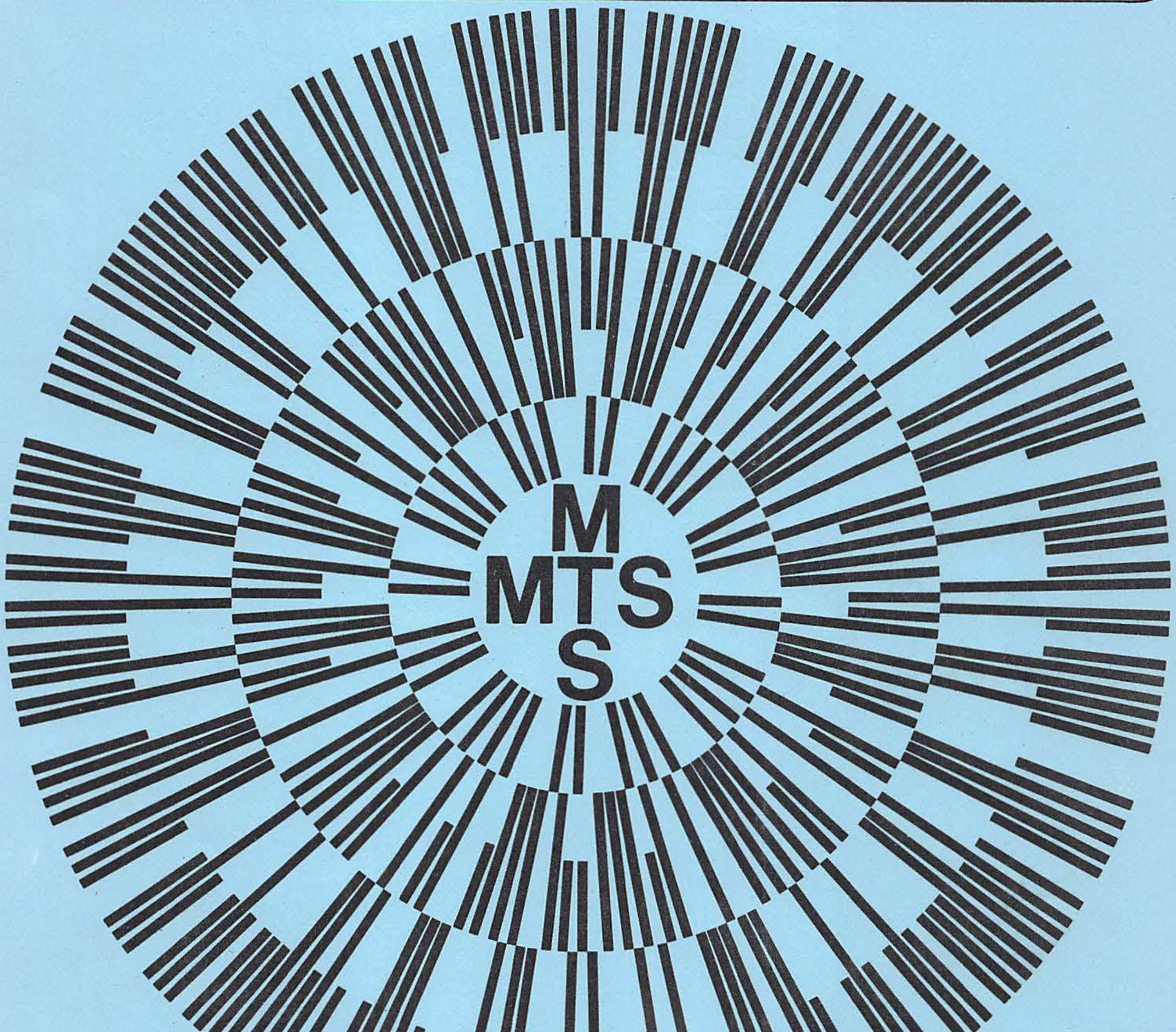
DAVIS

THE UNIVERSITY OF ALBERTA
COMPUTING CENTER
PUBLICATION



COMMANDS

ACKNOWLEDGEMENTS

This manual was largely compiled from material prepared by the staff of the University of Michigan Computing Center. Their documentation was invaluable and we are indebted to them for allowing us to use it. In particular, the following were most useful:

MTS Users' Manual, Second Edition, Volumes I and II

MTS Users' Manual, Third Edition, Volume 2

Introduction to MTS and the Computing Center (Flanigan)

Computing Center News Items

Computing Center Memos

The Computing Center wishes to personally acknowledge the assistance of Mike Alexander and Don Boettner who helped us to establish MTS at the University of Alberta.

Acknowledgement should also be made to the Computing Centre, University of British Columbia, for information obtained from some of their documentation and to I.B.M., whose manuals provided certain sections for our manuals.

DISCLAIMER

This MTS manual is a combination of earlier manuals, update notices, memos and limited experience with the system itself. Because of this, certain discrepancies are bound to occur and the Computing Center would appreciate being notified of all differences between what this manual says and what the system actually does.

