M T S

The Michigan Terminal System

The Michigan Terminal System

Volume 1

Reference R1001

Systems Edition

November 1991

University of Michigan
Information Technology Division
Consulting and Support Services

CONTENTS

OBSOLETE AND INTERNAL MTS COMMANDS

This section contains descriptions of obsolete MTS commands and privileged MTS commands and options that are generally of use only to systems programmers.

ALTER, MODIFY

MTS Command Description

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

Prototype: ALTER [CLS=xxx] {GRx | FRx} value ... ...
MODIFY [CLS=xxx] {GRx | FRx} value ... ...

CLS=xxx

If CLS=xxx is specified, then any following general or floating-point register specifications are altered for that command language subsystem (CLS). This is normally useful only for system programmers.

CONTROL

MTS Command Description

Purpose: To control the operation of certain types of files and devices.

Prototype: CONTROL FDname control-command

For *PRINT*, *PUNCH*, *BATCH*:

PRIO=n

Changes the batch-monitor priority to “n”, where “n” is an integer from 0 to 15. This option is valid only for privileged userIDs.

CREATE, DUPLICATE

MTS Command Description

Purpose: To create or duplicate a file.

Prototype: CREATE filename [VOLUME=volname]

DUPLICATE oldname [AS] newname [VOLUME=volname]

VOLUME=volname

The VOLUME parameter specifies the volume on which the file is created. The parameter “volname” is the label of the disk volume. If this parameter is omitted, the file is located on any public volume where there is sufficient space available.

DISPLAY

MTS Command Description

Purpose: To display the contents of a general registers, floating-point registers, or the program status word.

Prototype: DISPLAY [CLS=xxx] [format] {GRx | FRx | PSW | item}

CLS=xxx

If CLS=xxx is specified, then any following general or floating-point register or PSW specifications are displayed for that command language subsystem (CLS). This is normally useful only for system programmmers.

item

{AFDNAME | *AFD*}

AFDNAME or *AFD* displays the name of the current active file or device.

INSULT[(MILD | MEDIUM | STRONG | VERY)]

Display an insult. The default is INSULT(STRONG). GRELBER is a synonym for INSULT.

JOKE[(n)]

Display a joke(s). “n” is the number of jokes displayed. The default is n=1.

DUMP

MTS Command Description

Purpose:	To display the contents of general registers, floating-point registers, the program status word, and the virtual memory locations associated with the currently loaded program.
Prototype:	DUMP [{SYS | CLS=xxx}] [format] ...

{SYS | CLS=xxx}

If SYS is specified, a dump of the system-allocated storage for the user's task is produced. If CLS=xxx is specified, a dump of the storage allocated for that command language subsystem (CLS) is produced. This is normally useful only for system programmers.

ERRORDUMP

MTS Command Description

Purpose: To allow automatic program dumps in batch mode in the event of an abnormal program termination.

Prototype: ERRORDUMP

Program Key: *MTS.ERRORDU

Description: If an executing program terminates abnormally, a dump of the registers and program storage region is given. Common abnormal terminations are a program interrupt, a call to the subroutine ERROR, and the exceeding of global or local time, page, or card estimate. This command is equivalent to the command SET ERRORDUMP=ON.

If a global time or page limit is exceeded while producing a program dump, the dump is completed and the CPU time and page costs are charged to the user's account.

This command has no effect in conversational mode since dumps are never automatically produced. Terminal users may use the DUMP, DISPLAY, or debug mode DISPLAY commands to inspect all or parts of virtual memory.

Description of the format of the dump is given in the section "Using Errordumps and Load Maps" in *MTS Volume 5: System Services*, Reference R1005.

A *symbolic* program dump may be obtained via the symbolic debugging system facilities in a similar manner. For details, see *MTS Volume 13: The Symbolic Debugging System*, Reference R1013.

FILESTATUS

MTS Command Description

Purpose: To obtain file information, access information, and/or catalog information for a file, or information about pseudodevice names.

Prototype: FILESTATUS [name] [format] [information]

information

PROT

If OFF, this indicates that the file contains a system program that can be run with protection off.

GET

MTS Command Description

Purpose:	To establish a file or device as the currently active file.
Prototype:	GET FDname
Program Key:	*MTS.GET
Description:	The “FDname” parameter specifies the name of the file or device to become the currently active file. The file or device is opened (and locked for reading) and the pseudodevice *AFD* is associated with that file or device. An error comment is produced if the parameter is omitted, the file or device does not exist, or the user does not have access to it. The previously active file is released even if one of the above error conditions occurs. The line number and increment for automatic line-numbering are reset to 1. If the currently active file is modified, it is implicitly locked for modification and remains locked for modification until it is released.

HEXADD

MTS Command Description

Purpose:	To perform hexadecimal addition in command mode.
Prototype:	HEXADD operand operand **...**

One or more operands separated with intervening blanks are given with the command. Each operand may be given in one of the two following forms:

hhhh	A hexadecimal number to be used as the operand.
GRx	A general register whose contents is to be used as the operand.
RF	The current global relocation factor.

Program Key: *MTS.HEXADD

Description: The hexadecimal numbers or the contents of the registers specified by the operands are added. Overflows are ignored.

The results of the HEXADD command appear in the form:

SUM = xxxxxxxx

Example:

```
HEXADD 1A2 2E81D
```

In this example, the hexadecimal numbers 1A2 and 2E81D are added together to produce the sum 2E9BF.

HEXSUB

MTS Command Description

Purpose:	To perform hexadecimal subtraction in command mode.
Prototype:	<u>HEXS</u>UB operand operand **...**

One or more operands separated with intervening blanks are given with the command. Each operand may be given in one of the two following forms:

hhhh	A hexadecimal number to be used as the operand.
GRx	A general register whose contents is to be used as the operand.
RF	The current global relocation factor.

Program Key: *MTS.HEXSUB

Description: The second and all following operands are subtracted from the first operand. Negative results are given with a minus sign preceding the absolute value of the difference.

The results of the HEXSUB command appear in the form:

DIFF = xxxxxxxx

Example:

```
HEXS 2E9BF 1A2
```

In this example, the hexadecimal number 1A2 is subtracted from 2E9BF to produce the difference 2E81D.

INFO

MTS Command Description

Purpose: To interactively examine loader and low-core symbol tables.

Prototype: INFO [command]

Program Key: *INFO

Description: INFO is a CLS that allows the user to interactively examine loader and low-core symbol tables; initially, the IPL loader psect SYSSYMTB and LCSYMBOL, respectively. The program is a command-language subsystem (CLS) processor that reads commands from *SOURCE* and writes on *SINK*.

INFO is available as a staff-only MTS command which may be invoked as

    INFO [command]

If a command is given, only the single command will be processed, after which the program will return to MTS (a one-shot invocation). If no command is given, the program will enter command mode and prompt the user with the prefix "<". The following rules apply to command usage:

(1) A command starts with the first nonblank character; thus, commands need not start in column 1.

(2) Specification of the command name and command operands should contain no embedded blanks.

(3) Multiple operands must be separated by one or more blanks.

(4) Any command may be abbreviated by entering only an initial substring. The minimal initial substring allowed is underscored in each command description.

(5) A command line beginning with an asterisk (*) is treated as a comment and is not processed.

(6) A command line beginning with a dollar sign ($) is treated as an MTS command and is passed to MTS via a call to the MTS subroutine CMD.

The following table summarizes the available commands. In what follows, [...] denotes optional items and {... | ...} denotes alternatives.

Command	*Operand*
DEFINE	symbol ...
EJECT	[ON FDname]
EXPLAIN	command ...
FIND	symbol ...
HELP	
INFO	[ON FDname] {symbol | address | # | * | >pcid | <BLANK>} ...
JOB	[ON FDname] {symbol | *} ...
LCS	[ON FDname] {symbol | address | # | * | *@A | *@N} ...
LCSPR	[ON FDname] {symbol | # | * | CXD} ...
MAP	[ON FDname]
MCMD	mts-command
MTS	
QUERY	{symbol | address} ...
RETURN	
SET	option ...
STOP	
*comment	
$mts-command	

Command: DEFINE symbol ...

