the status of P1, interpreted as follows:-

Digit 0 -	7	SPARE
8 -	15	Channel Mask (set by WAIT command)
16		Empty
17		Active
18		On Execute List
19 -	21	Spare
22		Halted
23		Suspended
24		Waiting
25		Terminating
26 -	31	Spare

8) PRGC SPEC READ.PARAMETER (I32,I32,I32)

This procedure allows the supervisor of a process or a privileged process to read a specified parameter from the Process Register Block of the process.

Parameters:-

- P1 The SPN of the process. If negative, the process wishes to read from its own Process Register Block.
- P2 The PID of the process.
- P3 The index of the parameter table in the process register block. The parameter table is indexed as follows:-

0	USERID
1	SPN
2	SUPERVISOR SPN
3	PROCESS PRIORITY NUMBER
4	INTERRUPT STATUS
5	INTERRUPT MASK
6	SEGMENT LIMIT
7	SEGMENTS USED
8	WAIT TIME BALANCE
9	WAIT TIME LIMIT
10	FIRST TRAP
11	ALL TRAPS
12	TERMINATE CHANNEL
13	EXECUTIVE CHARGE
14	POSSIBLE COST
15	COST LIMIT
16	COST BALANCE
17	CPU LIMIT
18	CPU BALANCE (EXCLUDING THE AMOUNT IN LOCAL TIMER)
19	LOCAL TIMER
20	PROCESS IDENTIFIER
21	SUPERVISOR PROCESS IDENTIFIER
22	CREATE TIME

Output parameter:-

The parameter required is stored in Pw1 (and Pw2 if necessary).

This command may be used by a process to await the arrival of a message.

Parameters:-

P1

The CHANNEL.MASK used to indicate the channels on which messages are expected. This consists of 8 bits, a '1' in bit position 1 (where bit 0 is the least significant) indicates that a message on channel 1 should not awaken the process.

P2

The TIME.LIMIT in units of a second (modulo 2¹⁵). If a message does not arrive within the specified time, the process will be awakened. This 'overdue' case may be detected by examining the interrupt status. If the time limit parameter exceeds the process wait balance then it is set equal to the wait balance.

During a period of waiting, all message interrupts will be inhibited. Outstanding interrupts may, however, take effect immediately after leaving 'WAIT'. (Normally, control is returned to the instruction after the WAIT). When the process is awakened, the wait balance is reduced by the wait time used. If the balance becomes zero, the process is forced to enter trap 1.

Four types of interrupt can occur inside the virtual machine. They are:-

They are:-

0	Hardware violation
1	Limit violation
2	Timer
3	Message

There is a procedure associated with each of these interrupts

to which a call is forced if the interrupt occurs. A trap vector is kept in the PRB containing a descriptor which points to the address of the procedure associated with each interrupt. At process creation time each entry of the vector is preset to point to the address of the terminate procedure. By using the above command a user may substitute one of his own procedures specified by P2 for trap entry P1. On entry to a trap procedure the stack will be set up as

- : SAOTIOJ

LINK	O	TRAP NO.	REASON
			A
			1
			ST

Hence, user trap procedures must have a spec of the form:--

proc kind SPEC name (I32,I32)

where the two I32 parameters correspond to trap number and reason.

Parameters:-

P1

This value determines which of the trap addresses is to be reset.

The currently allocated traps are:-

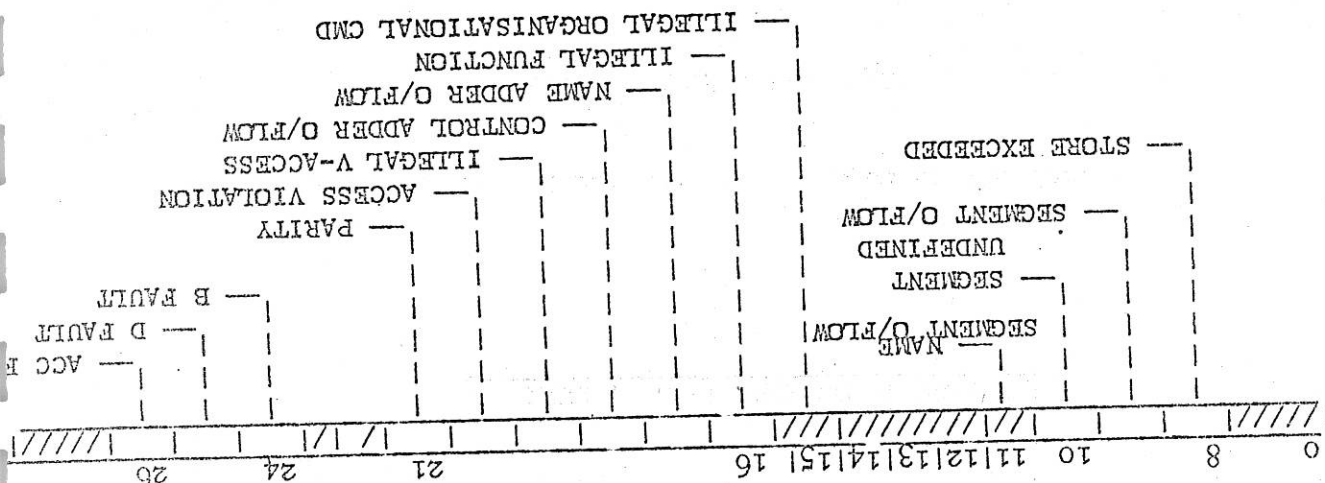
0	Program Fault, Illegal Order, Store Violation
1	Limit Violation
2	Local Timer Runout
3	Software Interrupt

This is a procedure type parameter specifying the appropriate trap procedure (for the MX/3 command this parameter is the address of the trap procedure).