This publication is intended to represent the current state-of-the-system. However, it should not be construed as an obligation to maintain the system as so stated. The MTS system, like most good systems, is continually being improved. As a result, additions, extentions, changes and deletions will occur. Notice of such changes will be made and provision for a manual updating service has been planned.

Errors, comments and suggestions should be sent to:

Information Coordinator
Computing Center
University of Alberta

TABLE OF CONTENTS

1.0 NOTATION

The description of the commands give the name, a prototype, explicitly giving the syntax, descriptions of the purpose, usage, and effect, and examples.

For those commands which may be abbreviated, the necessary part is identified by underlining in the prototype command.

Notation conventions used in the prototype are:

lower case	-	represents a generic type which is to be replaced by an item (such as a file name) supplied by the user.
upper case	-	indicates material to be repeated verbatim in the command (although this can be entered as uppercase, lower case or mixed)
brackets []	-	indicates that the material within the brackets is optional
braces { }	-	indicates that the material within the braces represents choices, from which exactly one must be selected.
dots ...	-	indicates that the preceeding syntactic unit may be repeated indefinitely
underlining	-	(1) indicates the default value where several choices exist. (2) indicates the abbreviated form

2.0 DEFINITIONS

Common Generic Types

hhhh -	a hexadecimal constant consisting of 1 to 8 hexadecimal digits
Fname -	a file name
FDname -	a file <u>or</u> device name
GRx -	the general register x, where x is a decimal integer from 0 to 15 or a hexadecimal integer from 0 to 9 or A to F.
FRy -	the floating point register y, where y is one of the integers 0,2,4, or 6.
RF=$\left\{\begin{matrix}\text{hhhh}\\ \text{GRx}\end{matrix}\right\}$-	a core storage relocation factor, where hhhh is the hexadecimal value of the relocation factor or GRx indicates the general register whose contents are to be used as the relocation factor.
limitspecs-	keyword parameters which specify limits for execution time, pages printed or cards punched (see complete description below).

iospecs - keyword parameters of the form

logical I/O unit=FDname

where logical I/O unit is one of the following:

SCARDS
SPRINT
SERCOM
SPUNCH
GUSER
numbers 0 to 9

locn - an address of a core storage location given by an optional relocation factor and a displacement, i.e.,

[RF= $\left\{\begin{matrix}\text{hhhh}\\ \text{GRx}\end{matrix}\right\}$] hhhh

(see discussion at Relocation Factors and Core Storage Addresses below).

Relocation Factors

A global relocation factor is maintained for referencing locations in core storage. Initially it's value is <u>zero</u>. The global relocation factor can be changed by the $SET $\overline{\text{RF=}}$ command.

A local relocation factor, which overrides the global relocation factor, can be given in certain of the commands, and remains in effect for the duration of the command, unless changed by a subsequent local relocation factor.

Core Storage Addresses

A displacement is added to the current value of the relocation factor to provide an absolute core storage address.

Limit Parameters

In order to prevent run-away jobs in batch runs, it is necessary to provide some means for the user to limit the time a job may use and the amount of output (paper and cards) it produces. It is also often desireable for users at a terminal to be able to limit a RUN of a program. To provide this facility, limit keyword-parameters are provided. They are of two types, global and local.

Global limits are limits for the entire job, from SIGNON to SIGNOFF. They are to be placed on the $SIGNON card for batch jobs. Default values are assumed for any omitted parameters. The current default values are given in the table below.

Within the global limits imposed at SIGNON, the user may specify separate local limits on any RUN, LOAD, RESTART, OR START command. If a RESTART or START command specifies no limits, then what is left of the limits specified by the original RUN or LOAD command is used.

The first limit to be exceeded, whether global or local, causes a comment to be printed and the job to be terminated (in batch operation), or a return to command mode (in terminal operation).

Quantity	Prototype	Units	Global Default
CPU time	$\underline{T}$IME=n[$\frac{S}{M}$]	seconds minutes	30 seconds
Pages of output	$\underline{P}$AGES=i	pages	50 pages
Punched cards	$\underline{C}$ARDS=i	cards	1 card

where $\underline{i}$ is an integer of up to 5 digits.
$\underline{n}$ is a number similar in form to a line number: up to 5 digits in front of the decimal point; up to 3 digits after the decimal point.

The necessary part of the keyword is underlined.

Examples:
```
T=6
Time=0.1m
P=62
CARDS=400
```

3.0 COMMANDS

In the following command descriptions, the dollar sign ($) preceeding the command name, when occurring as the first character in an input line from *SOURCE*, indicates that the line contains a command.

Under certain conditions in a conversational mode run only, the $ in a command line may be omitted (if there is no line line number at the front of the input line and automatic numbering is off or there is no active file). The leading $ is required when running in batch mode. The beginning user is well-advised to start all command lines, batch or conversational, with a $ until he has gained a reasonable acquaintance with MTS.