Example: DEFINE AXWCMPB2 AWXCMPA2

Explanation: The DEFINE command is intended for use by TMTS users. It copies the definition for "symbol" from the real LCSYMBOL to the first slot in TMTS LCSYMBOL. If invoked from a MTS task, the user must first make the TMTS LCSYMBOL the current LCS table via SET LCS=addr, where "addr" is the base address of the TMTS LCSYMBOL table, e.g.,

```
SET LCS=314330
DEFINE AWXCMPB2 AWXCMPA2
```

If the CLS is invoked from a TMTS job, then it is neccessary to specify the location of the real LCSYMBOL via SET REALCS=xxx. For example:

```
SET REALCS=1424F8
DEFINE AWXCMPB2 AWXCMPA2
```

The DEFINE command should not be used if the TMTS dsect has a different format or if the routine specified by "symbol" calls any MTS subroutines that have been changed in the TMTS system.

Command: EJECT [ON FDname]

Example: EJECT ON *PRINT*

Explanation: The EJECT command forces a page eject on the specified output device.

Command: EXPLAIN command ...

Example: EXPLAIN INFO MAP

Explanation: The EXPLAIN command will print a simple explanation of “command”.

Command: FIND symbol ...

Example: FIND GRAB3270

Explanation: The FIND command searches the low-core symbol tables LCSYMBOL, SYSDEFS, <EFL>, PL1SYM, <FIX>, SPIRLCS, and INTSUBS (in that order) for the specified symbol. If it is found, the value of the symbol is printed along with the name of the LCS table in which it was found. The program will search for all occurrences of “symbol” if the SET option ALL is ON (see the SET command description below). By default, ALL is OFF.

Command: HELP

Example: HELP

Explanation: The HELP command prints a list of all legal commands.

Command: INFO [ON FDname] {symbol | addr | # | * | >pcid | <BLANK>} ...

Example: INFO $POOL 1717B8

Explanation: The INFO command obtains information from the current loader psect, initially the IPL psect SYSSYMTB. Each parameter causes a call to LOADINFO for that item which returns pertinent loader information (e.g., if “symbol” is a control section, then its origin, relocation factor, and length are printed). The following are valid command operands:

symbol	LOADINFO is called with the specified symbol.
addr	The csect containing “addr” is printed along with the closest entry point name and value.
#	The total number of symbols in the loader table is printed.
*	Loader information for all symbols in the table is printed.
>pcid	Information for the specified private control section (pcid=1,2,...) is printed.

<BLANK>
Information for the most recent blank common section is printed.

By default, the loader table examined is the IPL psect SYSSYMTB. This may be changed with the SET option PSECT (see the description of SET below).

Command:	JOB [ON FDname] {symbol | *} ...
Example:	JOB TMTS
Explanation:	The JOB command examines JOBLST. The following are valid operands:

symbol	searches JOBLST for "symbol" and prints the entry if found.
*	prints out the job list.

Command:	LCS [ON FDname] {symbol | address | # | * | *@A | *@N} ...
Example:	LCS GRAB3270 2DA9C0
Explanation:	The LCS command obtains information from the current low-core symbol table, which is initially LCSYMBOL, but may be changed with the SET option LCS (see the description of SET below). The following are valid operands:

symbol	The current LCS table is searched for "symbol".
address	The symbols whose values bound "address" in the current LCS table are printed.
#	The number of entries in the table are printed.
*	The entire table is printed in a "nice" format.
*@A	The entire table is printed sorted by address.
*@N	The entire table is printed sorted by name.

The location of a LCS entry for the "symbol" operand will be printed if the SET option LOCN is ON. By default, LOCN is OFF. Slots in the current LCS table may be filled in with the SET option SLOT=(symbol,value), but if the table is in shared VM, only userID MTS is allowed to do this.

Command:	LCSPR [ON FDname] {symbol | # | * | CXD} ...
Example:	LCSPR IHEQLSA CXD
Explanation:	The LCSPR command is similar to the LCS command except that it examines the current low-core pseudoregister table, initially the PL/I table IHEPRD, but changeable via SET LCSPR=name. The valid command operands are:

symbol	Searches the current LCSPR table for “symbol”.
*	The entire table is printed.
#	The total number of entries in the current LCSPR table is printed.
CXD	The PR cumulative length of the current LCSPR table is printed.

Command:	MAP [ON FDname]
Example:	MAP ON *PRINT*
Explanation:	The MAP command prints a map for the current loader symbol table.

Command:	MCMD mts-command
Example:	MC CONTROL *PRINT* HOLD
Explanation:	The MCMD command calls the MTS subroutine CMD with the command “mts-command”. Alternatively, the user may issue the command “$mts-command” for the same effect.

Command:	MTS
Example:	MTS
Explanation:	A return to MTS is made without unloading the program. The program may be restarted with $RESTART if invoked via $RUN, or by “# x” (where “x” may be any character) if invoked as a CLS.

Command:	QUERY {symbol | address} **...**
Example:	QUERY 1C02BA INTSUBS
Explanation:	The QUERY command combines the functions of the INFO, FIND, and LCS commands. All information available for the specified symbol or address is printed. The IPL loader symbol table is first searched for specified parameter. If the symbol or address is not in the IPL symbol table (SYSSYMTB), then all known low-core symbol tables are searched (LCSYMBOL, SYSDEFS, PL1SYM, etc.). If an address is specified, the program will search all LCS tables for an entry whose whose value is closest (less than or equal) to the given address.

Command:	RETURN

Example: RET

Explanation: The RETURN command is the same as the STOP command.

Command: SET option ...

Example: SET LCS=2F3D88 LOCN=ON

Explanation: The SET command may be used to change various program defaults. The following are valid SET options; default values are underscored.

```
ALL={ON | OFF}
ATTN={ON | OFF}
DOTS={ON | OFF}
HDR={ON | OFF}
LCS={LCSYMBOL | SYSDEFS | <EFL> | PL1SYM | <FIX> |
        SPIRLCS | INTSUBS | address | lcsname | USERLCS}
LCSPR=lcspr-name
LOCN={ON | OFF}
MAPINFO={ON | OFF}
OUTPUT=FDname
ORL=length
PDMAP={ON | OFF}
PGNT={ON | OFF}
PRMAP={ON | OFF}
PSECT={SYS | * | address}
REALCS=address
SIN={* | xx}
SIN{<= | >= | < | >}xx
SLOT=(symbol,value)
```

The option ALL affects the FIND command; if ON, FIND will search for all occurrences of the specified symbol. ATTN and PGNT control the program's attention trapping and program interrupt trapping facilities, respectively. DOTS, MAPINFO, PRMAP, and PDMAP are equivalent to their MTS counterparts. HDR may be set OFF to suppress the header printed by the INFO command. The option LCS changes the current low-core symbol table used by the LCS command; USERLCS is the assigned name for the low-core symbol table initially specifed by "LCS=address". The program's output may be redirected via the OUTPUT option.

The current loader psect used by the INFO and MAP commands may be changed via the PSECT option; SYS is the IPL table, * is <LODTAB> (may not be around if the program is invoked as a CLS). The SIN keyword can be used to filter the symbols printed by the INFO command, e.g., if SIN=02, only symbols with storage index numbers wqual to 02 are printed. Other keyword separators are recognized; e.g., SET SIN>00 SIN<04 would instruct the INFO command to print only those symbols whose storage index numbers lie in the range (1,3) inclusive.

Finally, a slot (XL12'0') in the current low-core symbol table may be filled with the option SLOT=(symbol,value), but only if the userID is MTS if the LCS table is in shared VM. This option is useful to TMTS users, when it is desired to insert a value for GRAB3270 into the TMTS LCSYMBOL. Also for TMTS users, the option "REALCS=address" is used to specify the base of the real LCSYMBOL table, where "address" is available from the system map or INFOCLS command "INFO LCSYMBOL". This address is needed by the DEFINE command if the CLS is invoked from a TMTS job.

Command: STOP

Example: STOP

Explanation: The INFO command is terminated. This is also the same as the RETURN command. An end-of-file is also equivalent.

MOUNT

MTS Command Description

Purpose: To mount magnetic tapes or UMnet/Michnet Computer Network connections.

Prototype: MOUNT [request [;request] ...]

