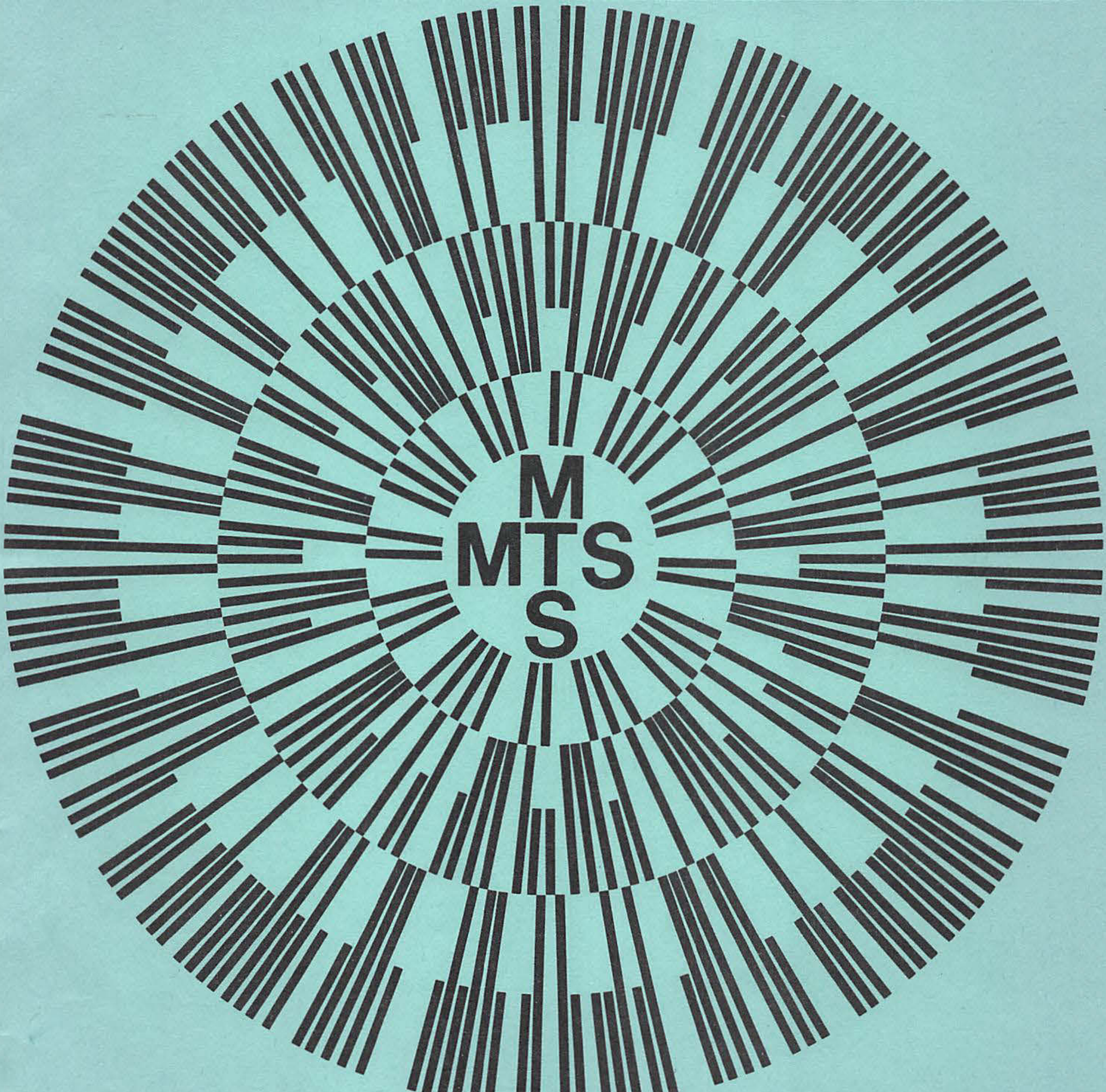
DAVIS

THE UNIVERSITY OF ALBERTA
COMPUTING CENTER
PUBLICATION



TERMINAL USERS' GUIDE

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

TERMINAL USERS' GUIDE
MAY 1970

TERMINAL USERS' GUIDE

TABLE OF CONTENTS

1. INITIATING A TERMINAL SESSION

A. For use as an MTS terminal the LCL/COM switch on the upper left side of the terminal must be set to "COM" and the main power switch must be "ON" before dialing the computer telephone number. The position of the "golf ball" carrier is not important; it will be reset automatically.

