

M T S

The Michigan Terminal System

Volume 4: Terminals and Networks in MTS

| July 1988

The University of Michigan Computing Center
Ann Arbor, Michigan

| *****
*
* This obsoletes the March 1984 edition. *
* *

DISCLAIMER

The MTS Manual is intended to represent the current state of the Michigan Terminal System (MTS), but because the system is constantly being developed, extended, and refined, sections of this volume will become obsolete. The user should refer to the U-M Computing News, Computing Center Memos, and future Updates to this volume for the latest information about changes to MTS.

| Copyright 1988 by the Regents of the University of Michigan. Copying is permitted for nonprofit, educational use provided that (1) each reproduction is done without alteration and (2) the volume reference and date of publication are included. Permission to republish any portions of this manual should be obtained in writing from the Director of the University of Michigan Computing Center.

July 1988

PREFACE

The software developed by the Computing Center staff for the operation of the high-speed processor computer can be described as a multiprogramming supervisor that handles a number of resident, reentrant programs. Among them is a large subsystem, called MTS (Michigan Terminal System), for command interpretation, execution control, file management, and accounting maintenance. Most users interact with the computer's resources through MTS.

The MTS Manual is a series of volumes that describe in detail the facilities provided by the Michigan Terminal System. Administrative policies of the Computing Center and the physical facilities provided are described in other publications.

The MTS volumes now in print are listed below. The date indicates the most recent edition of each volume; however, since volumes are updated by means of CCMemos, users should check the file *CCPUBLICATIONS, or watch for announcements in the U-M Computing News, to ensure that their MTS volumes are fully up to date.

- Volume 1: The Michigan Terminal System, January 1984
- Volume 2: Public File Descriptions, January 1987
- Volume 3: System Subroutine Descriptions, April 1981
- Volume 4: Terminals and Networks in MTS, July 1988
- Volume 5: System Services, May 1983
- Volume 6: FORTTRAN in MTS, October 1983
- Volume 7: PL/I in MTS, September 1982
- Volume 8: LISP and SLIP in MTS, June 1976
- Volume 9: SNOBOL4 in MTS, September 1975
- Volume 10: BASIC in MTS, December 1980
- Volume 11: Plot Description System, August 1978
- Volume 12: PIL/2 in MTS, December 1974
- Volume 13: The Symbolic Debugging System, September 1985
- Volume 14: 360/370 Assemblers in MTS, May 1983
- Volume 15: FORMAT and TEXT360, April 1977
- Volume 16: ALGOL W in MTS, September 1980
- Volume 17: Integrated Graphics System, December 1980
- Volume 18: The MTS File Editor, February 1988
- Volume 19: Tapes and Floppy Disks, November 1986
- Volume 20: Pascal in MTS, December 1985
- Volume 21: MTS Command Extensions and Macros, April 1986
- Volume 22: Utilisp in MTS, May 1988
- Volume 23: Messaging and Conferencing in MTS, July 1988

Other volumes are in preparation. The numerical order of the volumes does not necessarily reflect the chronological order of their

July 1988

appearance; however, in general, the higher the number, the more specialized the volume. Volume 1, for example, introduces the user to MTS and describes in general the MTS operating system, while Volume 10 deals exclusively with BASIC.

The attempt to make each volume complete in itself and reasonably independent of others in the series naturally results in a certain amount of repetition. Public file descriptions, for example, may appear in more than one volume. However, this arrangement permits the user to buy only those volumes that serve his or her immediate needs.

Richard A. Salisbury

General Editor

July 1988

PREFACE TO REVISED VOLUME 4

| The July 1988 edition is an interim version. Much outdated material
| has been removed and the remaining material has been reorganized.

| The sections "The UMnet Computer Network" and "The Merit Computer
| Network" have been combined into the section "Merit/UMnet."

| The material on the \$NET command has been moved from the "Merit
| Computer Network" into its own separate section.

| The following sections have been deleted:

| Introduction to Terminals
| The LA36 DECwriter
| Graphics Terminals
| Microcomputers and UMnet
| Introduction to the Audio Response Unit
| The Audio Response Unit

| A new version of MTS Volume 4 is planned for the future and will
| cover in more detail terminal access, host-to-host communications, file
| transfer, and servers. The new edition will be entitled Networking in
| MTS.