For magnetic tapes:

UAM={ON | OFF}

UAM=OFF disables the restriction against mounting magnetic tapes during unattended mode operation.

NLC={ON | OFF}

NLC=ON disables label checking when mounting magnetic tapes. This option is valid only from certain userIDs.

NEW, NEW2, NEW3

MTS Command Description

Purpose:	To make a new version of an MTS command (CLS) temporarily available.
Prototype:	NEW [options] NEW2 [options] NEW3 [options]
Program Keys:	*NEW, *NEW2, and *NEW3
Description:	These commands may be used to make experimental versions of a CLS available. The command SET VERSION(command)=NEW is an alternative to the NEW command.

NUMBER

MTS Command Description

Purpose: To enable automatic numbering of input lines being read from *SOURCE* in MTS command mode and being written to the currently active file (*AFD*).

Note: The EDIT and the COPY commands provide much simpler and safer methods of inserting, deleting, or changing data lines in a file. These methods are preferred over the use of the GET, NUMBER, and UNNUMBER commands.

Prototype: NUMBER [par]

One of the two following parameter sequences may be given:

[beg] [[,]inc]

The beginning line number for automatic numbering is given by “beg”. If this is omitted, numbering begins with 1. The beginning line number can take any one of the forms:

number
LAST
LAST±number
FIRST
FIRST±number
*L
*L±number
*F
*F±number
MAX
MAX–number
MIN
MIN+number

where LAST or *L is the line number of the last line in the currently active file, FIRST or *F is the line number of the first line of the file, MAX is 99999.999, and MIN is –99999.999. If the file is empty, the value of LAST and FIRST is zero.

The line number increment is given by “inc”. It may be specified in the same manner as the “beg”. If “inc” is omitted, line numbers are incremented by 1. The parameter “inc” must be separated from “beg” by a blank or a comma. If only “inc” is given, it must be preceded by a comma.

CONTINUE

The parameter CONTINUE resumes automatic numbering from the point where automatic numbering was when last terminated by the UNNUMBER command. Changing the currently active file does not disable automatic numbering, but resets the beginning number and increment to the default values. If CONTINUE is specified and automatic numbering has not been previously enabled, the beginning line number and the increment both default to 1.

Program Key: *MTS.NUMBER

Description: The NUMBER command enables the automatic numbering of input lines read from *SOURCE*. In terminal mode, this number is printed at the front of the input line in the form:

```
#$NUMBER
#     1_
#     2_
```

The printing of the number may be suppressed by the SET PFX=OFF command. The line numbers provided by the automatic numbering sequence are taken as the line number for the input line; the first characters of the line are treated as part of the line.

Provided there is a currently active file, any input line not recognized as an MTS command line has a line number assigned to it before it is written to the currently active file. If automatic numbering is disabled, this line number is taken from the first characters of the input line if they form a valid line number; if they are invalid as a line number, the line is treated as an invalid MTS command.

The line number assigned to the input line is used for an indexed write operation to the currently active file. Therefore, the currently active file should be a line file or a device. If the currently active file is a sequential file, the command SET SEQFCHK=OFF must be issued first; however, the line number assigned to the input line is ignored and the line is written sequentially at the end of the file.

In terminal mode, when automatic numbering is enabled and there is an active file, all MTS command lines must begin with a $. Consequently, the UNNUMBER command must always begin with a dollar sign ($).

Automatic numbering may be enabled or disabled and the starting line number and increment may be set from a program by calling the CUINFO subroutine (see *MTS Volume 3: System Subroutine Descriptions*, Reference R1003).

Examples:

```
NUMBER
```

This enables automatic numbering with a beginning line number of 1 and an increment of 1.

```
NUMBER 10,5
```

This enables automatic numbering with a beginning line number of 10 and an increment of 5.

```
NUMBER -100
```

This enables automatic numbering with a beginning line number of –100 and an increment of 1.

PEEK

MTS Command Description

Purpose: To examine and modify the virtual memory of another task.

Prototype: PEEK

Program Key: *MTS.PEEK

Description: The PEEK command allows one task to examine and modify the private virtual memory (VM) of another task. PEEK provides some extensions to the SYSTEMSTATUS "DISPLAY JOB=nnn addr" command, in that it can display things in formats other than hexadecimal without retyping "JOB=nnn" once the task to PEEK at has been specified.

PEEK can also look up addresses and symbols in either the resident-system loader tables, or in the post-IPL load or NAS load-map files.

Permission to examine and/or modify the private VM of another task is controlled by an access file. See the file MTS:PEEK.ACCESS for more information.

PEEK Commands

The following are all of the available PEEK commands (the underlined portions when given are the minimum legal abbreviations of the command or its parameters):

BREAK [break-type] address-range

The BREAK command is used to set breakpoints in the current task. "address-range" is used as the name of the breakpoint, and may be given in the REMOVE command to remove a specific breakpoint. The modifier "break-type" is one of the following:

@EXECUTE (for instruction-fetch breakpoints)
@MODIFY (for storage-modification breakpoints)
@BRANCH (for successful-branch breakpoints)

The default is @EXECUTE.

Note: If a job is suspended that has some system-wide critical resource on a production system, a large number of users could be affected.

Multiple breakpoints are supported. Note that setting several breakpoints can substantially increase the number of interrupts that must be processed before a breakpoint is reached. Also, setting any @BRANCH-type breakpoints causes an interrupt to be processed for each and every successful branch.

Use the CONTINUE command to wait for the breakpoint to occur. Use the REMOVE command to remove a breakpoint. Use the LIST BREAKS command to list the current breakpoints.

The SET options BREAK_GRS and PER_TRACE control the amount of output from various breakpoint related commands. The CLEAN command removes all breakpoints.

CALCULATE address

The CALCULATE command prints the value of the arbitrary expression “address”. Any numeric constants used are assumed to be in hexadecimal format unless specified as F'n'. The result is printed in hexadecimal format.

CLEAN {ALL | BREAKS}

The CLEAN command removes all breakpoints from the current task.

CONTINUE [NOWAIT]

The CONTINUE command continues a task after it has been suspended. The CONTINUE command is a synonym for the WAIT command. Use CONTINUE NOWAIT to continue the task without waiting for a breakpoint.

COPY address-range

The COPY command writes a copy of another task’s VM into your task’s VM, so that SDS can be invoked to display it symbolically. The addresses given in “address-range” specify where to get the information from within the other task’s VM. PEEK will indicate where it put the copy so you can then issue an $SDS command, INCLUDE the neccessary dsects, and issue an SDS USING command. PEEK also indicates how many bytes it was able to copy (which may be less than the number requested).

DISPLAY [modifiers] address-range

The DISPLAY command displays the specified task’s VM. “modifiers” is an SDS-style modifier (see below).

DUMP [modifiers] address-range

The DUMP command gives a hexadecimal/character dump of the specified storage.

The DUMP command also allows the modifier @WIDTH=wid, where “wid” is a number from 1 to 32, inclusive. This indicates how many bytes should be displayed on each line of the dump output. The default width is 16.

FIND symbol-name [IN list-of-tables]

The FIND command searches all known LCS and LCSPR tables for the given symbol. If it finds the symbol, it will print the address of the entry in the LCS or LCSPR table, the name of the table it was found in, and the entry itself.

For LCS entries, it will use the information from the map file if the map file is already read in (by a READ command, or some other action). If no map file has been processed, it will only display the information that is available without checking the map file.

If the keyword IN is given after the symbol name, the specified list (separated by blanks) of LCS and LCSPR tables will be searched.

HELP [PEEK-command]

The HELP command enters the help facility. If a PEEK command is given, that command will be explained.

IST (or INITIALIZE_SYMBOL_TABLE) {SEG2MAP | NASMAP}

The IST command initializes the symbol table with either the Segment 2 or NAS map. This may need to be done after a component is rePISTLEd or reNASLOADed.

JOB task-number

The JOB command is a synonym for the SET JOB=task-number command.

LIST object

The LIST command lists the information that is contained in various data structures of the CLS. “object” may be any of the following:

BREAKS lists all breakpoints.

COMPONENTS lists all components from the current map.

COPIES lists all copy buffers (results of the COPY command).