B. The transmission of information between the 2741 and the CPU is accomplished via the 2703 Transmission Control and 2870 Multiplexor Channel.

The 2741 is connected to the 2703 Transmission Control unit either directly (hard-wired) or via common carrier connections requiring the use of a standard data phone and a 103A data set. (dial-up)

1. Hardwired 2741 terminals establish a connection when the power is switched on.
2. To make the telephone connection for dial-up 2741 terminals, depress the "TALK" button on the data phone panel, pick up the hand set, and dial

```
432 - 4811      Note:  for terminals on campus
432 - 4821             only the last 3 digits
432 - 4831             are required.
```

These are trunk hunting lines, so you are actually choosing any available line among many. If MTS is running and there is a free line, you will hear a high pitched tome. When the high pitched tone is heard depress the "DATA" button and hang up the hand set. If you hear a ring with no answer, MTS is not currently accepting phone calls! If MTS is up and all the lines are busy, you will hear a busy signal. In either of these cases, your only re-course is to hang up and try again later.

C. A completed call to the system will cause the following message to be typed at your terminal

```
MTS (LAnn-nnn)
```

This process includes some clicking noises as your terminal identifies itself to the computer and all necessary initialization is done. After the above heading is typed on the terminal there may follow another line which represents a current message from the Computing Center to all users; you should read this line if it is offered to you. Finally, a # is typed on the left margin of the terminal paper; MTS is now waiting for an input from you!

Whenever MTS wishes a command from a user, the # is typed on the left margin of the paper; the user should then respond with a command.. The first command must be a SIGNON command which takes the form:

```
      $SIG  CCID
```

Where:

```
      CCID  is your Computing Center id number
```

The system will then type

```
      #ENTER USER PASSWORD.
      ?███████
```

and leave the type ball positioned at the first character of the mask. The user should then enter the password on top of the mask, which will obscure the password.

If your signon id (CCID) is currently active and is allowed terminal time, and if the password given is correct for the pertiment CCID, MTS will type out further information for the user on the terminal. This information takes the form:

```
      #**LAST SIGNON WAS: time date
      #  USER "CCID" SIGNED ON AT time ON date
```

This information gives the date and time for the last time the pertinent CCID was signed on the machine and the same information for this signon time and date. This information may be useful in detecting illegal use of the CCID. After the above information has been typed, MTS again types a # on the terminal indicating that the user is now signed on and that MTS is awaiting his next command. The user may now start his "conversation" with MTS, requesting the services he needs and providing the information MTS needs.

2. CONVERSATION OPERATION

A. Terminal modes: During operation the 2741 may be in one of three modes: receive, control-receive, or transmit. The keyboard (with the exception of the "attention" key) is locked except when in transmit mode. Normally the 2741 is placed in transmit mode only when MTS expects a line to be entered.

B. Prefixing: So that the user can know "who is speaking" and so he knows when input is expected, the first character of all lines on consoles is a special prefix character. On output lines this is typed ahead of the message. When input is requested, either the prefix character (automatic numbering off) or the prefix character followed by the line number (automatic numbering on) is typed at the front of the line. The prefix characters are:

#	issued by MTS monitor
blank	issued by user's program at run time:
.	issued during loading
>	issued during LIST or COPY
?	issued to prompt user for reply
=	issued by *PIL (Pittsburgh Interpretive Language)
+	issued by *DEBUG (Symbolic Debugging System)
:	issued by *EDIT (the Editor)

C. Entering MTS Lines: Alphabetic charaters in command lines are always converted to upper case before the command line is analyzed; thus $SIGNON and $signon produce the same effect. Alphabetic characters in data lines are automatically converted to upper-case. If this automatic conversion is not desired, entry of lower-case characters can be accomplished using the %K device command, the $SET command, or the @LC modifier. sp.