Name: SIGNON

Purpose: To identify a user to the system

Prototype: $SIGNON ccid [batch specs] ['comment']

Usage: If the abbreviation $SIG is used, its meaning is taken in context: If no one is signed on, SIG means $SIGNON; if someone is signed on, $SIG means $SIGNOFF.

ccid is the user's identification assigned by the computing center. Complaint will be made if it is omitted or incorrect. If more than four characters are given for ccid, only the first four will be used.

If the given ccid has a password, the system will prompt for the user's password at the terminal or, in the case of a batch job, expect the card following the $SIGNON card to contain the password in columns 1-6.

batchspecs are any of the following batch job global limit specifications

TIME = n $\left\{ \frac{S}{M} \right\}$

PAGES = n

CARDS = n

or the copies specification

COPIES = n

(see the Batch Users' Guide).

Comment is any character string enclosed in quotes ('). The batch user should enter his name to facilitate output identification

Example: $SIG SID1

Name: SIGNOFF

Purpose: To notify the system of a user's departure

Prototype: $SIGNOFF [SHORT]

Usage: An abbreviated form of the sign-off information is typed at the terminal when 'SHORT' or S is given as a parameter.

Effect: All devices attached (and storage aquired) are released, all files are closed, and the system quiescently awaits the arrival of the next user on that terminal.

Example: $SIG S

Name: CREATE

Purpose: To create either a permanent or scratch file

Prototype:

```
                                 n          LINE
$CREATE Fname[SIZE={ nT}][TYPE={ SEQ  }][VOLUME=volname]
                     nP          SEQWL
```

Usage: Complaint will be made if Fname already exists.

The estimated SIZE of the file is given by:
n - the number of 40 byte lines.
nP - the number of 4096 bytes pages, or
nT - the number of 7294 byte tracks.

If the SIZE parameter is omitted, the default size of a permanent file is approximately 50-100 lines and the default size of a scratch file is approximately 650 lines.

The file TYPE may be line (LINE), sequential (SEQ) or sequential with line numbers (SEQWL). If the TYPE parameter is omitted, the default type is LINE.

The SIZE parameter may be as large as 12800 for LINE files, and as large as ? for SEQuential files. Complaint will be made if the size parameter given is larger.

Volname is the name of the (presumably private) VOLUME on which the file is to be created. If the VOLUME parameter is omitted, the file will be placed on a public volume where there is space available for it.

Explicit creation of scratch files is only necessary when the estimated file size exceeds the default size, or the file type is sequential, since scratch files are created automatically when named in a command.

Effect: The empty file Fname is created and becomes the active file, i.e., *AFD*

Examples:

```
#$ create -tempfile size=1000
#THE FILE"-TEMPFILE" HAS BEEN CREATED.

#$cr fit3stanza7 size=10p type=seq
#THE FILE "FIT35STANZA7" HAS BEEN CREATED.
```

Name: DESTROY

Purpose: To destroy a file

Prototype: $DESTROY Fname

Usage: Complaint is made if the parameter is missing, or the file specified does not exist or is a public file.

If the parameter is correct, confirmation is requested before a permanent file is destroyed. It is not requested for temporary files. The command is confirmed by the response OK or O.K. Any other response causes the command to be cancelled.

Effect: The file is deleted from the user's file catalog and space occupied by the file is returned to the public domain. The user is informed the file has been destroyed successfully.

Examples:

```
#$de my file
#FILE "MYFILE" IS TO BE DESTROYED. PLEASE CONFIRM.
? ok
#DONE.

#$destroy -tempfile
#DONE.
```

Name: EMPTY

Purpose: To discard the contents of a file without destroying the file

Prototype: $EMPTY Fname

Usage: Refer to DESTROY command USAGE

Effect: All current contents of the file Fname are discarded.

Examples:

```
#$empty sr
#FILE "SR" IS TO BE EMPTIED.  PLEASE CONFIRM.
? ok
#DONE.
```

Name: GET

Purpose: To obtain a file as the currently active file

Prototype: `$GET Fname`

Usage: Complaint is made if the parameter is omitted or the file is non-existant

Effect: The file Fname is opened and becomes the current file, i.e., it can be referenced as the pseudo-device *AFD*

Examples:

```
#$get myfile
#READY
```

Name: RELEASE

Purpose: To release the current active file (if there is one)

Prototype: `$RELEASE`

Effect: The current active file is closed and the pseudo-device *AFD* is disassociated with the file

Example:

```
$REL
```

Name: SOURCE

Purpose: To change the source of input lines

Prototype:

```
$SOURCE { FDname   }
        { PREVIOUS }
```

Usage: A one level pushdown of source devices is maintained. The previous source is restored as the current source if PREVIOUS is given as the parameter (even if a file name PREVIOUS exists).