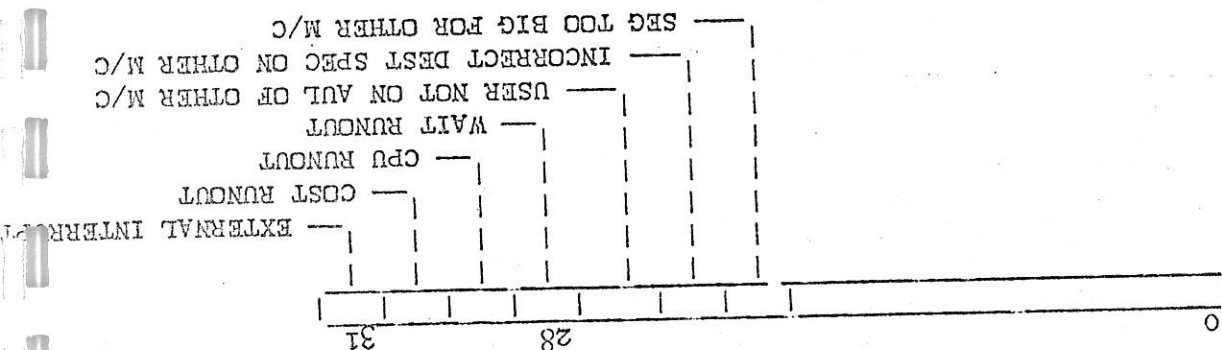
Further Notes

For all interrupts the message INTERRUPT.MASK is set to inhibit any further message interrupt before the trap procedure is called. The other interrupts may interrupt each other. The conditions on entry to each trap are now considered individually.

0) The reason word will be:-



1) The reason word will be:-



25.6.74/4

The first occurrences of CPU limit and cost limit exceeded cause additional allowances (equivalent to 5 cost units at priority 15) to be given before the trap procedure is entered. A further fault of the same type will cause the process to be terminated.

2) The timer will be set to the CPU balance before entering the trap procedure. The reason will be this CPU balance.

3) The reason will be a bit pattern in bits 24 -> 31 representing any unmasked channels containing a message.

11) PROC SPEC FORCE.INT (I32,I32,I32)

This procedure may be used by a supervisor or privileged process to force another process into its 'External Interrupt' trap procedure (trap 1).

Parameters:-

P1	The SPN of the process to be trapped
P2	The Process Identifier of the process
P3	This is the reason for the external interrupt, this parameter is only used as the reason if the process issuing the command is privileged.

12) PROC SPEC READ.INT.TRAP (132)

This procedure allows a process to read the address of one of its interrupt trap processing procedures.

Parameter:-

P1 The Trap Number

Output parameter:-

The trap address in Pw1.

21.3.72

13) PROC SPEC SET.TIMER (132)

This procedure allows a process to set its local timer.

Parameters:-

P1 The value to which the local timer is to be set
(units as for CPU limit in CREATE PROCESS).

If P1 > CPU remaining, it is set equal to CPU remaining.

This procedure enables a supervisor or privileged process to alter the priority of one of its processes.

N.B. Only privileged processes can specify priorities 14 and 15.

P1	is the SPN of the process
P2	is the PID of the process
P3	is the new priority

The virtual store organisational procedures perform tasks related to the virtual store structure. In particular, it can be seen by examining the list below that their actual functions are to create segments and alter their characteristic parameters.

When a segment is created, a size parameter defines how much store the segment requires and a further parameter defines the page size for the segment. The segment size will normally be rounded up to the nearest multiple of either 1K or the page size. Since the page size may vary within the system, segments may not always be created the same size. The maximum segment size is 64K. Store is not allocated unnecessarily as segment size merely dictates the limiting size. The actual allocation of real store only occurs as it is referenced in paged units.

The page size parameter gives a user the facility of specifying his preferred page size for the segment as a power of 2. However, the system will have the ability to override this estimate if it wishes. A negative page size would indicate that the user did not know the page size he required. In this case the system would specify the page size.

In a datum and limit system, it is unlikely that the whole of the specified segment will be allocated initially. To start with a smaller amount of real store will be allocated. This will then be extended in steps up to the specified limit as limit check fails interrupts occur.

There are five access control bits associated with a

segment, $\bar{S}$, $\bar{U}$, $\bar{R}$, $\bar{W}$, $\bar{O}$.

$\bar{S}$ = Software access bit. If set (= 1), the process may alter its access to this segment.

$\bar{U}$ = User/Executive bit. If set (= 1), the process may access the segment while in user mode.

$\bar{R}$ = Read Permission Bit. If set (= 1), the process is allowed to read the segment as data.

$\bar{W}$ = Write Permission Bit. If set (= 1), the process is allowed to write to the segment.

$\bar{O}$ = Obey Permission Bit. If set (= 1), the process is allowed to obey the segment as procedure.

When a segment is created, it is given the access $\%1E - 1.e.$,

full data access. The access permission can be changed using the CHANGE.