Four characters are assigned special control functions for the 2741 communications with MTS. These are:

1. Underscore - this causes deletion of all previous characters of an input line. Any characters entered after an underscore, but before line termination, will be text for the next line. An underscore followed by a carriage return causes the line to be deleted and the words LINE DELETED to be printed on the next terminal line to indicate this.

2. Back space - this causes the preceding character of an input line to be deleted. Consecutive backspaces may be used to delete several previous characters or even an entire line; however, if an entire line is

wiped out with backspaces and then the carrier return key is depressed a zero length line is transmitted to the MTS routines. (Note that this differs from a line deleted by the underscore, which is never transmitted to MTS routines.

3. Cent sign - this is used to indicate logical end of file; the contents of the input containing a ¢ are not transmitted to MTS, only the end-of-file signal is transmitted.

4. Exclamation point - this is used as the "literal next character" character. Should it be desirable to actually enter a backspace, cent sign, or exclamation point into a command or data line, these characters can be preceded by one "exclamation point". In this context the pair of characters is taken as a single character with the normal graphic value of the second rather than as a sequence of control characters.

The order for analyzing input lines is as follows:

a. Literal next characters are applied (note that literal next characters have no meaning unless they precede one of the four special characters and are ignored if out of context).

b. If any underscore characters remain they are applied to delete all characters preceding the underscore.

c. If any backspaces remain they are applied to delete the appropriate previous characters.

d. If an end-of-file character remains, a logical end-of-file is returned to MTS; otherwise the edited line is returned.

e. Any line which constitutes a valid device command is intercepted and acted upon rather than being transmitted as an ordinary input line.

Any length of time may be used to enter a single input line via the 2741; however if there is no activity for a span of approximately 15 minutes the user and terminal will be automatically signed off. Actually a "timeout" occurs (in the 2703) if no character is entered within 28 seconds of the previous character. In this event all characters transmitted are saved and the 2703 is again prepared to receive text from the 2741 so that another segment of the input line can be entered. An input line is thus accumulated over a relatively long time interval. It may occur that a user enters a character while the 2703 is being reset for the next line segment (very unlikely but it can happen); in this case the message LINE DELETED: LOST DATA will appear and the entire line will have to be reentered. Input lines may contain up to 128 characters.

D. Continuing lines: If the last character in the source stream (prefix char #) line is a minus sign ("-"), then the next input line is assumed to be a continuation. Continuation begins with the first character of the next line, which may be assumed to replace the "-" continuation character in the previous line. As many continuation lines as desired may be used, with the restriction that their total length may not exceed 255 characters. This is effective only for lines read by the MTS monitor, i.e., read when the prefix character is #.

E. Indication of Execution: The type ball will "twitch" at approximately 28 second intervals during execution of various commands to indicate that execution is in progress.

F. Attention Interrupts: An attention interrupt is a signal to MTS to interrupt whatever it is doing for you and to return for another command line. One may interrupt the execution of a program, the listing of a file, etc. by depressing the attention key. This may be done during either terminal input or output operations. What happens next depends upon many things, but eventually you should get the comment

```
ATTENTION INTERRUPT AT xxxxxxxx
```

or the comment

```
ATTN!
```

The first comment occurs only if some program was in execution at the time you hit the ATTN button; in this case, xxxxxxx is the hexadecimal address at which execution was interrupted by the break. The second comment is given if no program was in execution when you hit the ATTN button. After the above attention message has been printed, you will get the MTS # prefix to indicate that once again MTS is ready for an input command from you. (Note: some system components field the terminal interrupts themselves, rather than allowing MTS to service the interrupt; for such a component, the prefix printed after the attention message will be the prefix character used by the component itself.) At this time you may enter a new command including $RESTART which causes execution to resume where it was interrupted.