LCS lists all LCS tables known by the CLS.

LCSPR lists all LCSPR tables known by the CLS.

LCS=sym lists all information about the LCS (or LCSPR) with the given name “sym”.

LCSPR=sym is a synonym for the phrase LCS=“sym”.

UNDOS displays a list of available UNDOs.

MODIFY address [:] newvalue

The MODIFY command modifies the virtual memory of a task. This command is currently restricted to certain signon IDs, even if you are modifying your own private VM. “address” is an arbitrary expression and “newvalue” is the new value to be assigned to the locations starting at the specified address.

“newvalue” may be any of the following:

C'xxxxxxxxx'	a character string.
X'hex-string'	an arbitrary hexadecimal string.
XI'hex-string'	an arbitrary hexadecimal string, but the confirmation will print the old and new values at @T=XI.
F'integer'	a fullword integer.
H'integer'	a halfword integer.
R'rrr.rr Enn'	a real number (defaults to type RL4).
RL4'rr.rr En'	a 4-byte real number (so-called “short real”).
RL8'rr.rr En'	an 8-byte real number (“long real”).
RL16'rrr.rr En'	a 16-byte real number (“extended real”).
V'xxxxxx'	a varying string.
hex-string	an arbitrary hexadecimal string.

A confirmation prompt will be given before any changes are made.

A LIFO list of MODIFY commands issued is maintained so that they can be undone using the UNDO command.

MTS

The MTS command returns control to MTS.

QUERY [modifiers] address
QUERY [modifiers] esd-name

The QUERY command finds out what csect an address is in, or what address a symbol is at. “address” is an arbitrary expression and “esd-name” is the name of a csect or entry. “modifiers” may be one of @ALL, @NOMAP, @MTS, @TMTS, and/or one of @NORES or @RES. Defaults for the modifiers can be specified with the SET command.

QUERY searches for the address or symbol in either or both of the resident-system loader table or the current map file. Modifiers can be used to restrict the search.

The modifiers @ALL, @NOMAP, @MTS, and @TMTS control searching of definitions from the map file. If ALL is in effect, all occurrences of a symbol will be found. If NOMAP is in effect, the map file will not be searched. If MTS is in effect, only the definitions from the MTS part of the map will be recognized. If TMTS is in effect, only definitions from the TMTS part of the map will be recognized. The default is @ALL.

The modifiers @RESIDENT and @NORESIDENT determine whether the resident-system loader table is searched. The default is RESIDENT.

PEEK [modifiers] address-range

PEEK is a synonym for the DISPLAY command.

POKE address [:] newvalue

The POKE command is a synonym for the MODIFY command.

READ [map-file-name]

The READ command forces PEEK to read a loader map. If "map-file-name" is omitted, the default file (see the SET MAPFILE command) or the file specified by a SET MAP command will be used.

Note that PEEK can also process a map file produced by an MTS command:

```
$LOAD program MAP=–MAP
$PEEK
READ –MAP
```

REMOVE {ALL | breakpoint-name}

The REMOVE command removes breakpoints from the current task. If REMOVE ALL is entered, all current breakpoints are removed. If "breakpoint-name" is specified, the breakpoint referenced by that name are removed. REMOVE is a synonym for RESTORE.

RESTORE {ALL | breakpoint-name}

The RESTORE command is a synonym for REMOVE.

```
SET BREAK_GRS={ON | OFF}             Default:OFF
SET CSECT=symbol
SET MAPFILE=FDname
SET OUTPUT=FDname
SET PER_TRACE={ON | OFF}             Default:OFF
SET PREFIX[(DISPLAY | DUMP | MODIFY | ALL)]=pfx-mode
SET {QUERY | QUERY_MODE}={MTS | TMTS | ALL | NOMAP}
SET Rnn=address
SET RESIDENT={ON | OFF}
SET TASK=nnn | JOB=nnn
SET TYPE[(DISPLAY)]=type-option
SET WIDTH[(DUMP)]=n
```

The SET command sets PEEK options.

The keyword BREAK_GRS={ON | OFF} specifies whether to display the general registers when displaying the status of a "broken" task.

The keyword CSECT=symbol specifies which csect to use. Currently PEEK does not actually use this csect for anything.

The keywords TASK=nnn and JOB=nnn specify which task PEEK should use for COPYing, DISPLAYing, etc. Use TASK=ME to select your own task.

The keyword MAPFILE=FDname specifies which loader map to read. The default is SEG2:MAP.

The keyword OUTPUT=FDname specifies the output file/device for the DISPLAY and DUMP commands (and the WRITE command, although that output is not very useful).

The keyword PER_TRACE={ON | OFF} controls the output of additional information about the PER interrupts used for breakpoints.

The keyword PREFIX(DISPLAY)=pfx-mode or PREFIX=pfx-mode sets the default prefix mode that the DISPLAY command uses (see “Prefix Modifiers” for details). Similarly, there is PREFIX(DUMP)=pfx-mode and PREFIX(MODIFY)=pfx-mode for the DUMP and MODIFY commands. PREFIX(ALL)=pfx-mode specifies the default prefix mode for all commands. The default prefix mode (for all commands) is ADDRESS.

The keyword QUERY_MODE={MTS | TMTS | ALL | NOMAP} specifies how to process QUERY esd-name commands when the name is multiply defined. The default is ALL. Specifying MTS means only information from the “real” system is given; specifying TMTS means only information from the TMTS load is given. Specifying NOMAP means the map file will not be processed; only resident-system definitions will be known.

The keyword Rnn=address (where “nn” is a decimal number from 0 to 15) sets the value of the register variable indicated by “nn”. Currently, these values are only used by the @PREFIX=R? and @PREFIX=Rnn options on the DISPLAY and DUMP commands (see “Prefix Modifiers”).

The keyword RESIDENT={ON | OFF} specifies whether to search the resident-system tables as well as the map file. The default is ON.

The keyword TYPE(DISPLAY)=type-option or TYPE=type-option specifies the display format used by the DISPLAY command if none is given (as a modifier to DISPLAY). The default is XL4.

The keyword WIDTH=n or WIDTH(DUMP)=n specifies the default width used by the DUMP command if none is given (as a modifier to DUMP). The default width is 16.

STOP

The STOP command returns to MTS.

TASK nnn

The TASK command is a synonym for the SET TASK=nnn command.

UNDO

The UNDO command undoes the most recent MODIFY command after prompting for confirmation. If the UNDO command succeeds, note that the next UNDO will *not* UNDO the UNDO. That is, issuing a series of UNDO commands will not toggle a location between two values.

WAIT

The WAIT command is a synonym for the CONTINUE NOWAIT command. It continues a task. WAIT is the only PEEK command that will report the occurrence of a breakpoint.

$mts_command

This command executes the MTS command given.

Addresses

Many of the above commands require addresses as one of their operands. An address is an expression composed of hexadecimal numbers, decimal numbers in the form F'integer', and entry or csect symbols, combined using the operators +, −, *, /, and parentheses as necessary.

If a symbol is specified, it is looked up in the resident system and/or post-IPL load map as determined by the SET options QUERY_MODE and RESIDENT. An error message will be given if the symbol is not uniquely defined.

If an "address-range" is requested, the word EVERYWHERE can be used, referring to the addresses X'00000000' to X'7FFFFFFF'.

Other forms for "address-range" are the following:

address [length]

"length" specifies the amount of memory to display. If "length" is of the form LENGTH=byte-count, "byte-count" number of bytes will be used as the length; otherwise, "length" is the number of fullwords to be used as the length.

address1**...**address2

This is the range from "address1" to "address2".

address**...**(num-words)

This is the range from "address" to "address+(num-words)", where "num-words" is a number of fullwords.

address**...**+offset

This is the range from “address” to “address+offset”.

Initfile Processing

The PEEK initfile can contain any valid PEEK command, and all such lines will be processed by PEEK before it processes any commands from the user (even in “one-shot” mode). If an attention occurs during initfile processing or an invalid command is encountered, PEEK will print an error message and abort any further processing.

Modifiers

Some of the above commands take modifiers, which have the same form as a standard MTS modifier, i.e., @something=something-else. The following is a list of the legal options for PEEK.