Effect: *SOURCE* is reassigned to the file or device specified, i.e., the next input line will be taken from the file or device specified. The master source (*MSOURCE*) remains as the terminal (terminal mode) or the card reader (batch mode). Responses to error messages requiring user action are read from *MSOURCE* and attention interrupts occur on *MSOURCE* (terminal mode only).

Examples:

```
$SOURCE  -CMDS
$SOU *TAPE*
```

Name:	SINK
Purpose:	To change the destination or sink for "normal" output
Prototype:	$S<u>I</u>NK $\left\{\begin{array}{l}\text{FDname}\\\text{PREVIOUS}\end{array}\right\}$
Usage:	Analagous to SOURCE
Effect:	*SINK* is reassigned to the file or device specified. The master sink (*SINK*) remains as the terminal (terminal mode) or the line printer (batch mode). Error messages requiring user action are directed to *MSINK* (terminal mode only).
Examples:	$SI PREVIOUS $SINK PRINTSYSOUT

Name: COPY

Purpose: To copy a file

Prototype: $COPY [fromFDname] [[TO] toFDname]

Usage: If fromFDname is omitted, lines will be read from *AFD*. If toFDname is omitted, lines will be written on *SINK*. If fromFDname is omitted or follows toFDname, TO must precede toFDname. In the ambiguous case $COPY TO toFDname, TO is taken as the "noise" word and fromFDname will default to *AFD*.

Part of a file may be copied by specifying a line number range for fromFDname.

If an exact copy of a line file is wanted (i.e., each line under the same line number as in the original file), then the indexed modifier @I must be appended to toFDname.

Complaint will be made if either fromFDname or toFDname is a file that is non-existant, or is a device that is either not available or is the wrong type (output or input, respectively).

Effect: Lines are read sequentially from fromFDname, for the specified line number range (if any), until an EOF condition is read or the ending line number is reached. Line numbers are simulated for sequential files or devices.

For line files, lines are written on toFDname sequentially or indexed. If a line file is written sequentially, renumbering of lines occurs, however, the user can specify the beginning line number and increment for toFDname.

For sequential files or devices, lines are written on toFDname sequentially. If a line file is copied to a sequential file, line numbers are lost.

Examples:

```
$COPY SNARK TO BANDERSNATCH
$C FILE(5,20) TO EXACTCOPY@I
$C FILE TO LINEFILE(10,,10)
$C TO TOFILE FROMFILE
$C F1+F2(1,10)+(25,100) TO F3
```

Name: LIST

Purpose: To list a file with line numbers

Prototype: `$LIST [fromFDname] [[ON] toFDname]`

Usage: If fromFDname is omitted, lines will be read from *AFD*. If toFDname is omitted, lines will be written on *SINK*. If fromFDname is omitted or follows toFDname, ON must precede toFDname. In the ambiguous case $LIST ON toFDname, ON is taken as the "noise" word and fromFDname defaults to *AFD*.

Part of a file may be listed by specifying a line number range for fromFDname.

Complaint will be made if either fromFDname or toFDname is a file that is non-existant or is a device that is not available or is the wrong type (output or input, respectively).

Effect: Lines are read sequentially from fromFDname, for the specified line number range (if any), until an EOF condition is read or the ending line number is reached. Line numbers are simulated for sequential files or devices. The line number is converted to 12 EBCDIC characters and appended on the front of each line. This extended line is written on toFDname (according to the modifiers given in the command).

Examples:

```
$LIST
$L FIT3stanza7 ON PTR1
$L FILE(20,32)
$LIST OFILY(117,117)
$LIST ON LISTFILE
$L LINEFILE(1,5)+(LAST-5)
```

Name: NUMBER

Purpose: To start automatic numbering of input data lines coming from *SOURCE* and being written to *AFD*

Prototype:

```
$NUMBER  {[starting number][[,]increment]}
                     CONTINUE
```

Usage: To use automatic line numbering with sequential files, the $SET SEQFCHK=OFF command must be issued first.

The starting line number is given by STARTINGNUMBER. If the parameter is omitted, line numbering begins at 1. The starting number can take any of the forms

```
line number
LAST
LAST ± line number
```

where LAST is the line number of the last line in the current active file. If the file is empty, the value of LAST is zero.

The line number increment is given by INCREMENT. If the parameter is omitted, line numbers are incremented by 1. If only the increment is given, it must be preceeded by a comma.

Effect: Provided there is a current active file, any input line not recognized as a command line has a line number automatically assigned to it before it is written in the current active file.

Example:

```
#$number
#     1_
#     2_

#$n 10,2
#    10_
#    12_
```

Name: UNNUMBER

Purpose: To stop automatic numbering of input lines

Prototype: $UNNUMBER

Effect: Any input line, not recognized as a command line or a data line (i.e. has a line number), is consdered as an invalid command.