If there is no response to your interrupt (i.e., if nothing happens after you have pushed the ATTN button), then you have lost communication with the system due to dropping of your line or to hardware or software malfunctions. You should try to reestablish communication as outlined earlier.

<u>Note</u> An attention interrupt will restore SINK to MSINK and SOURCE to MSOURCE.

3. DEVICE COMMANDS

This system provides Terminal users with a number of device commands to set margins, tab-stops and perform a number of other functions. A device command consists of a % sign in column 1 followed by a command identifier and an operand.

A device command must appear exactly as described in the command descriptions given below. Line termination must occur immediately after the last character of the device command. The addition of a trailing blank (or blanks) will cause the line to be transmitted in the normal fashion. All input lines are monitored (after the usual editing for literal next, delete previous, delete line and end of file characters) so that any line which constitutes a valid device command is intercepted and acted upon rather than being transmitted as an ordinary input line.

Device commands fall into five groups as follows:

1. Commands that allow the user to describe the carriage format for his terminal; these include left and right margin settings and tab-stops.
2. Commands that allow the user to specify upper case conversion and/or hexadecimal input as input modes.
3. Commands that allow the user to redefine the characters having special significance on input ("literal next" character, etc.)
4. The "length" command that allows the user to establish the truncation length for output lines.
5. The "reset" command (reinitializes everything that can be changed by a device command).

The character that signifies a device command is % (percent). All device commands must include this character as the <u>first</u> character (except as indicated in the %DCC command). Alphabetic characters of a device command may be entered in either upper or lower case (or mixed) if the terminal device permits.

If an input line cannot be recognized as a device command the line will cause the appearance of the message "INVALID COMMAND." Device commands may be entered at any time the terminal is in input mode, even if MTS itself is not in command mode.

If an input line is recognized as a device command but the parameters for the command violate the specified constraints for that command, the comment "LINE DELETED: INVALID DEVICE COMMAND" will appear.

Name: LEN

Purpose: To define the truncation length for output lines.

Prototype: $\%LEN=\left\{\begin{matrix}ddd\\OFF\end{matrix}\right\}$

Where ddd is a decimal integer, between 1 and 225 inclusive, defining the truncation length for output lines.

Effect: Output lines will be truncated at the length specified by the command. The truncation length applies to the output line but not to the line prefixes. If the length specified is greater than the difference between the logical carriage length and the prefix length, the output line will be continued on as many successive printed lines as required to print the specified number of characters (except for trailing blanks). The use of

```
%LEN=OFF
```

will cause a return to the default output mode. The default truncation specification is equal to the logical carriage length (as determined by the left and right margins) minus the prefix length for the line.

Comments: Each continuation line of output begins with one, two, or three asterisks (depending on the length of the prefix).

Examples:

```
%LEN=123
%LEN=OFF
```

Name: RMAR (for right margin)
LMAR (for left margin)

Purpose: To indicate the position of the left and right margin stops.

Prototype:

```
%RMAR=dd
%LMAR=dd
```

where dd is a decimal integer representing the column number of the margin. The dd is subject ot the constraint that $0 \le dd \le$ the physical carriage length, also, the right dd > the left dd.

Effect: The maximum number of printed characters (logical carriage length) of each output line is set equal to the difference between the right and left margin stops. There is no effect on the length of the input lines.

Examples:

```
%RMAR = 30
%LMAR = 2
```

Name: TABI (for input)
TABØ (for output)

Purpose: To set or release the logical tab stops.
To define the logical tab character.
To establish the position of the logical tab stop.

Prototype:

```
      I    ØN       ;x
%TAB{   }={   }[[{     }]][,dd...]
      Ø    ØFF      ;dd
```

where x is the logical tab character and dd is a decimal integer representing the column position of the tab stop. The default for x is "TAB" and all tab stops are normally cleared.