Type Modifiers

@T=A	Address format. Assumes the location is an address, prints the value in hexadecimal, and then determines (if possible) what csect/entry point is represented by that address.
@T=C	Character format (default length is 4 bytes).
@T=D	Doubleword format (same as @T=XL8).
@T=DUMP	Similar to @T=XC (see below), except that it displays 16 bytes per line (unless told otherwise). This is the format that the DUMP command uses.
@T=E	Real (floating-point) format (defaults to 4-byte reals).
@T=F	Fullword format (defaults to 4-byte integers).
@T=H	Halfword integer format.
@T=I	Instructions mnemonics format.
@T=LCS	Prints locations that are individual entries in an LCS table.
@T=LCSPR	Prints locations that are individual entries in an LCSPR table.
@T=R	Real (floating-point) format (same as @T=E).
@T=STCK	Assumes the locations are the result of the Store-Clock (STCK) instruction, and prints the time (defaults to an 8-byte STCK value).

@T=X	Hexadecimal format (default length 4 bytes).
@T=XC	Prints in both hexadecimal and character formats. @T=XC really means @T=XCL4, which displays 4 bytes per output line.
@T=XCI	Prints instructions in three formats: hexadecimal, characters, and instruction mnemonics.
@T=XI	Prints in both hexadecimal and instruction mnemonic formats.
@T=V	Prints a varying string. The first byte of the location is assumed to be the length and is used to determine how many of the following bytes will be printed.

Most of the above allow an “L=number” to be appended to the end of them to indicate the length of the indicated type (in bytes). For example:

@T=AL3 for a 3-byte address constant.

@T=RL8 for a “long” real number.

@T=FL1 for a 1-byte integers.

@T=STCKL4 if the location only has the first 4 bytes of the result of a STCK instruction.

@T=VL100 for varying strings that have a maximum length of 100 bytes. (This is only used when calculating the end of the variable, in order to find the start of the next one). The default maximum is 255 bytes (not counting the byte used for length.)

Types D, H, I, XI, XCI, LCS, and LCSPR do not allow a length field.

Prefix Modifiers

Another type of modifier is the “prefix-modifier”. Both the DUMP and DISPLAY commands use prefix modifiers to allow the user to select how the memory locations are displayed. The modifiers can be placed immediately following the command or can follow the address range.

@P=ADDRESS	Address format. This indicates the location by displaying the address of it (the default).
@P=BLANKS	This prints blanks at the start if each line displayed (i.e., no indication is given of the location being displayed.)
@P=NONE	Do not print any prefix.

@P=CSECT	If PEEK can determine what csect includes the addresses being displayed, this will print a prefix that indicates the csect and the offset from the start of the csect.
@P=COUNT	The prefix is the count of the lines the current display line is from the starting address in the range being displayed.
@P=Rn	"n" is a specific number from 0 to 15. The prefix will indicate the offset from the value of Rn, if the offset is between X'000' and X'FFF'.
@P=R?	"?" is the actual character question mark. This determines which register has the value closest to (but less than) the location being displayed and displays the offset from that register (as in @P=Rn).

For options that determine some offset, the offset is printed in hexadecimal. If the program cannot find an appropriate csect or register, it will display that line with the plain address prefix.

Examples:

```
#$PEEK
*Set TASK=ME
*Disp 0...(5)
000000/ 00080000
000004/ 80059600
000008/ 00000000
00000C/ 00000000
000010/ 00000000
*Dump 0...+5
000000/ 00080000 8005 "??????"
*Stop
#
#$LOAD PROG MAP=-MAP
#$PEEK
*Read -MAP
*Dis 10...20@t=i@p=r?
R15+10/ DC X'00000000'
R15+14/ DC X'00000000'
R15+18/ DC X'030F'
R15+1A/ DC X'00000000'
R15+1E/ DC X'0000071D'
*List LCS
<EFL> at 361660 ...
CCDEFS at 1C9180 ...
DSPLCS at 2E7730 ...
```

RELEASE

MTS Command Description

Purpose:	To release the currently active file or device.
Prototype:	RELEASE [*AFD*]
Program Key:	*MTS.RELEASE
Description:	If the parameter is omitted or if *AFD* is specified, the currently active file or device is closed and the pseudodevice *AFD* is disassociated from the file or device and becomes undefined. The RELEASE command is normally used in this case after all changes to a file have been made. It protects the user from accidentally entering data into the file through the mistyping of a command. In batch mode, it protects against having unread data cards mistakenly entered into the currently active file after abnormal program termination.

RESTART

MTS Command Description

Purpose: To restart (or initiate) execution of a program following either initial loading, an interrupt, or a subroutine call to ERROR, MTS, or MTSCMD.

Prototype: RESTART [[AT] location] [CLS=xxx]

CLS=xxx

If CLS=xxx is specified, then the corresponding command language subsystem (CLS) is restarted. This is normally only useful for system programmers.

RUN

MTS Command Description

Purpose: To load and execute a program.

Prototype: RUN [program] [I/Ounits] [limits] [mapoptions] [PROT={ON | OFF}] [{EXECPKEY | PKEY}={key | OFF}] [PAR=parameters]

PROT={ON | OFF}

If PROT is OFF, the program is run with system protection turned off. The default is ON unless the user has issued the SET PROT=OFF command. This is option is valid only for privileged userIDs.

The PROT option may also be used with the DEBUG, LOAD, RERUN, RESTART, and START commands.

SET

MTS Command Description

Purpose:	To set MTS global options.
Prototype:	SET option ...

Any number of the following options may be given in a single SET command. The options must be separated by blanks; there must be no blanks within any option.

Almost all of the values set by these options may be interrogated by programs via the GUINFO subroutine and may be changed by programs via the CUINFO subroutine. See the subroutine description for GUINFO and CUINFO in *MTS Volume 3: System Subroutine Descriptions*, Reference R1003, for details.

AFDECHO={ON | OFF} Default: OFF

If the AFDECHO option is ON, all lines written to the active file in MTS command mode are echoed to *SINK* if *SINK* differs from *SOURCE* and to *MSINK* if *MSINK* differs from both *SINK* and *SOURCE*. Each line echoed is preceded by its line number in the active file. If AFDECHO is OFF, echoing is suppressed.

CASE={UC | LC | MX | MIXED} Default: LC

The CASE option specifies global uppercase (UC) or lowercase (LC) conversion. If UC is specified, all data lines read in MTS command mode have lowercase letters converted to uppercase letters; the data lines affected are those lines read in NUMBER mode or those lines prefixed with a line number when there is an active file. If LC is specified, the data lines are not converted. MX and MIXED are equivalent to LC.

CREAFD={ON | OFF] Default: OFF

If the CREAFD option is ON, a successful CREATE command causes the created file to become the currently active file. If CREAFD is OFF, the created file is not made the currently active file.

DEVCHAR=character Default: >

The DEVCHAR option specifies a single character to be used to indicate that the following FDname is a device name, not a file name. This is valid only for privileged userIDs.

FILECHAR=character Default: #

The FILECHAR option specifies a single character to be used to indicate that the following FDname is a file name. This is only required if the first character of the file name is a flag character.

FILEREF={ON | OFF}	Default: ON

The FILEREF option specifies whether the last file reference date is updated when a file is accessed. If FILEREF is OFF, the last reference date is not updated. This option is valid only from the userID MTS.

LNS=character	Default: ,

The LNS option specifies a single character to be used as a line-number separator. The line-number separator is used to terminate the line number for the input line read in MTS command mode and indicate that the remaining characters are a part of the data line itself. This separates a line beginning with numeric information from its line number (e.g., 174,112233445566 has a line number of 174 and a data line of 112233445566).

LSS={ON | OFF}

The LSS option may be used to request a change of state into or out of limited-service state. A limited-service user requesting OFF is converted to full-service state only if the system load at the time of the request permits. A full-service state user requesting ON is converted to limited-service state. Limited-service state normally is only enforced when system malfunction necessitates such action.

NAMEID=chars

The NAMEID option sets the user's 4-character nameID.

PROT={ON | OFF}	Default: ON

The PROT option specifies whether a program is executed with protection turned on or off. When PROT is OFF, the program can issue privileged SVCs and access the system segment. This option is valid only for privileged userIDs.