ACCESS command. Access combinations including both $\bar{W}$ rite and $\bar{O}$ bey permissions are illegal (i.e., a procedure segment cannot be written;

a data segment cannot be obeyed). Permissible access combinations for a user segment are:-

$\%09$	Execute-only procedure segment
$\%0D$	Execute-and-read procedure segment
$\%0C$	Read-only data segment
$\%0E$	Read-write data segment

List of Virtual Store Commands

1. CREATE SEGMENT (I32,I32,I32)
2. CHANGE.SIZE (I32,I32)
3. FREE.STORE (I32,I32)
4. INTERCHANGE (I32,I32)
6. CHANGE.ACCESS (I32,I32)
7. RELEASE.SEGMENT (I32)
8. READ.SEGMENT.STATUS (I32)
9. TRAN.SEGMENT (I32,u64,u64)
10. CREATE.C.SEG (I32, I32)

1) PROC SPEC CREATE, SEGMENT (I32, I32, I32)

This procedure creates a scratch segment or segments within

the current process.

The segment access is initially set to $\frac{2}{1}E$ (Read/Write segment with permission to change access).

Parameters:-

P1 This specifies the SEGMENT NUMBER by which the new

segment is to be known. If P3 is preset negative

the procedure will allocate a number and store it

in the name segment as an output parameter.

P2 This specifies the SIZE of the segments in words and

acts as described previously.

P3 This specifies the preferred PAGE SIZE as a power of

2 and acts as described previously.

Output parameter:-

Segment number in PW1.

Actual size created in PW2.

2) PROC SPEC CHANGE SIZE (I32,I32)

5.25

This procedure changes the size limit of a specific segment.

Parameters:-

P1

This specifies the SEGMENT NUMBER.

P2

This specifies the new SIZE limit. The usual rounding

up is carried out if necessary. The maximum size allowed

is 64K, if the size is 0 the segment will be released.

The page size is unaltered.

3) PROC SPEC FREE.STORE (I32,I32)

This procedure releases the real store allocated to a segment above a specified size. The size limit of the segment is unaltered. This procedure has no effect on paged machines, that is KVS and the 1905E.

Parameters:-

P1 This specifies the SEGMENT NUMBER.
 P2 This specifies the SIZE above which all current real store reference is to be released.

4) PROC SPEC INTERCHANGE (I32,I32)

This procedure interchanges the specifications of two segments. If one of the segments does not exist, the segment is in effect renamed.

Parameters:-

P1 The SEGMENT NUMBER.
P2 The SEGMENT NUMBER.

6) PROC SPEC CHANGE.ACCESS (I32,I32)

This causes the access associated with a specific segment

to be altered.

Parameters:-

P1

The SEGMENT NUMBER.

P2

The new ACCESS specification is situated in the

bottom 5 bits of this parameter. The bits are

in the order described previously. If the current

access to this segment does not permit alteration

a fault occurs.

This causes a specific segment to be removed from the virtual store of the calling process. The count of the total number of processes which have this segment allocated is decremented. If this count becomes zero, the segment and its contents are destroyed even if the segment was open as a file.

Parameter:- P1 The SEGMENT NUMBER.

7) PROC SPEC RELEASE.SEGMENT (132)

8) PROC SPEC READ.SEGMENT.STATUS (I32)

This command permits a process to request information about
of one its segments.

Parameter:-

P1

The SEGMENT.NUMBER of the segment about which

information is required.

Output parameters:-

In PW1, the SIZE of the segment.

In PW2, the present ACCESS of the process to the segment

(usual 5 bits at least significant end).

In PW3, the PAGE.SIZE of the segment as a power of 2.

In PW4, the SHARED.SEGMENT.COUNT for the segment.

9) PROC SPEC TRAN.SEGMENT (I32,U64,U64)

This command may be used to transfer the ownership of a segment to a specified user.

Parameters:-

P1 Segment number of the segment to be transferred.
P2 The USER NAME of the user who is to be charged
for the segment.
P3 The PASSWORD of the user.

A fault is indicated if the recipient user's store balance is not large enough.

10) PROC SPEC CREATE.CSEG (1,1)

This procedure creates a scratch common segment (P2), which is a shared segment with the private segment (P1). The segment is created with a size of 16K. Only privileged users may use this command.

Parameters:-

P1

Common segment number.

P2

This specifies the segment number by which the

new segment is known to this process. If P2 is

negative, a segment number will be allocated.

Output parameter:-

Segment number in PW1.

File Commands

The file store is a collection of segments which may be linked into the virtual stores of processes by means of the file organisational commands. Each authorised user has a set of files which he can address through his File Directory. This directory is itself a file and is addressed from the users entry in the Authorised User List. A mechanism is provided by means of which a user may provide other users with access to his files.

Each file in the system is allocated a unique System File Number (SFN), and all the information which the executive requires to know about a file is held in the System File Table in the position corresponding to the SFN. Users address files by name and their directories contain entries giving users file name, access permission and SFN.

The file commands are as follows:-

1. FILE (U64, I32, I32)
2. OBTAIN.FILE (U64, I32, I32)
3. OBTAIN.USER.FILE (U64, U64, I32, I32)
4. OPEN.FILE (U64, I32, I32)
5. OPEN.USER.FILE (U64, U64, I32, I32)
6. DELETE.FILE (U64)
7. SECURE.FILES ()
8. DUMP.FILE (U64)
9. CATALOGUE (I32)
10. RENAME.FILE (U64, U64)
11. GIVE.FILE (U64, U64, U64, I32)
12. TRAN.FILE (U64, U64, U64)

This procedure defines a specified segment to be a file in the directory of the user who owns the process. If the file already existed then the old copy is updated with the segment.

Parameters:-

P1 The FILENAME by which the file is to be known

in this directory.

P2 The Segment Number of the segment to be defined as

a file.