Effect: The selection of I or Ø determines whether the remainder of the device command pertains to input or output. The selection of ON indicates that lines transmitted are to be expanded according to the logical tab character placed in position x and the tab stop values currently in effect. Expansion of the lines is accomplished as described in the writeup on the public file *TABEDIT.

Examples: %TABI=ON; ,10,16,36 explicitly a blank according to the following text.

This command will cause input lines to be expanded using the character 'blank' as the logical tab character with tab stops set at 10, 16, and 36. Any logical tab characters encountered after position 36 will result in the insertion of a single blank.

%TABI=OFF

This causes the releasing of input tab expansion

%TABO=ON;,,10,20,30,40,50

This command will cause the expansion of output lines using the comma as the logical tab character.

Comments: The default case for tab parameters are as follows:

1. Tab expansion is OFF for both input and output.
2. The default logical tab character is TAB
3. All tab stops are cleared

A maximum of nine tabstops may be set. Tabs may be enabled or disabled without affecting the positions of the logical tab stops. Entering even a single tab stop has the effect of clearing all of the old tab stops.

Name: HEX

Purpose: To enable or disable the use of hexadecimal input editing and to define the hexadecimal input delimiter.

Prototype: %HEX= $\left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\}$[;x]

where x is the hexadecimal input delimiter.

Effect: %HEX=ON enables hexadecimal input editing. The default parameters for the HEX command are as follows:

1. The Hexadecimal editing is OFF
2. The Hexadecimal delimiter is ' (prime).

x, if used, specifies the new delimiter. The delimiter may be redefined either when enabling or disabling hexadecimal editing.

Hexadecimal editing, if enabled, occurs after the usual editing for literal next, delete previous, etc., and after monitoring for device commands. Upon encountering the delimiter in an input string the following characters are interpreted as hexadecimal input, two characters per byte, until the delimiter is again encountered. Commas that appear at byte boundaries in hexadecimal input are ignored. Hexadecimal mode may be entered or left any number of times in an input line. The line must be terminated in normal character mode. To enter the delimiter as a test character, two consecutive delimiter characters must be entered.

Examples:

```
%HEX=ON
```

enables hexadecimal editing with the currently defined delimiter.

```
%HEX=OFF;*
```

disables hexadecimal editing and establishes * as the hex delimiter for the next time hex editing is enabled.

Name: K

Purpose: To specify the alphabetic conversion mode for input lines from the keyboard of the terminal.

Prototype: $\%K\left[\begin{matrix}@UC\\@LC\end{matrix}\right]$

Effect: %K@UC causes all alphabetic input from the keyboard to be forced to upper case (this is the default specification for 2741 terminals).

%K@LC causes alphabetic input to be entered in the same case as it is keyed.

Examples:

```
%K@UC
%K@LC
```

Name:	DCC
Purpose:	To redefine the device command characters.
Prototype:	%DCC=x
Effect:	The character in position x replaces the previously established command character.
Comment:	The default device command character is %.
Example:	%DCC=+

Name:	DLC
Purpose:	To redefine the "delete line" character.
Prototype:	%DLC=x
Effect:	The character placed in position x replaces the previously established "delete line" character.
Comment:	The default is _ (underscore)
Examples:	%DLC=?

Name:	DPC
Purpose:	To redefine the "delete previous" character.
Prototype:	%DPC=x
Effect:	The character placed in position x replaces the previously established "delete previous" character.
Comment:	The default is backspace.
Examples:	%DPC=Q

Name:	EFC
Purpose:	To redefine the "end-of-file" character.
Prototype:	%EFC=x
Effect:	The character placed in postition x replaces the previously established "end-of-file character.
Comments:	The default is ¢ (cent sign)
Example:	%EFC = " (a quote sign)