PWCONFIRM={ON | OFF}	Default: OFF

The PWCONFIRM option controls the action taken by MTS when a terminal user attempts to set a new password. When this option is ON, the terminal user will be prompted to enter the current user password before the password may be changed. When this option is OFF, the current user password is not required. Once the PWCONFIRM option has been set ON during a session, it may not be set OFF during the same session. By setting PWCONFIRM ON from the project or user sigfile, an extra level of security may be obtained.

SCRFCHAR=character	Default: –

The SCRFCHAR option specifies a single character to be used to indicate that a following FDname is a temporary file, not a permanent file.

SDSMSG={ON | OFF}   Default: ON

The SDSMSG option specifies whether SDS or MTS prints certain exceptional condition messages, e.g., attention interrupt, program interupt. If SDSMSG is OFF, MTS prints a less-detailed message.

SHFSEP=character   Default: :

The SHFSEP option specifies a single character to separate the signon ID from the file name when referring to a shared file (e.g., 1AGA:DATAFILE).

UNLOAD={ON | OFF}   Default: ON

The UNLOAD option specifies whether the previous program is unloaded when a new program is loaded. If UNLOAD is OFF, the previous program is not unloaded.

VERSION(command)={TEST | STAFF}

Similar to VERSION(command)={NEW | OLD | CURRENT}. Normally, the test and staff versions are restricted to ITD staff userIDs.

VERSION(command)=FD=filename

Instead of using one of the predefined versions of a CLS command, the version is loaded from “filename”, which must contain an object module that can be executed as a CLS. This is restricted to privileged userIDs.

VERSION(command)=LCS=symbol

Instead of using the normal low-core symbol table, the low-core symbol table “symbol” is is used to resolve the symbols needed by the CLS. This is restricted to privileged userIDs.

SIGNON

MTS Command Description

Purpose:	To identify a user to the system.
Prototype:	SIGNON {userid | *} [SYSTEM={MTS | TMTS}]

SYSTEM={MTS | TMTS}

SYSTEM=TMTS causes HASP to start a TMTS job instead of an MTS job. This valid only for HASP batch and TMTS must be loaded. The default is MTS.

SWAT

MTS Command Description

Purpose: To allow general-purpose system-level debugging.

Prototype: SWAT

Program Key: *SWAT

Description: The SWAT command invokes the SWAT debugger. SWAT provides debugging support for a number of system tasks (like MTS) which require direct or indirect support from the UMMPS environment.

The basic idea behind SWAT is the simulation of an UMMPS interface to a task that is usable in conjunction with SDS. Thus, SWAT provides SVC simulation, CPU queue simulation, interrupt handling, and multi-task dispatching facilities in concert with the debugging capabilities of SDS.

In essence, despite outward appearances, the relationship between the test system being debugged and SWAT is the same as that between SDS and any user program. The test system, when running, is always in user mode. The special services (SVC handling for example) are provided through SWAT's privilege of system mode accorded by the fact that SWAT is a CLS. Routines in the real system called by the test system are entered through a gating mechanism in SWAT (and thus run in system mode as part of the SWAT CLS). SDS is called by SWAT using the standard CLS interface to cover the normal SDS functions (break-pointing, displaying, etc.).

SMTS - Symbolic MTS

SWAT can be used to test almost any part of the MTS system. It can test the Resource Manager, the Disk Manager, the Paging Device Processor, or any other job program. One of the most common uses, however, is to test a new version of the MTS job program (referring in this case not to the entire operating system, but simply to the job program called "MTS".)

Since this is such a common use of SWAT, there is a test system, called SMTS, that is based on the normal MTS system. For example, during the testing of a new system you may wish to test a new MTS command. You can do this by invoking SWAT which will allow you to start and stop tasks on the new system (these tasks are called "test tasks", and the system is called the "test system") which use the new command. The nice aspect of this testing is that you can use all of the SWAT and SDS commands, and the system being tested appears very similar to the real MTS system.

Starting and stopping tasks is also easy, and switching from the test system (which accepts all input as MTS commands) to SWAT (which accepts all input as SWAT and SDS commands) is done simply by hitting attention. Switching from SWAT back into your test system is usually done with an SDS CONTINUE command.

The terms “SMTS test system”, “SMTS system”, and “test system” all refer in this case to the system which simulates the MTS system after your changes have been installed.

SMTS is composed of a set of decks similar to the set of decks comprising the TMTS system. There are no special assemblies other than the possibility of a deck containing SYM records.

One of the requirements for using SMTS is that alternate accounting and statistics files be used (since SMTS is run as a pseudotask within a real task) and that a substitute File Transfer Vector (FTV) be used. This is accomplished through the use of the Multiple Definition Load (MDL) option on the load of the SMTS system, which means that no patches or reassemblies are required when bringing a deck from the real system into SMTS.

The UMMPS/SVC interface provides an alternate job table for the test task. An alternate CNFGINFO table is also provided giving the segment allocation for the test task. The alternate system segment is SSYSEG+2 (which would be segment 8), and the alternate user segment is USERSEG+2 (which would be segment 10, hex A).

The file routines normally used by SMTS are the resident file routines. It is possible to load up an alternate copy of the file routines for the purpose of debugging the file routines, or when the debugging situation dictates. The alternate file routines have the option of working with a real disk or a virtual disk (using the channel and disk routines of the virtual machine). The test system (in this case, SMTS) always works with an alternate IC-table. If the real file system is being used, IC-locks and unlocks are handled through appropriate calls to the real MTS LOCK and UNLOCK subroutines.

With alternate file routines, the LOCK and UNLOCK are handled by IC-routines.

Some current problems with the use of the real file routines include the following:

(1) Since alternate accounting files are used, changes in the status of disk space is not reflected in the real accounting record.

(2) Also, scratch files are created with the pseudotask number (these start at 1 to help avoid conflicts with real tasks). This means that tasks blasted under SWAT will not have their scratch files destroyed and they will not be cleaned up until the next IPL.

(3) Furthermore, program keys are not reflected from pseudotasks to the “real” program-key. This means, for example, that *STATUS will not run from an SMTS task, because the PKEY will be *SWAT and it expects a PKEY of *STATUS.

Note that these problems do not accompany the use of alternate file routines.

Shared VM is handled by providing alternate entries for the things, such as CHGFLG, which MTS likes to change. The set of entry points which resolve to the real system routines is controlled by a set of predefined symbols.

Invoking SWAT

SWAT itself operates as a CLS. SWAT intercepts execution intercepts of the test system, and either processes them or calls SDS as a subroutine to examine the intercept. Note that SWAT does not obtain SYM records through test loads (because this information is only automatically passed to the DEBUG CLS), so symbolic debugging of a deck is accomplished through the use of the SDS INCLUDE command for that deck.

SWAT is invoked by the (privileged staff only) $SWAT command, which prints the following message:

```
SWAT/SDS commands?
```

The available commands are all of the SDS commands plus the following SWAT commands:

ALT {FILE | FILE_SYSTEM | OFF}   Default: OFF

The ALT command causes the test system to use a private set of file routines which use a virtual disk.

ATN virtual-device

The ATN command queues an attention exit for the given virtual device.

BLAST task# **...**

The specified task(s) are terminated and cleaned up.

{CONTROL | CTL} virtual-device string

The CONTROL command passes the given string to the virtual device "xxxx" control entry. CTL is a synonym for CONTROL.

DEBUG {task# | ALL}   Default: ALL

The DEBUG command causes breakpoints and AT-points to occur only in the specified task. The ALL parameter allows breakpoints and AT-points in all tasks. (A side effect of the implementation causes SDS to go into TERSE mode).

With no parameters specified, the DEBUG command prints the address of SWAT's psect and routine addresses. A new command name for this function will be chosen later.

DEFINE device-name device-specification

The DEFINE command predefines device specifications. The specification is the device name followed by its specification, e.g., “T999 3420 -TAPE” would define device “T999” as a virtual 3420 using the scratch file “-TAPE”. A real device is designated by the 4-character real device name.

GET task#

The GET command makes the job with the specified task number active. The GRS and PSW are displayed.