P3 The Access which will be permitted when opening the

file. This parameter consists of the usual S bits (S, U, R, W, O - see 5.26), but the S and U bits are ignored. Write permission is interpreted as

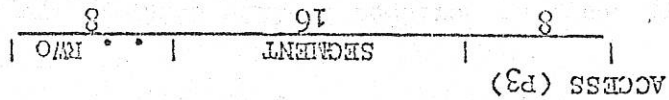
permission to update the file. The second and

third bytes of the access specify the segment

into which the file should be opened (zero implies

any segment).

For example:-



2) PROC SPEC OBTAIN.FILE (064,132,132)

This procedure obtains a file for the current process. The OPEN.FILE and REPLACE.FILE commands might halt the calling process under certain circumstances (see the relevant descriptions). If a process does not wish to be halted, it should first reserve the file using this command. When the file is available for immediate use by the process, a message with the file name and preferred segment in the header and the file as the segment will be sent to the process. It may then use the file with no danger of being halted.

The format of the short message is:-

6 BYTES	1 BYTE	8 BYTES	2 BYTES
CONTROL	PWA	FILE NAME	SEG

Parameters:-

P1 The FILENAME of the file to be obtained, which must be in the directory of the user who owns the calling process.
P2 The access which will be required when the file is opened (REPLACE implies Write access). This is specified as for CREATE.FILE.
P3 The channel on which the file is to be sent.

3) PROC SPEC OBTAIN.USER.FILE (U64,U64,U64,I32,I32)

As for OBTAIN.FILE (2, above) except that the file is

in another users directory.

Parameters:-

P1 The USERNAME of the user in whose directory the
file exists.

P2 The PASSWORD of this user.

P3,P4,P5 as P1,P2,P3 in OBTAIN.FILE.

This procedure creates a segment as the virtual address location of a specified file from the current users directory.

Parameters:-

- P1 The FILENAME by which the file is known in this directory.
- P2 The segment number of the segment into which the file is to be opened. If the parameter is negative, the system will choose a segment.
- P3 The access required. This consists of the usual 5 bits, (S, U, R, W, G - see 5.50), and an extra bit (digit 26)X. If this bit is set (= 1), the process requires exclusive access to the file.

In this case, if it is already open, the process will be halted. If after the file has been opened any other process tries to open it, then that process will be halted. Exclusive access can only be requested if the directory entry indicates that write permission is allowed. The S and U bits are ignored by the file system (and are always assumed to be 0 and 1 respectively).

The process may also be halted if the file has been offloaded to offline storage.

Output parameter:-

In Pw1, the segment number actually used.

5) PROC SPEC OPEN.USER.FILE (u64,u64,u64,i32,i32)

As for OPEN.FILE (4, above) except that the file is in another users directory.

Parameters:-

P1 The USER NAME of the user who owns the file.
P2 The PASSWORD for this user.
P3,P4,P5 as P1,P2,P3 in OPEN.FILE.

6) PROC SPEC DELETE.FILE (U64)

This deletes a file from the directory of the user owning the current process. If any other process has the file open or has requested the file, then the command is faulted.

Parameter:-

P1 The FILENAME by which the file is known in the current users directory.

26.6.72/2

26.6.72/1

No parameters.

[Not initially implemented].

current user in their present status.

This procedure secures the files and directory of the

7) PROC SPEC SECURE FILES ()

8) PROC SPEC DUMP.FILE (UG4)

This dumps a file onto secure backing store.

Parameter:-

P1

The FILENAME of the file in the current users

directory.

This causes a segment containing the current users directory to be placed in the process' virtual store. The directory segment contains an area of fixed information at the beginning of the segment followed by entries of 16 words for each file.

Parameter:-
P1

The segment number of the segment in which the directory is to be placed.

DIRECTORY

UID	FIRST ENTRY DISP
LAST ENTRY DISP	
'OTHER DIRECTORY ATTEMPT'	
FILE COUNT	

VALID STATUS	
FILE	NAME
SPN	
	EMC
PREFERRED SEGMENT	
ACCESS	
OWNERS UID	
SIZE	

0
1
2
3
4
5
6
7
8
9
10
11
15

10) PROC SPEC RENAME.FILE (u64,u64)

This alters the name of a file in the current users directory.

Parameters:-

P1 OLD FILENAME by which the file is currently known.
P2 NEW FILENAME by which the file will be known.

11) PROC SPEC GIVE.FILE (u64,u64,u64,132)

This procedure allows the current user to give another user shared access to one of his files.

Parameters:-

- P1 The USERNAME of the user to be given shared access.
- P2 The PASSWORD of this user.
- P3 The NAME of the file in the current users directory.
- This is also the name by which the new user will know the file.
- P4 The access which is to be permitted to the new user.

12) PROC SPEC TRAN.FILE (u64,u64,u64)

This procedure transfers the ownership (and therefore the charges of a file from the current user to another user (who may already have a copy of the file). The new owner will be given the access permission that the original owner had.

Parameters:-

P1 The USER NAME of the new owner.

P2 The PASSWORD of the new owner.

P3 The FILENAME by which the file is known in the

present owners directory. [The name may be different in the new owners directory].

1. SET.CH.STATUS (I32,I32,I32,I32)
2. SEND.MESSAGE (S,S,I32,I32,I32)
3. READ.MESSAGE (S,S,I32,I32)
4. LOOK.UP.PROCESS (U64)
5. LOOK.UP.PROCESS.NAME (I32,I32)
6. READ.CH.STATUS (I32)
7. SET.INT.MASK (I32)
8. READ.INPUT.STATUS ()
9. SEND.EXEC.MESSAGE (S,S,I32,I32,I32)

1) PROC SPEC SET.CH.STATUS (I32,I32,I32,I32)

A process may change the status of any of its channels by

this procedure.