Example:

```
#$NUMBER
#     1_
#     2_
#     3_$UNNUMBER
#$LIST
```

Name:	COMMENT
Purpose:	To allow insertion of comments
Prototype:	$COMMENT any text
Effect:	The command is ignored and echoed to *MSINK*
Example:	$COM THIS IS A COMMENT COMMAND

Name: ENDFILE

Purpose: To provide an end-of-file indication on *SOURCE*

Prototype: $ENDFILE

Usage: Note there is no abbreviation allowed

In batch mode, this is the only indication of an end-of-file. In interactive mode, the cent character ¢ has the same effect as a $ENDFILE command.

The command appears in the source stream *SOURCE*:

(1) following source decks, eg FORTRAN, Assembler, PL/1

(2) following object decks

(3) following data decks

(4) following data lines on *SOURCE* for COPY or LIST commands

Example:

```
$RUN  *FORTG
        .
     FORTRAN SOURCE
        .
$ENDFILE
$RUN -LOAD# 5=*SOURCE*
        .
     data
        .
$ENDFILE
```

Name: RUN and LOAD

Purpose: To load a program and, in the case of RUN, initiate execution.

Prototype:

```
{$RUN }  [objectFDname]  [MAP[=mapFDname] [XREF]]
 $LOAD
         [iospecs] [limit specs] [PAR=parameter]
```

Usage: The object deck is loaded from objectFDname, or *SOURCE* if objectFDname is omitted.

The ~~MAP parameter~~ mapFDname specifies the file or device on which the loader is to write the load map. If omitted, ~~no load map is written~~ *SINK* is assumed. The XREF parameter specifies that a cross reference of external symbols occuring in loader programs is to be produced in addition to the load map.

Error comments (if any) are directed to *MSINK*

Assignment of logical I/O units, used by the user in his program, to files or physical devices, to be used during execution, is specified by iospecs (described in section 2.0). Where no specifications are stated, the following default assignments occur:

```
SCARDS = *SOURCE*
SPRINT = *SINK*
SPUNCH = *PUNCH* (batch mode only)
SERCOM = *MSINK*
GUSER = *MSOURCE* (interactive mode only)
```

Local limits for CPU time, pages printed and cards punched are specified by limit specs (described in section 2.0).

Parameters to be passed to the program on initiation of execution follow the PAR=. The parameter list is terminated by a blank.

Effect: The loader is called to load the object program into a region in core. If there are unresolved external symbol references after loading from objectFDname loading will be continued from *LIBRARY (the system library). Only those parts of *LIBRARY required to resolve the references will be loaded. If there are still unresolved external symbol references, a fatal loading error exists.

If files are non-existant or devices not available, an error comment is produced and the logical I/O unit referring to the unavailable file or device is set up in such a way that the first time the program, being executed, refers to the logical I/O unit, either the user is given a chance to respecify the name (interactive mode) or execution is terminated (batch mode).

The parameter (set up by the PAR= keyword specification) is passed as follows. Register 1 contains the location of a full word address constant. The address constant is the location of a half-word count (halfword aligned) which is immediately followed by an EBCDIC character region (of the byte-length specified in the count) which continas the parameters. The left most bit of the address constant is 1 (standard OS convertion).

If LOADing, control is returned to the user in command mode, the program can be displayed and/or altered, and execution begun with the $START command.

If RUNing, and there were no fatal loading errors, the comment 'EXECUTION BEGINS' is printed and control is transferred to the entry point of the program by calling it with the entry point address in GR15, the return address in GR13, the save area location in GR13, and the parameter location in GR1 (standard OS convertion).

If the program terminates exectuion by restoring the registers and returning via GR14, the comment "EXECUTION TERMINATED" is printed and the RUN command is terminated.

All storage, files and devices used are automatically released unless the user has issued the $SET UNLOAD=OFF command or execution was not terminated normally (for example the program calls ERROR) and object FDname is not a public file, or the user has issued the $SET LIBR=OFF command, when RUNning a public file.

If storage, files and devices are not released, the user can use $DISPLAY, $ALTER, and $START to debug the program.

Examples:

```
#$run -load#
#EXECUTION BEGINS
#EXECUTION TERMINATED

#$run objdeck+*ssplib map=mapfile 5=inputfile
#EXECUTION BEGINS
ATTENTION INTERRUPT AT 7050347E
#$start
#EXECUTION TERMINATED

#$load objdeck
#
```

Name: START

Purpose: To initiate execution of a program following either LOADing or an interrupt.

Prototype: $START [[AT] locn] [MAP=mapFDname] [iospecs] [limitspecs]

Usage: The address to which control is to be given is specified by locn (described in section 2.0). Since this replaces the right hand 32 bits of the PSW, the displacement given in locn specifies the instruction length code, the condition code, and the program mask as well.

The user can re-designate the destination of the load map with the MAP parameter. This is only useful if the program is loading dynamically.

The user can re-assign logical I/O units to files or devices by iospecs (described in section 2.0).

The user can re-specify limits for cpu time, pages printed and cards punched by limitspecs (described in section 2.0).