FLIP

The FLIP command is used when running the control task and test SMTS task on the same 3270-type device. The test task believes that it is a grabbed task. Thus, the test DSR allows the %FLIP DSR command, which has the effect of transferring the device back to the real task. If the test task was “WAYTing” (e.g., in GETFROM), the FLIP command has the same effect as the SDS CONTINUE command. If the test task was executing at the time it was %FLIPped, the screen is flipped (by waiting 10 seconds for the test DSR to time-out) to the test task, but only the DSR task is actively executing (the other task is suspended). The test task must be %FLIPped back to SWAT and a CONTINUE command issued before full execution of the test task will be resumed.

{HELP | EXPLAIN} [SWAT-command]

The HELP command lists the available SWAT commands. If SWAT-command is specified, a brief description of the command is given. EXPLAIN is a synonym for HELP.

JOBLIST joblist-name {entry-point-name | address}

The JOBLIST command searches the current loaded symbols for the given “entry-point-name”. If found, a joblist entry is created with the name “joblist-name” and the associated address as the job header address. See also predefined jobs under the START command.

LCS entry-point [address] **...**

The LCS command searches for the given name in the test system. If found, the name is added to the test systems LCSYMBOL.

LOAD FDname joblist-name [NOMERGE] [LCDEF] [PROT]

The LOAD command loads a test task. A joblist entry is created with the entry point from the load used as the job header address, and name as given by “joblist-name”.

Currently, “joblist-name” is required in order that systems can be UNLOADed. This may be changed in the future.

NOMERGE loads the test task with a non-merged table.

LCSDEF allows this load to redefine LCSYMBOL. If the jobname is either “SMTS” or “MTS” definition of LCSYMBOL is automatically allowed.

MTS

The MTS command returns control to MTS.

PGNT {ON | OFF} Default: OFF

The PGNT command controls action on program interrupts. If PGNT is OFF, execution suspends on the interrupt. Otherwise, the task is given an exit code “PGNT” provided the exit is enabled.

PREDEF {ADD | DELETE} [GATE={YES | NO}] [ENTRY=name] sym **...**

The PREDEF command adds or deletes symbols from the initial ESD list used on loads. The ENTRY symbol specifies an alternate symbol to be used to resolve the address. GATE=YES causes subroutine calls to the symbol to be gated. GATE=NO is for table locations. Defaults are ADD, GATE=YES, and ENTRY=sym.

PISTLE FDname {IPL | NOIPL} [LCS=lcs-table] [lcs-sym] **...**

The PISTLE command causes a “Post-IPL-Load” from the given FDname. If the file contains a COM card, additions are made to the LCS table accordingly. Similarly, any symbols following the FDname on the PISTLE command will be defined in the LCS table. The default table is LCS=LCSYMBOL. By default, the module is loaded as a non-IPL componant, with symbols resolved through LCSYMBOL. If IPL is specified, symbols are merged with the resident-system load table.

RELEASE

The RELEASE command cancels the effect of the previous SUSPEND command.

START joblist-name [job-parameters]

The START command searches the current joblist for the “joblist-name”. If found, a new task is created and dispatched with a parsed version of the given job parameters. SWAT does not yet support preallocation of devices from the job header. Note that “/” is a synonym for “START ”, e.g., “/SMTS” starts an SMTS task.

SWAT has a small set of predefined jobs. These are a set of known joblist names which associate the name of the file from which to load the system (if the system is not already loaded). Currently, the set consists of:

SMTS	loads an SMTS system and starts an SMTS task
T	loads and invokes the TASKS job
INIT	loads and invokes the catalog initialization job.

SVCHALT {ON | OFF} Default: ON

The SVCHALT command controls the disposition of SVC ERRORS. SVCHALT ON causes suspension (with the exit set up) when an SVC error occurs. SVCHALT OFF allows execution to continue at the exit routine address or terminates the task if there is no exit set.

SUSPEND {ON | OFF} Default: OFF

The SUSPEND command affects the dispatching of multiple tasks. If ON, a task which causes a SWAT command read will not be redispatched until a CONTINUE or other SDS execution resumption command is issued for that task, or until a RELEASE command is issued removing all suspensions. The GET command may also be used to reactivate a suspended task before the CONTINUE command is issued.

SVCTRACE {ON | OFF} Default: OFF

The SVCTRACE command turns SVC tracing on or off. Tracing is done on MSINK and includes a trace of each intercepted SVC and its parameters as well as a trace of asynchronous interrupt disposition.

TERMINATE task# [termination-code]

The TERMINATE command terminates the given task with the code termination-code. The default termination code is “KILL”.

UNLOAD joblist-name

The UNLOAD command unloads the test system associated with this “joblist-name”. Currently, the “joblist-name” must be the one defined on the LOAD command, not a “joblist-name” generated with the JOBLIST command. A future version will possibly allow other methods (such as entry point) as the unload specification.

Examples:

```
#$SWAT
<SWAT>: SWAT/SDS commands?
<SWAT>: /smts
<SWAT>: ** Starting SMTS task: Job 1
<SWAT>: ** Running one device: "DS04"
< 1>: Protection exception at 000000
< 1>: blast 1
< 1>: ** Job 1 Terminated.
<SWAT>: /smts oper
<SWAT>: ** Starting SMTS task: Job 2
< 2>: ** ICLOCKR IC error: LOCK RC=12 File: *HWTERM
< 2>: 17:28:44 THE PAGE PRINTER AT CNTR IS DOWN.
< 2>: 17:28:44 < 2>: sig mtss
< 2>: 17:28:54 < 2>: dis ccid
< 2>: 17:28:59 User ID: MTSS
< 2>: 17:28:59 < 2>: {hit ATTN}
< 2>: ** ATTN!
< 2>: include cmds>1oa
< 2>: dis runret
< 2>: runret I: SVC 6
< 2>: c
< 2>: 17:29:56 < 2>: run *time
< 2>: 17:30:03 Execution begins
< 2>: 17:30:03 CLOCK 17:30:03 DATE 09-04-87
< 2>: 17:30:04 Execution terminated
< 2>: 17:30:49 < 2>: sig $
< 2>: 17:32:14 MTSS 17:28:52 to 17:32:14, Thu Sep 04/87
< 2>: 17:32:14 $.32
< 2>: ** Task 2 terminated.
< 2>: ** No more jobs to dispatch.
<SWAT>: stop
<SWAT>: SWAT terminated.
#
```

SYSTEMSTATUS

MTS Command Description

Purpose:	To provide information about various aspects of system operation.
Prototype:	SYSTEMSTATUS [systemstatus-command]

SCREEN [{ALL | station}]

The SCREEN command displays continuously the status of each batch, *PRINT*, and *PUNCH* job for the specified station. If ALL or no parameter is specified, the status of all jobs in the system is displayed. The name SCREEN derives from the fact that a television-like display of relevant job status information is generated.

The format of the display is as follows. First, a summary line is printed consisting of the date and time, the number of terminal users, the number of batch jobs executing and awaiting execution, and the approximate batch throughput rate. Second, the status of each job is given by a one-character code following the job receipt number. The codes are:

A	awaiting execution
E	executing
M	awaiting printing (line printer)
P	printing (line printer)
S	awaiting UMnet/Michnet transmission
T	awaiting punching
U	punching
W	being sent to UMnet/Michnet
X	awaiting printing (page printer)
Y	printing (page printer)
*	done (line printer/punch)
#	done (page printer)

Summary lines are printed giving the number of jobs printing, waiting to print, punching, and waiting to punch.

In terminal mode, this command continues to print information until interrupted by an attention interrupt; in batch mode, one copy is printed.

The output from this command is continuously displayed on the job status display screens at the Campus Computing Site batch stations.

Description: Several privileged SYSTEMSTATUS commands are available. These are described in the MTS Operators Manual.

UNLOAD

MTS Command Description

Purpose: To unload the currently loaded program in virtual memory.

Prototype: UNLOAD [CLS=xxxx]

If CLS=xxxx is specified, then the corresponding command language subsystem (CLS) is unloaded. This is normally only useful for system programmers.