Parameters:-

P1 Channel number (0 - 7)
P2 Channel type

The significance of these digits is as follows:-

Digit	when = 1
26	Close channel after first message (see P3)
27	Open to ATL
28	Open to EXEC
31	Open to SPN

If digits 27 - 31 are zero, the channel is closed.

P3 Until MUSS/MX3 are abandoned, P3 is used to set digit 26 of P2 as follows, if P3 = 1, then digit 26 of P2 is set. SPN (to become PID when MUSS/MX3 abandoned) of the process to which P1 is dedicated when digit 31 is set = 1. In the case of 'not dedicated to SPN' P4 is irrelevant.

2) PROC SPEC SEND.MESSAGE (S,S,I32,I32,I32)

This command sends a message to an input channel of a process. It also forms some additional information which appears with the message. Part of this allows the receiver to establish the origination of the message. The rest is a conversion of some of the input parameters to this command. These specify if a reply is expected or if a segment is being passed over with the message.

Parameters:-

P1 This is a string message descriptor specifying the

address of the message in the senders virtual store.

P2 This is a vector descriptor (of 32-bit elements)

specifying the 'address' of the message destination.

The vector must be laid out as follows:-

Word

0

SPN of receiver.

1

Process Identifier of receiver.

2

The INPUT CHANNEL of the receiving process

to which the message is to be sent. This

channel must be open to the sending process.

If the channel is counting then the count

will be decremented. If the count becomes

zero, the channel is closed.

(The receiving process will receive with the message a

similar 'vector' giving a 'return address').

This specifies a channel in the sending process which

may be used as a 'reply channel' by the receiving process.

This channel number will be placed in the 'return address'

vector given to the receiving process when it reads the

message. If no reply is expected, the parameter should be

set negative. If this parameter is set to 16, the process

will be suspended after sending the message, awaiting a

'FREE.PROCESS' command from its supervisor or a privileged

process.

This specifies a SEGMENT NUMBER in the senders virtual store that is to be passed over in a long message.
 A short message has P4 set negative.
 This specifies the ACCESS permission on the segment P4.
 In a short message P5 is irrelevant.

P4

P5

3) PROC SPEC READ.MESSAGE (S,S,I32,I32)

This procedure reads the first message on an input channel of the calling process to an address in its virtual store. All reference to this message on the input channel is lost and the next message in the queue becomes the leading entry on that channel.

Parameters:-

P1

This is a string descriptor pointing to the address in the virtual store of the calling process at which the message string is to be stored.

P2

This is a vector descriptor pointing to the address in the virtual store of the issuing process at which some information about the sender of the message is to be stored. This vector is similar in layout to the 'address' vector used by SEND.MESSAGE to specify the destination of a message.

Word

0

1

2

SPN of sender.

Process Identifier of sender.

Senders REPLY CHANNEL. This will

be the parameter specified by the

sender as a channel on which he

is prepared to receive a reply.

It will be negative if he does

not expect a reply. A reply

channel of 16 indicates that the

process requires no reply, but has

been suspended and requires awakening

by a FREE.PROCESS command from its

supervisor or a privileged process.

This vector may thus be used as the 'address' of a reply

message.

This specifies the INPUT CHANNEL (0 - 7) from which the

message is to be read.

P3

This specifies the SEGMENT NUMBER by which the

P4

calling process will address a segment passed over in a long message. If a segment with this number already exists in the virtual store of the issuing process, then the command is faulted. If this parameter is preset negative, a segment number will be allocated by executive and stored in word 1 of the name segment. This parameter is irrelevant when reading a short message.

Output parameters:-

In PW1, the segment number used to hold the long message, (this will be a copy of P4 unless that parameter is negative). PW1 will be negative if no segment is sent.
In PW2, the access granted to the segment, (this is irrelevant if no segment is sent).
In PW3, the user identifier of the sender.

A process may find the SPN and Process Identifier of a

'named' process by this procedure.

Parameter:-

P1

The NAME of the process required. If the next to last character is '/', then the 'name' is interpreted as 'process name'/'machine identifier', and the process is looked up in the appropriate machine. MUG is identified by M and the 1900 by N. Since the complete 'name' cannot be bigger than 6 characters in the 1900 and 6 characters in MUG, all 1900 processes are accessible from MUG, but only MUG processes with names less than 5 characters are accessible from the 1900.

Output parameter:-

SPN and Process Identifier for the named process in Pw1

and Pw2 respectively. The SPN for processes in other machines will have the machine number (2 for MUG, 1 for 1900) in the top 2 bits of the 12 bit SPN.

5) PROC SPEC LOOK.UP.PROCESS.NAME (I32,I32)

A process may find the NAME of another process from its SPN and process identifier using this procedure. If P1 is preset negative, the calling process wishes to find its own name.

Parameters:-

P1 The SPN of the process.
P2 The process identifier of the process.

Output parameter:-

The NAME of the process in PWW1.

6) PROC SPEC READ.CH.STATUS (132)

A process may find the status (type, count, SPN) of any

of its channels by this procedure.

Parameter:-

P1

The number of the channel whose status is required.

Output parameters:-

Type in PW1, SPN (relevant only if dedicated), in PW2.

7) PROC SPEC SWT.INT.MASK

A process may set this mask to inhibit messages from causing interrupts on specific input channels.