Effect: A 32 bit address is computed from the locn specification, and replaces the right hand 32 bits of the PSW. If locn was omitted, the PSW remains unaltered.

If logical I/O units have been reassigned, the files and devices originally assigned are closed, and the newly assigned files and devices are opened.

Examples:

```
#$r -load#
#EXECUTION BEGINS
THIS IS PROGRAM OUTPUT WHICH I DON'T REALLY WANT ON
THE TERMINAL.
ATTENTION AT 7050347E
#$start sprint=file
#EXECUTION TERMINATED

#$start at rf=20800 258
#$S at rf=20800   28000258 (program is started at
     20A58 with condition code set to 2, fixed point
     overflow interruption enabled and the other
     program mask interrupts disabled)
```

COMMAND DESCRIPTION

Name: UNLOAD

Purpose: To release storage and devices from the previous $LOAD or $RUN if the execution did not terminate normally (normal termination is via the subroutine SYSTEM or by RETURNing).

Prototype: $UNLOAD

Name: ALTER

Purpose: To alter the contents of a general register, floating point register, or specified core location(s).

Prototype:

```
                                 hhhh
                  GRx            x'hhhh'
Prototype: $ALTER { FRy }       { C'xxxx'} ...
                  locn           F'yyyy'
                                 H'yyyy'
```

Usage: Each alteration requires a pair of parameters, the first specifying what is to be altered and the second specifying the new contents. Any number of items may be altered with a single ALTER command.

Specification of GRx, FRy and locn is described in Section 2.0. Remember that any occurrence of a relacation factor in a locn specification sets a local relocation factor value which remains in effect for the duration of the command or until reset by a subsequent relocation factor specification.

The new contents are specified by any one of the following constant expressions:

<u>hexadecimal</u> hhhh or X'hhhh'
<u>character</u> C'xxxx' Any EBCDIC character including blank may be given between the delimiting primes; a prime in the character string must be represented by two consecutive primes.
<u>fullword decimal</u> F'yyyy' or <u>halfword decimal</u> H'yyyy' consist of a sign followed by the decimal digits all enclosed by primes. The "+" sign is optional; the "-" sign is required. Decimal constants may <u>not</u> be specified for floating point registers.

Complaint is made if invalid register numbers, addresses or constant expressions are specified.

Effect: <u>General registers</u> are altered as follows:
A character constant is truncated or padded with trailing blanks to four bytes or characters and placed, left justified, in the register.

The integer value of a hexadecimal constant (consisting of one to eight hexadecimal digits including leading zeros) is loaded into the register.

The integer value of a decimal constant is loaded into the register.

Floating point registers are altered as follows:

A character constant is truncated or padded with trailing blanks to eight bytes or characters and placed, left justified, in the register.

A hexadecimal constant is truncated or padded with trailing zeros to eight bytes and placed, left justified with leading zeros retained, in the register.

Core storage is altered as follows:

A character constant is placed, one character per byte in consecutive core locations.

A hexadecimal constant is placed, two hexadecimal digits per byte with leading zeros retained, into consecutive core locations. If an odd number of hexadecimal digits is given, the last byte of core storage altered will have bits 4-7 set to zero.

The integer value of a decimal constant is loaded, without regard to boundary alignment, into the fullword (or halfword) core location whose high order byte is specified by locn.

Examples:

```
$ALTER GR3 1A3E0 FR6 X'41104'
$ALT RF=18AE2 2BE X'D502CC7E6000' 3E0 X'05EF GRA 0
$A RF=1A800 AEC F'-1000' RF=19600  2B6 C'DON''T DO IT'
```

Name:	DISPLAY and DUMP
Purpose:	To display the contents of general registers, floating point registers, and/or specified core locations.
Prototype:	$DISPLAY [ON FDname] [format specs] contentspecs $DUMP [ON FDname] [format specs]

Usage: DISPLAY is used when the user wants to specify what is to be displayed, while DUMP is used to cause the general registers, floating point registers, and core storage associated with the job to be displayed.

If FDname is omitted, output is directed to *SINK*. Complaint is made if FDname specifies a non-existant file or a device that is unavailable.

The only restriction on the order of parameters in the command line is that "ON FDname" must appear first if it appears at all.

The format of the display is specified by formatspecs which may be any of the following:

HEX	hexadecimal conversion
NOHEX	
MNEM	mnemonic conversion
NOMNEM	
EBCD	EBCDIC conversion
NOEBCD	
SP1	single spacing
SP2	double spacing
ORL=S	short output record (70 characters)
ORL=L	long output record (130 characters)

If not specified, the following default format specs occur:

HEX	
SP1	
ORL=L	for line printers
ORL=S	for terminals

Format specifications remain in effect until a subsequent entry in the command line changes the specification.