UNNUMBER

MTS Command Description

Purpose:	To disable automatic numbering of input lines from *SOURCE*.
Note:	The EDIT and the COPY commands provide much simpler and safer methods of inserting, deleting, or changing data lines in a file. These methods are preferred over the use of the GET, NUMBER, and UNNUMBER commands.
Prototype:	UNNUMBER
Program Key:	*MTS.UNNUMBE

Description:

Automatic numbering of input lines being read from *SOURCE* is disabled. Any input line, not recognized as a command line or a data line (a line starting with a valid line number), is treated as an invalid command.

In conversational mode, when automatic numbering is enabled and there is an active file, all MTS commands must begin with a “$”. Consequently, the UNNUMBER command must always be preceded by a “$”.

Automatic numbering may be enabled or disabled from a program by calling the CUINFO subroutine (see *MTS Volume 3: System Subroutine Descriptions*, Reference R1003).

Example:

```
#NUMBER
#       1_THIS
#       2_IS DATA
#       3_$UNNUMBER
```

Automatic numbering was started with a beginning line number of 1 and an increment of 1. Three lines later, automatic numbering was suspended with the UNNUMBER command. Automatic numbering may be resumed with a beginning line number of 3 and an increment of 1 by issuing the NUMBER CONTINUE command.

#CLS

MTS Command Description

Purpose:	To directly load an object module and execute it as a CLS.
Prototype:	#CLS [filename] [options]
Program Key:	*#CLS

Description: The #CLS command directly loads the object module “filename” and executes it as a CLS. The first time this command is used, the following form should be used:

```
#CLS filename options
```

To subsequently reenter the CLS, the following form should be used (assuming that the CLS has not been unloaded):

```
#CLS options
```

The CLS can be unloaded with the command

```
UNLOAD CLS=#CLS
```

The command SET VERSION(command)=FD=filename is an alternative to the #CLS command.

The #CLS command is valid only for privileged userIDs.

THE ACTIVE FILE OR DEVICE

This section describes the concept of the active file or device. While this feature is not generally recommended, its description is presented here for the sake of completeness. The EDIT and COPY commands provide much simpler and safer methods of inserting, deleting, or changing data lines in a file. These methods are preferred over the use of the GET, NUMBER, and UNNUMBER commands.

The following description indicates how a user enters command and data lines.

All lines read in MTS command mode are either

(1) interpreted as MTS commands, or

(2) interpreted as data lines for the active file.

If a line is interpreted as a command line but the name entered is neither a legal command name nor a valid abbreviation of a command name, an error comment is produced. Similarly, if a line is interpreted as a data line but there is no active file, an error comment is produced. The currently active file may be established by entering the command

```
GET FDname
```

if the file already exists or if FDname specifies a temporary file. The CREATE command may be used to establish the active file (if the SET command has been used to set the CREAFD option ON). The RELEASE command may be used to deactivate the active file. Obviously, if the active file is DESTROYed, there will be no active file.

The following discussion describes the method of entering data lines into the active file in MTS command mode. An alternative and more general method of data entry is available in edit mode which is described in *MTS Volume 18: The MTS File Editor*, Reference R1018.

As stated above, data entry in MTS command mode is strongly oriented toward line files. The following discussion assumes that a line file has been established as the currently active file. Results obtained when a sequential file or some other device is assigned to *AFD* are discussed at the end of this section.

A single dollar sign ($) occurring as the first character of an input line from *SOURCE*, is used to indicate that the line is an MTS command. An input line that does not begin with a single dollar sign may be interpreted as a data line. In MTS command mode every data line must have a line number associated with it. The line number may be assigned in one of two ways; either MTS supplies the line number (automatic line-numbering enabled via the NUMBER command) or the user supplies the line number (automatic line-numbering disabled).

When automatic line-numbering is enabled, MTS assigns a line number to each input line. When automatic line-numbering is enabled, an input line is interpreted as a data line unless the first character is a dollar sign *and* the second character is not a dollar sign. For example:

```
$COMMENT
```

is interpreted as a COMMENT command and the line is not entered into the active file. On the other hand, the line:

```
COMMENT
```

is treated as a data line. It is given the line number generated by MTS and placed in the active file. To enter the line "$COMMENT" into the active file, the line:

```
$$COMMENT
```

must be entered. The double dollar sign indicates that the line is a data line. The first dollar sign is removed from the line before it is entered into the active file.

When automatic line-numbering is disabled, the user must supply a line number with each data line. This is done by entering a line that consists of the line number followed by the data. The line number is entered starting at the first character position of the line. A legal MTS line number is a decimal or integer number between −2147483.648 and +2147483.647 inclusive. MTS scans the beginning characters of the input line for a line number and, if one is found, strips it from the line. The remaining characters are used as the data to be placed in the active file using the specified line number. For example, the line:

```
15HI THERE
```

causes the characters "HI THERE" to be entered into the active file as line 15. The *line-number separator* character (by default the comma ",") may be used to separate the line number from the rest of the line when the line number and the data might be confused. For example:

```
21.6,10312 CONTINUE
```

places "10312 CONTINUE" at line 21.6 in the active file. The line-number separator may be omitted if there is no ambiguity in determining where the line number ends and the data characters begin. If the first character of the data portion of the input line is the same as the line-number separator, the user should enter *two* line-number separator characters; the first one is treated as the line-number separator (it is removed) and the second one becomes the first data character. For example:

```
25,,ABCDE
```

causes the line ",ABCDE" to be entered as line 25 in the active file. Alternatively, the SET command may be used to change the character assignment for the line-number separator. Thus, the sequence

```
$SET LNS=|
25|,ABCDE
```

produces the same result as the example given above. Similarly, the sequence:

```
$SET LNS=|
25,ABCDE
```

produces the same result since there is no ambiguity in determining which portion of the line is the line number and which portion is the data. The CUINFO subroutine may also be used to change the line-number separator character (see *MTS Volume 3: System Subroutine Descriptions*, Reference R1003).

The line "number" may also be given as FIRST, LAST, *F, *L, MIN or MAX. FIRST and *F refer symbolically to the first line of *AFD*; LAST and *L refer to the last line of *AFD*; MIN and MAX are −2147483.648 and +2147483.647, respectively. Each of these symbolic names for particular lines of *AFD* may be modified by a signed decimal increment (or decrement). For example,

```
LAST+2.7
```

is a valid expression for a line number. All of these characters will be peeled from the beginning of the input line and the appropriate line number assigned.

Users are cautioned that a line is treated as a command if the first character after the line number (and line-number separator, if applicable) is a dollar sign and the second character is not a dollar sign. For example:

```
10,$COMMENT
```

is treated as a COMMENT command, but

```
10,$$COMMENT
```

causes the data line "$COMMENT" to be placed in line 10 of the active file.

If automatic line-numbering is disabled and the input line does not begin with a line number, the line is interpreted as a command line. Note, however, that if this condition arises in a *batch* job, the command is illegal unless it begins with a dollar sign. For terminal jobs the dollar sign preceding a command name is optional unless there is an active file or device *and* automatic line-numbering is enabled. If the dollar sign is omitted under these conditions, the input line is considered to be a data line. If there is an active file and automatic line-numbering is enabled, the dollar sign is required to distinguish a command line from a data line. The dollar sign is *always* required for MTS commands in batch mode to avoid interpreting data as commands and causing damage to files.

Both command lines and data lines may be continued by using the MTS line-continuation character. This character is set by the CONTCHAR option of the MTS SET command; the default is the minus sign.

When an input line is determined to be a command line, lowercase letters in the line, if any, are converted to uppercase before the command is processed. Data lines are converted to uppercase if the UC option has been set ON by the SET UC=ON command or by an appropriate call to the CUINFO subroutine (see *MTS Volume 3: System Subroutine Descriptions*, Reference R1003).

When a data line is written to the active file or device, an *indexed* write operation is specified (using the line number provided by the automatic line-numbering facility or the number given on the input line). For a line file, the data line replaces the line with the same line number if it exists. For a sequential file, an error comment is produced unless sequential file checking is disabled. When sequential file checking is disabled, the line becomes the last record in the file. If *AFD* is a device (e.g., a magnetic tape or line printer), the line is written at the current position of the device; the indexed modifier for the write operation is ignored. For all but line files, the COPY command provides a preferred method of data entry.