Parameter:-
P1

The interrupt mask. The l.s. 8 bits are used such that if bit '1' is set (= 1) then the interrupt state of channel '1' is inhibited.

3) PROC SPEC READ.INPUT.STATUS

This allows a process to examine which of its channels

have outstanding messages.

Output parameters:-

In PW1, the interrupt mask.

In PW2, the interrupt status of the form, if bit 1
is set then channel 1 has an outstanding interrupt
(bit 0 1.s.).

9) PROC SPEC SEND.EXEC.MESSAGE (s,s,132,132,132)

This command allows a privileged process to send a message to a channel of a process which is open to EXEC. This command has the same specification as SEND.MESSAGE.

Parameters:-

p1 These parameters are identical to
 p2 those for SEND.MESSAGE
 p3
 p4
 p5

Procedures for User Management

A number of procedures exist to enable a user to create new users, and to allocate resources to them. In the following descriptions, these definitions will apply:-

1. The current user. The user on whose behalf the calling program is running.

2. The manager of a user. The user who created this user.
3. A sub-user of a user. A user created by (managed by) this user, or by one of his sub-users.

1) PROC SPEC NEW.U (U64, U64, U32, U32, U32, U32)

This procedure creates a sub-user of the current user.

The new user will have resources, and name and password as

defined by the parameters.

- P1 - USER NAME to be given to the new user.

- P2 - PASSWORD to be given to the new user.

- P3 - The 1900 CPU income for the user.

- P4 - The 1900 CPU income for the user.

- P5 - The Offline O/P income.

- P6 - The Online I/O income.

- P7 - The 1900 store limit.

- P8 - The 1905 store limit.

- P9 - The file store limit.

2) PROC SPEC DEL.U (U64)

This procedure will delete a sub-user of the current

user, delete his files and return his resources to the current user.

- P1 - USER NAME of the user to be deleted.

3) PROC SPEC NEW.P (U64)

This procedure will change the password of the current user.

- P1 - PASSWORD to be given to the current user.

4) PROC SPEC SET.U (U64, 132, 132)

This procedure sets the value of one of the allowances

for a sub-user of the current user.

P1 - USERNAME of the sub-user.

P2 - The allowance to be changed:-

0 => 1900 CPU income

1 => MUS CPU income

2 => Offline I/O income

3 => Online I/O income

4 => 1900 store limit (in K)

5 => MUS store limit

6 => File store limit

P3 - The new value of the allowance.

5) PROC SPEC READ.U ()

This procedure reads the allowances and current

balances of the current user as follows:-

PW1 - 1900 CPU income PW2 - 1900 CPU balance

PW3 - MUS CPU income PW4 - MUS CPU balance

PW5 - I/O Materials income PW6 - I/O Materials balance

PW7 - I/O time income PW8 - I/O time balance

PW9 - 1900 store limit PW10 - 1900 store balance

PW11 - MUS store limit PW12 - MUS store balance

PW13 - File store limit PW14 - File store balance

Miscellaneous Commands

1) PROC SPEC CREDIT.USER (U64,U64,I32)

This procedure will credit a user's account with a specified amount, deducting the equivalent amount from the issuing process's current account.

Parameters:--

P1	The USERID of the user to be credited.
P2	The user's PASSWORD.
P3	The AMOUNT with which the user is to be credited.

2) PROC SPEC DEBIT.USER (U64,U64,I32)

This procedure debits a user's account by a specified amount, transferring the equivalent amount into the issuing process's current account.

Parameters:-

- P1 The USERID for the user.
- P2 The user's PASSWORD.
- P3 The AMOUNT by which the user's account is to be debited.

3) PROC SPEC TIME.AND.DATE ()

This procedure returns the current time and date, in packed decimal form, to the process.

Parameters:-

None.

Output parameters:-

The Time and Date, in PW1 and PW2 respectively, in packed decimal form, e.g.,

13.45 and 35 seconds, on 12th March 1971

is represented *00134535, *00120371.

4) PROC SPEC NAMES ()

This procedure returns user name, password and process name in PW1, PW2 . . . PW6.

5) PROC SPEC FIND.U(U64,U64)

This procedure validates the user name and password in P1, P2 and returns the internal user ident.

Parameters:-

P1 The User name

P2 The User's password

Output parameters:-

The internal user ident in PVI.

(6) PROC SPEC U.NAME (1)

This procedure returns the user name in PW1 given the user identifier.

Parameters:-

P1

The user identifier

Output parameters:-

The user name in PW1.

Disc Commands

1. UPDATE.C.S (I32, I32, I32)
2. READ.DISC (I32, I32)
3. WRITE.DISC (I32, I32)

5.65

5.8.74/1

1. PROC UPDATE.C.S (I32, I32, I32)

This command allows a common segment to be updated on a non current version of the SS. For each common segment there is one authorised user who is allowed to update that segment. The size of the new segment must not be greater than the size of the segment it replaces.

P1	MVS Disc Switch number (0 -> 31). The SS defined by the initial block on the first block under the switch is to be altered.
P2	The common segment that is to be updated.
P3	The segment that is to update the common segment..

2. PROC READ.DISC (I32, I32)

This command reads a 256 32 bit block from the MUS disc.

P1 Address (in 32 bit words) in a users segment where the
 block will be read to. P1 assumed to start on a 256
 word boundary.

P2 MUS disc address.

3. PROC WRITE.DISC (I32, I32)

This command writes a 256 32 bit block to the MVS disc. The disc switches for the block must be set manually to not isolated before the command is issued.