Name:	LNC
Purpose:	to redefine the "literal next" character.
Protytype:	%LNC=x
Effect:	The character in position x replaces the previously established "literal next" character.
Comment:	The default is ! (exclamation)
Example:	%LNC = #

Name:	RESET
Purpose:	To reset everything that can be changed by a device command back to its initial condition.
Prototype:	%RESET
Effect:	Margin stops are set at extremes. Tabs are cleared and disabled for input and output. Line length truncation is disabled. Alphabetic input is forced to upper case Hexadecimal input editing is disabled and the delimIter is reset to the default value. The device command character, literal next character, delete previous character, delete line character and end-of-file characters are all reset to their default value for the terminal device.
Example:	%RESET

4. USE OF PSEUDO-DEVICE NAMES

When a user is signed on at a terminal, the system defines the pseudo-devices *MSOURCE* and *MSINK* to be the terminal, and initially defines *SOURCE* and *SINK* to be the terminal. This means that if *SOURCE* is assigned to a logical I/O unit (by the $RUN command) that a read operation to the logical I/O unit will cause a read operation on the terminal, and conversely a write operation to a logical I/O unit assigned to *SINK* will cause a write operation on the terminal.

The user can redefine *SOURCE* and *SINK* by using the $SOURCE or $SINK commands, however it should be noted that an attention interrupt on the terminal will cause *SOURCE* and *SINK* to be redefined as the terminal.

The pseudo-device name *PUNCH* is not defined for the terminal user.

5. TERMINATING A SESSION

With the 2741 in transmit mode enter the command $SIGNOFF. After this command line is scanned MTS will properly close all of your files (this may take a few seconds) and then type out a number of statistics gathered about the use of the computer during the conversation. These statistics include:

the time of day of the signoff
the elapsed time during which the terminal was signed into MTS
the actual CPU time used by the conversation
the storage used by the conversation
the number of drum reads required during the conversation
the approximate cost (in dollars and cents) of the conversation
the file storage used and the approximate cost of this storage

If one does not wish to have all of this information printed at the terminal as part of the signoff procedure, he may modify this action by modifying the $SIG command to read:

```
$SIG  SHORT
```

The result is that the signoff statistics are greatly abbreviated.

The line will be automatically disconnected. The user should turn power OFF on the terminal before leaving.