The user specifies what is to be DISPLAYED by contentspecs which may be any of the following:

GRx	where x is the character "S" if all general registers are to be displayed.
FRy	where y is the character "S" if all floating point registers are to be displayed.

locn (described in section 2.0) If a local relocation factor is specified in locn, it remains in effect for the remainder of the command unless subsequently changed. A range of displacements can be given in locn by

hhhh...hhhh

PSW Program staus word

VMSIZE the current size of the user's virtual memory (in pages).

One ambiguity may occur. EBCD can represent either a format parameter or a hexadecimal address displacement. It is interpreted as a format parameter. To display the single location EBCD,use OEBCD.

If the last entry in a command line is not a content parameter, an appropiate comment is made.

Complaint is made if the register number is illegal.

Effect: General registers, floating point registers and the PSW are displayed in labelled hexadecimal format.

Blocks of storage are displayed by calling the subroutine SDUMP.

Description of the dump format is contained in the writeup on SDUMP (refer to SYSTEMS SUBROUTINE MANUAL).

Whenever a contentspec is encountered in a DISPLAY command line it is processed immediately using the format parameters and relocation factor in effect at that time.

The format parameters in effect at the end of a DUMP command line govern the format of the core storage output information.

Examples:

```
$DISPLAY GR3 FRS EBCD 18E08...18FA6
$D ON DISPLAYFILE ORL=L GRS FR6
$DUMP
$DU HEX EBCD SP2
```

Name HEXADD, HEXSUB

Purpose: To perform hexadecimal addition and subtraction

Prototype: $\left\{\begin{matrix}\$\underline{H}\text{EXADD}\\ \$\underline{\text{HEXS}}\text{UB}\end{matrix}\right\}\left\{\begin{matrix}\text{hhhh}\\ \text{GRX}\end{matrix}\right\}\left\{\begin{matrix}\text{hhhh}\\ \text{GRX}\end{matrix}\right\}$

Usage: The hexadecimal numbers are entered with one or more intervening blanks as delimiters. The contents of a register is used in the arithmetic operation if GRX is specified.

Effect: The hexadecimal numbers or the contents of the specified registers are added or subtracted.

The results of a $HEXADD command appear in the form

```
SUM=XXXXXXXX
```

The results of a $HEXSUB command appear in the form

```
DIFF=XXXXXXXX
```

Overflows are ignored and negative results are given with a minus sign preceeding the absolute value of the difference.

Example:

```
#$h 1A2 2E81D
#SUM = 2E9BF
```

Name: SET

Purpose: Set various global switches and quantities.

Prototype: $SET kywd=quan...

kywd	quan	Comments
AFDECHO	ON OFF	If ON all lines written to the active file in command mode will be echoed to *SINK* and *MSINK*, the same as commands are echoed. Each line echoed will be preceeded by its line number in the active file.
CASE	UC LC	If UC is in effect, all data lines read by the MTS monitor will have lower case letters converted to upper case. Useful for terminals like 2741 and 1050 which have both cases, to avoid having to use the shift key so much.
CONTCHAR	character -	One character specifying the character that indicates that a command line is continued if it appears as the last character in a line.
DEVCHAR	character >	One character specifying the character that indicates that a following FDname is a device, not a file.
ECHO	ON OFF	If SOURCE device and SINK (orMSINK) device differ command lines from SOURCE are echoed onto SINK (or MSINK). A $SET ECHO=OFF will turn off the echoing and a $SET ECHO=ON will restore it.
ENDFILE	ON OFF	If ON, a $ENDFILE line will be recognized as an end of file whenever it is read, not just from *SOURCE* or *MSOURCE* as is normally the case.
ERRORDUMP	ON OFF	If ON and an execution in batch mode terminates abnormally a dump is given. This has no effect in terminal usage.
FILECHAR	character #	One character specifying the character that indicates that a following FDname is a file, not a device.

IC	ON OFF	If ON implicit concatenation is active (see FILES AND DEVICES manual.) If OFF no check is made for "$CONTINUE WITH" lines and they are treated as any other lines. This can be overridden by the IC modifier on I/O operations
LFR	ON OFF	If ON, storage occupied by library files during a RUN is always released when they return to the system, no matter what the reason. If OFF, library files are treated the same as any other file in this respect. (See also $RUN description)
LIBR	ON OFF	Normally the file *LIBRARY is searched after loading a program if there are any unresolved external symbols. If LIBR is set OFF then this automatic search will not be made.
LNS	character ,	One character specifying the line Number Separator, i.e., that character which, if it terminates the line number at the beginning of an input line, is not considered as part of the line but only as a separator. Hence a line commencing with numeric information may be easily entered. E.g., a line beginning: 174, 10LINE = 2
PFX	ON OFF	Normally a prefix of either one character or a line number is printed at the front of each line input from or output to a user's terminal. If PFX is set OFF then no prefixes will be printed.
PW	character string	Any sequence of zero to six characters none of which are blank. If at least one character is given the character string becomes the user's password and must be given correctly before the user is allowed to signon with his user id. If zero characters are given the password is no longer required. See description of $SIGNON for how to specify the password when signing on.