P1	As in READ.DISC.
P2	As in Read Disc.

- 0 - not allowed
- 1 - word
- 2 - double word
- 3 - descriptor

Three parameter types are permitted:--
Thus there is no provision for routines with more than 11 parameters.
parameter but the first two bits specify the mode of the result.
In the parameter specification two bits are used for each

Also bits 22 and 23 of the count will be ignored.
The count should be in byte units and the pointer in word units.
justified within an integral number of words and zero filled.
name should be packed three symbols to a word and must be left
of the name and a pointer to the first symbol of the name. The
requires a 48 bit parameter containing a count giving the size
is the routine all compilers should use to look up names. It
Entry 1 in the directory is for the routine FIND.N which

- Word 0 - the first 3 symbols of the routine name
- Word 1 - a pointer to the rest of the name
- Word 2 - a parameter specification
- Word 3 - either the serial number of the routine or
a jump to its 1st instruction

four words are:--
even though the routine may be recompiled at a new address. The
Once a routine has been allocated a position it will not be changed
and consists of a list of 4-word items, being one item for each routine.
The main part of the directory is at the beginning of segment 55
user access to every library routine.

hence in so far as language conventions allow, all compilers can allow
specification and a jump to the start of each routine in the library,
commands. There is also a directory which gives the name, parameter
it does contain a compatible set of I/O routines and organisational
organisation. It is therefore less general than the MVS system but
routines with fixed parameter areas and not on an Algol like procedural
The library system provided under the 1900 system is based on

General Organisation

Type 1 is a 24-bit parameter and types 2 and 3 occupy 48 bits. In type 3 the convention described for the parameter of FIND.N apply except if the descriptor refers to a word vector the count should be in word units. A fixed area of segment 0 (104 - 127) has been allocated for parameter passing. The calling sequence should place the actual parameters in this area in the correct order packed so as to leave no empty words.

The output parameter mechanism is less formal. Some routines may yield an integer result in register 0 or a double word result in ACC. There is also a 10 word output parameter area at address 80 which compilers will provide access to through predefined names. Word 0 of this area is the command status returned by organisation commands.

Word 3 in a directory entry will contain a jump to the starting address in the case of ordinary routines and a serial number in the case of organisational commands. It is expected that the following call sequences will be compiled:-

ORGANISATIONAL COMMAND

SET PARAMETER 1
SET PARAMETER 2
SET PARAMETER M

176 0 N

ORDINARY ROUTINE

SET PARAMETER 1
SET PARAMETER 2
SET PARAMETER M

071 7 X

where X is a lower data word allocated by the compiler to hold the address of word 3 of the directory entry for the routine. The result returned to compilers by the FIND.N routine is the address of word 2 in the directory or 0 if the name is not present.

It is expected that the treatment of actual parameters will be language dependent. Algol, for example, might allow the following:-

- type 1 - an integer expression yielding its value
 type 2 - either a real expression yielding its value, or a string of not more than 6 symbols which will be packed right justified and zero filled.
 type 3 - a symbol string which will be packed appropriately yielding a pointer and count, or a vector name yielding a pointer and count, or a variable yielding its address.

As in MUG any library routine can be called from the program since the syntax of the actual parameters indicate their type. This syntax is as for the MUG program control sequence described earlier. In SPG the parameter spec is again ignored. Any actual parameter which is an SPG expression will be treated as type 0. Types 1 and 2 may be encoded as in the program control sequence. Also a type 2 parameter can be encoded as two type 1 parameters.

The Library Content

The list below gives the routines in the basic MX/3 library and their parameter spec. Organisational commands are numbered. The parameter types are denoted thus:-

- type 1 - I
 type 2 - II
 type 3 - S

I results are returned in BO, II results in ACC. It should be noted that PROC type parameters do not exist in the 1900 system hence there is some loss of compatibility. Those MUG procedures which have PROC parameters such as SET TRAP are given I type parameters on the 1900. It is expected that a label will be substituted as the actual parameter.

Most routines are sufficiently compatible with the corresponding MVS procedures and no further description is given except for those concerned with traps. Only 8 traps are provided by the 1900 system. They will be used for the same reasons as the first 8 traps on MVS. The system reserves in lower data three vectors of 8 words each. Word 1 of each is used as follows:-

- vector 1 - address to which control is passed if trap 1 occurs (set from parameter of SET.TRAP).
- vector 2 - reason for entering trap 1.
- vector 3 - address from which entry to trap 1 occurred.

```

FIND.N (S) I
INIT.I.O (I)
IN.SPEC (I, I, I)
SET.INPUT (I, I)
SELECT.INPUT (I)
CURRENT.INPUT (I)
I.POS (I)
SET.I.POS (I)
SELECT.TEXT (I)
SELECT.HEADER (I)
IN.SYM (I)
DELETE.INPUT (I)
END.INPUT (I)
CREATE.INPUT (I, S)
IN.FILE.SPEC (I, I, I, I, I)
SET.FILE.INPUT (I, I)
IN.STATUS (I) I
IN.SOURCE (I, S)
IN.REPLY (I, I, I, I)
TRAN.IN (I, I, I, I)
NEXT.SYM (I)
SET.TRAP (I, I)
ENTER.TRAP (I, I)
STOP (I)
STAMP (I)
PPC.SEO (I)
DUMP.REGISTERS (S)
LOAD.REGISTERS (S)
OUT.SPEC (I, I, I, I, S, I, I, I, I, I)
SET.OUTPUT (I, I, I)
OUT.FILE.SPEC (I, I, I, I, I)
SET.FILE.OUTPUT (I, I)
SELECT.OUTPUT (I)
BREAK.OUTPUT (I)
CURRENT.OUTPUT (I)
O.POS (I)
SET.O.POS (I)