6. SAMPLE TERMINAL SESSION

```
MTS (LA85-0016)
#WELCOME TO THE WONDERFUL WORLD OF MTS.
#$sig sid1
#ENTER USER PASSWORD.
[redacted]
#**LAST SIGNON WAS: 08:42.03   04-26-70
# USER "SID1" SIGNED ON AT  08:42.54 ON 04-26-70
#$run *users
#EXECUTION BEGINS
 THERE ARE   2 TERMINAL USERS,    0 BATCH TASKS,    1 AVAILABLE LINES,
 AND   7 NON-MTS JOBS USING    25 VIRTUAL PAGES AND  25 REAL PAGES.
#EXECUTION TERMINATED
#$run *status
#EXECUTION BEGINS

 STATUS OF SID1 AT LAST SIGNOFF         USED      MAXIMUM    REMAINING

 CUMULATIVE CHARGE          ($)          6.20     9999.00      9992.80
 CURRENT DISK SPACE         (PAGES)        16         999          983
 CUMULATIVE TERMINAL TIME   (HR)         1.15

#EXECUTION TERMINATED
#$create forprog
# FILE "FORPROG" HAS BEEN CREATED.
#$get -t
#READY.
#$number
#     1_ namelist /nl/a,roota
#     2_10 read (5,nl)
#     3_ roota = sqrt(a)
#     4_ write (6,nn)
#     5_ go to 11
#     6_ end
#     7_$unnumber
#$release
#$list -t
>      1       NAMELIST /NL/A,ROOTA
>      2      10 READ (5,NL)
>      3       ROOTA = SQRT(A)
>      4       WRITE (6,NN)
>      5       GO TO 11
>      6       END
#END OF FILE
#$r *fortedit scards=-t spunch=forprog
#EXECUTION BEGINS
#EXECUTION TERMINATED
```

```
#$list forprog
>      1               NAMELIST /NL/A,ROOTA
>      2       10      READ (5,NL)
>      3               ROOTA = SQRT(A)
>      4               WRITE (6,NN)
>      5               GO TO 11
>      6               END
#END OF FILE
#$r *fortg scards=forprog
#EXECUTION BEGINS

 0004        WRITE (6,NN)
                          $
 01) IEY007I ID CONFLICT
                                        IEY022I     UNDEFINED LABEL
      11


 MAIN   0000 0000 0002
#EXECUTION TERMINATED
#$get forprog
#READY.
#4,      write (6,nl)
#$rel
#$list forprog
>      1               NAMELIST /NL/A,ROOTA
>      2       10      READ (5,NL)
>      3               ROOTA = SQRT(A)
>      4               WRITE (6,NL)
>      5               GO TO 11
>      6               END
#END OF FILE
#$r *edit
#EXECUTION BEGINS
:ENTER FILE NAME:
:forprog
:scan 'go to 11'
:       5               GO TO 11
:c '11'10'
:       5               GO TO 10
:mts
#$r *fortg scards=forprog

LINE DELETED: DATA CHECK
#$r *fortg scards=forprog
#EXECUTION BEGINS
#EXECUTION TERMINATED
```

```
#$r -load#
#EXECUTION BEGINS
 &nl  a=25 &end
 &NL
 A= 25.000000    ,ROOTA=  5.0000000
 &END
 ¢
END OF FILE
#EXECUTION TERMINATED
#$r *catalog
#EXECUTION BEGINS
 CMSTOMTS.SYS
 TAPELBL.FORT
 CS018A.PLI
 SELECT.TEXT
 FORPROG
 USER SID1 HAS   5      FILE(S) WITH TOTAL SIZE OF    18           PAGES
#EXECUTION TERMINATED
#$des forprog
#FILE "FORPROG" IS TO BE DESTROYED.  PLEASE CONFIRM.
?ok
#DONE.
#$sig
#OFF AT 09:16.26
#ELAPSED TIME     2011.616  SEC.
#CPU TIME USED      10.966  SEC.
#STORAGE USED      167.07   PAGE-SEC.
#DRUM READS          0
#APPROX. COST OF THIS RUN   $2.41
#FILE STORAGE       18 PG-HR.    $.01

MTS (LA85-0016)
#WELCOME TO THE WONDERFUL WORLD OF MTS.
#
```

7. SUBMITTING BATCH JOBS FROM A TERMINAL

By invoking the public file *BATCH (described on the following page) the terminal user can submit a job to the batch facility. This is useful for two reasons. Firstly, the terminal user can not directly obtain output from line printers or card punches. This restriction occurs because these devices will be controlled by the batch facility (i.e. HASP) for efficient device utilization and job output control. Secondly, execution of jobs, which aren't interactive,under the batch facility is more economic than from a terminal.

*BATCH

Contents: The object module to monitor remote batch entry.

Usage: *BATCH is invoked by the $RUN command.

Logical I/O units referenced:
SCARDS - the file or device containing records to be entered as an MTS job.

Examples:

```
$RUN *BATCH
     (SCARDS defaults to *SOURCE*)
$RUN *BATCH SCARDS=AFILE
```

Description: The content of SCARDS's reference is treated as any "batch" job run in MTS. A "receipt number" by which the user may pick up the output is returned to the user. The job's output may be picked up at the computing center when ready (at the time of this writing, usually the morning following its entry) and should be retrieved within one week.

The first statement entered into SCARDS must be $SIGNON

If SCARDS references a device (such as the user's terminal), rather than a file, the following are applicable:

1. Only 480 characters of information will be accepted (six 80 byte lines, twelve 40 byte lines, etc.)
2. If a line of zero length is entered, the line pointer is decremented by one line, that is, the previous line is deleted.