RF	hhhh GRx	This sets a global relocation factor quantity which is used in DISPLAY and ALTER commands. The relocation factor is zero initially. If GRx is specified the relocation factor is set to the contents of the specified general register.
SCRFCHAR	character -	One character specifying the character that indicates that a following FDname is a scratch file.
SEQFCHK	ON OFF	Normally an attempt to do an indexed operation on a sequential file or an attempt to do a sequential operation starting at other than line 1 on a sequential file will cause an error message to be generated. If SEQFCHK is set OFF then the message will not be issued and the operation will be performed as if not indexed.
SHFSEP	character :	One character specifying the character used to separate the userid from the file name when referring to a shared file.
SYMTAB	ON OFF	If ON the loader symbol table is retained whenever a program has been loaded, allowing external symbols used in a program to be used by MTS and user programs.
UNLOAD	ON OFF	If ON storage and devices from previous $LOAD or $RUN will automatically be released. If OFF they must be released using the $UNLOAD command.

More than one parameter may be given on a single $SET command. The parameters should be separated by blanks.

Example: $SET PW=NEW CASE=LC

Name:	ERRDUMP
Purpose:	To allow automatic dumps in batch mode
Prototype:	$ERRDUMP
Usage:	The command is effective in batch mode only. It is equivalent in effect to the $SET ERRORDUMP=ON command.
Effect:	If an executing program terminates abnormally, a dump of the registers and storage is given.
Example:	$ERR

Name:	MODIFY
Purpose:	A synonym for ALTER

Name:	RESTART
Purpose:	A synonym for START

Signing on and off

```
$SIGNON ccid [limitspecs] [HASPspecs]
     ['comments']

$SIGNOFF [SHORT]
```

Creating, destroying and emptying files

```
                        n              LINE
$CREATE Fname [SIZE={ nP}] [TYPE={ SEQ    }]
                        nT             SEQWL
     [VOLUME=volname]

$DESTROY Fname
                         } confirmation  OK
$EMPTY Fname
```

Assigning pseudo-devices

```
$GET Fname

$RELEASE

$SINK { FDname
        PREVIOUS }

$SOURCE { FDname
          PREVIOUS }
```

Copying and listing

```
$COPY [{ fromFDname }] [[TO]{ toFDname }]
           *AFD*                *SINK*

$LIST [{ fromFDname }] [[ON]{ toFDname }]
           *AFD*                *SINK*
```

Automatic line numbering

```
             b         |
$NUMBER { [{ 1 }] [[,]{ 1 }] }
           CONTINUE

$UNNUMBER
```

Miscellaneous

```
$COMMENT any text

$ENDFILE
```

Loading and executing programs

```
$RUN [{ FDname
        *SOURCE* }] [MAP=FDname[XFER]]
     [iospecs] [limitspecs]
     [PAR=paramters]

$LOAD  see $RUN

$START [[AT]locn] [MAP=FDname]
     [iospecs] [limitspecs]

$UNLOAD
```

Examining and changing core

```
                   hhhh
          GRx      X'hhhh'
$ALTER { FRy } { C'xxxx' }
          locn     F'yyyy'
                   H'yyyy'


                                 GRx
                                 FRy
$DISPLAY [formatspecs] { locn} ...
                                 PSW
                                 VMSIZE

$DUMP [formatspecs]

$HEXADD { hhhh } { hhhh }
          GRx      GRx

$HEXSUB { hhhh } { hhhh }
          GRx      GRx
```

Setting system parameters

```
         [AFDECHO={ ON|OFF}] [CASE={UC|LC}]
         [CONTCHAR=-] [DEVCHAR=>]
         [ECHO={ ON|OFF}] [ENDFILE={ ON|OFF}]
         [ERRORDUMP={ON|OFF}] [FILECHAR=#]
$SET     [IC={ ON|OFF}] [LFR={ ON|OFF}]
         [LIBR={ ON|OFF}] [LNS=,]
         [PFX={ ON|OFF}] [PW=password]
         [RF={ hhhh|GRx}] [SCRFCHAR=-]
         [SEQFCHK={ ON|OFF}] [SHFSEP=:]
         [SYMTAB={ ON|OFF}] [UNLOAD={ ON|OFF}]

$ERRDUMP
```

iospecs

```
  SCARDS
  SPRINT
 {SPUNCH}=FDname
  SERCOM
  GUSER
  0 thru 9
```

limitspecs

```
           S
  TIME=n[ - ]
           M
  PAGES=i
  CARDS=i
```

limit defaults

```
  30 seconds
  50 pages
   1 card
```

locn

```
          hhhh
  [RF={        }]hhhh
          GRx
```

formatspecs

```
  HEX |NOHEX
  MNEM|NOMNEM
  EBCD|NOEBCD
  SP2 |SP1
        L
  ORL={   }
        S
```