```


Line	Statement
1	CREATE SEGMENT (I,I,I)
2	CHANGE SIZE (I,I)
3	FREE STORE (I,I)
4	INTERCHANGE (I,I)
5	
6	CHANGE ACCESS (I,I)
7	RELEASE SEGMENT (I)
8	TRAN SEGMENT (I,II,II)
9	READ SEGMENT STATUS (I)
10	CREATE PROCESS (II,II,II,II,II,II,II,II,II,II,II,II)
11	
12	
13	SUSPEND PROCESS (I,I)
14	FREE PROCESS (I,I)
15	WAIT (I,I)
16	TERMINATE PROCESS (I,I,I)
17	INITIALISE CHANNEL (I,I,I,I,I,I,I,I)
18	
19	READ STATUS (I,I)
20	READ PARAMETER (I,I,I)
21	SET INT TRAP (I,I)
22	FORCE INT (I,I,I)
23	READ INT TRAP (I)
24	SET TIMER (I)
25	
26	
27	SWOP SUPERVISOR (I,I)
28	RESCHEDULE (I,I,I)
29	TIME AND DATE (I)
30	SEND MESSAGE (S,S,I,I,I,I)
31	
32	READ MESSAGE (S,S,I,I)
33	SET CH STATUS (I,I,I,I,I,I)
34	READ CH STATUS (I)
35	SET INT MASK (I)
36	READ INPUT STATUS (I)
37	LOOK UP PROCESS (II)
38	LOOK UP PROCESS NAME (I,I)
39	
40	FILE (II,I,I,I)

5.8.74/5

41
42 OPEN.FILE (II,I,I)
43
44 FIND.U (II,II)
45 U.NAME (I)
46 DELETE.FILE (II)
47
48
49 CATALOGUE (I)
50 RENAME.FILE (II,II)
51
52
53 NAMES ()
54 NEW.U (II,II,I,I,I,I,I,I,I,I)
55 DEL.U (II)
56 NEW.P (II)
57 SET.U (II,I,I)
58 READ.U ()
59 OPEN.DIR
60 RELEASE.DIR
61 READ (I,I)
62 WRITE (I,I)

The Organisation of I/O

The I/O system has a minimum requirement of 6 words in lower data and an I/O control segment. In addition there will be one further segment for each buffered stream. In lower data the words are used as follows:-

- 1) a pointer to the current word in the input buffer (INPUT)
- 2) a pointer to the last word output to the buffer (OUTPUT)
- 3) a pointer to the control area of the currently selected input (IN.CONTROL)
- 4) a pointer to the control area of the currently selected output (OUT.CONTROL)
- 5) the current word from the input buffer (I.BUFF), its least significant byte is the last character read, and the original first character will have 128 added
- 6) the current word from the output buffer (O.BUFF), the next character output will be placed in its least significant byte and 128 is added onto the first character output to the word.

In the I/O control segment a 32 word area is allocated for each of the possible 8 input streams and 8 output streams. These control areas are laid out as follows.

Input Control Area

IN.PTR	I.BUFF	PID	CH/SPN	D	S	M	LINE	NO	MESSAGE	*4	S	M
24	24	24	24	8	8	8	24	75	x	8	8	8

Output Control Area

OUT.PTR	O.BUFF	PID	CH/SPN	S	D	M	COST	MESSAGE	*4	S	M
24	24	24	24	8	8	8	24	75	x	8	8

- D - is destination device number
- S - is source device number
- M - mode of output

Input Buffer Segments

The input system assumes the following layout for the segment:-

BYTE	BYTE	DEVICE	1st
DISPLACEMENT	COUNT	TYPE	No.
32	32	32	32
----	----	----	----
1st			

Output Buffer Segments

Before an output segment is sent the first 16 bits are set up

as above. Whilst the output is being accumulated it is used as follows:-

BREAK	BREAK	COST	DEVICE	1st
COUNT	LIMIT	LIMIT	No	BYTE
24	24	24	8	16
----	----	----	----	----
1st				

INPUT 0

OUTPUT 0

179

512	OFF-LINE I/O WORKSPACE	543
544	PARAM DUMP FOR PPC.SEO	551 575
576	STRING STACK	588
590		

576	PARAM DUMP FOR PPC.SEO	STRING STACK	→
			\$80

576

544

512 OFF-LINE I/O WORKSPACE 543 [R.CNT]

Small Business Administration and Economic Development Department, U.S. Department of Commerce, Washington, D.C. 20540-0001

179

0 10000

INPUT 0

0	7	8	18	22	29	31
UIDS DEFAULT HEADER // USED BY FILE COMMANDS						
USED BY BREAK						

Overall layout of I/O Control Segment

Al. 12

1024	FILED PROCEDURES WORK SPACE	2047
	AND COMMUNICATION AREA	

AND COMMUNICATION AREA

1024

590

576

544

512 OFF-LINE I/O WORKSPACE 543 [R.CNT]

Small Business Administration and Economic Development Department, U.S. Department of Commerce, Washington, D.C. 20540-0001

179

0 10000

INPUT 0

0	7	8	18	22	29	31
UIDS DEFAULT HEADER // USED BY FILE COMMANDS						
USED BY BREAK						

Overall layout of I/O Control Segment

Al. 12

16.5.74/3