Contents

Preface	3	TABI	47
Preface to Revised Volume 4	5	THAW	48
The Ontel Terminal	11	UCI	48
The Ontel Keyboard	11	WB, WF, WL, WR	49
Cursor-Positioning Keys	13	Appendix B: Ontel Keyboard	
Special Editing Keys	14	Values and Function Codes	50
Additional Control Keys	15	Merit/UMnet	55
Conversational Operation	16	Terminal Connection	
The Conversation Buffer,		Procedures	56
Input Area, and Window	16	Access Into Merit	56
Input/Output Procedures	16	Access Through Another	
Suspending Output	20	Network	58
Line Reentry	20	Arranging and Paying for	
Automatic Blank Fill	21	Network Use	59
Device Commands	22	Getting Help with Network	
Moving the Window	23	Use	59
Output Modes	24	Network Documentation	60
Program Function Keys	25	Host Documentation	60
The FUNCTION Device		Consultation	60
Command	26	Line-Editing and Control	
The Numeric Pad	28	Functions	61
Appendix A: Ontel Device		Device Commands	62
Commands	33	Using Network Batch	64
BAUD	35	Network Execution Jobs	64
BELL	35	Routing Output from MTS	
BLANK	36	Batch Jobs	66
CBEGIN, CEND	36	Network Print Jobs	66
DCC	37	Routing Output to Xerox	
DEFINE	38	Page Printers	66
DEF?	39	Batch File Copying	67
DUMP	39	Host-Dependent Options	68
EMPTY	40	Data Transmission	70
ENTER	40	MTS FDname Modifiers	72
FAST	41	Modes of Operation	73
FREEZE	41	Appendix A: Merit/UMnet	
FUNCTION	42	Device Commands	76
HIST	43	BACKSPECIALECHO	80
LCI	43	BINARY	81
LINE	43	BLANK	82
PAD	44	BROADCAST	83
PAGE	44	CC	84
PFAn, PFBn	45	CLOCK	85
PF?	46	CONNECTIONS	86
RESET	46	CRDELAY	87
ROLL	47	DCC	88
		DEQUEUE	89

DONT	90	Appendix G: FDname	
DUPLEX	91	Modifiers for MTS	
ECHO	92	Connections145
EDIT	93	Appendix H: Network Error	
FLIP	98	Messages146
FORMFEED	99		
GRAB100	The NET Command149
HELLO101	Examples of Using the NET	
HEX102	Command152
INLEN103	Appendix A: NET CLS	
INTERLOCK104	Commands in MTS155
JOB105	Appendix B: .COPY--The	
LCI106	Merit Interactive File	
LINES107	Transfer Facility160
MIX108		
MODE109	Screen-Support Routines165
NPC111	Introduction165
OPERATOR112	Output Devices166
OUTLEN113	Basic Use of the	
PAGEWAIT114	Screen-Support Routines166
PARITY116	Running a Program166
PFX117	Initialization167
QUIT118	Initializing, Defining,	
READER119	and Ending a Screen167
RESET120	Defining Fields168
SCROLL121	Specifying Field Locations	169
SENSE122	Writing the Text of a	
SNS123	Field169
STATUS124	Setting Field Attributes	.170
TABI125	Cursor Position171
TABSPECIALECHO126	Deleting a Field or Screen	171
TBDELAY127	Ending the Screen	
TERMINAL128	Definition Process172
TEST129	Writing and Reading	
UCB130	Screens172
UCI131	Information, Error	
UCO132	Checking, and Attention	
WAIT133	Handling174
WIDTH134	Termination175
?135	Appendix A: Screen-Support	
n+n136	Routine Descriptions176
Appendix B: Terminal		SSATTR177
Characteristics137	SSBGNS179
Appendix C: ASCII Special		SSCREP180
Characters139	SSCTNS181
Appendix D: Merit/UMnet		SSCTRL182
Return Codes141	SSCURS184
Appendix E:		SSDEFF185
Carriage-Control Characters	.143	SSDELF186
Appendix F: Internal Format		SSDELS187
of the Answerback144	SSENDS188
		SSINFO189
		SSINIT192

SSLOCN193	EOF245
SSREAD194	EQ246
SSTERM196	EXPAND, CONTRACT246
SSTEXT197	FAST247
SSWRIT198	FOOTER247
Appendix B: Visual Mode in the File Editor199	FREEZE248
Appendix C: Example Pascal/VS Program201	GRAB, FLIP248
Appendix D: Example VS Fortran Program206	HEX249
Index209	HIST249
The IBM 3278 Display Station .219		INSERT250
The Keyboard221	JF, JB250
Control Keys222	JOBID251
Cursor-Positioning Keys223	JSET, JRESET, JCLEAR251
Special-Editing Keys223	KEYBOARD252
Additional Control Keys224	LINE252
Conversational Operation225	LNC253
Initiating the Session225	ORL253
Conversation Buffer226	PAGE254
Input/Output Procedures226	PAUSE254
Pause/Continue230	PF255
End-of-Files230	PFSAVE, PFRESTORE, PFDELETE256
Attention Interrupts230	PF?257
Device Commands230	QUEUE257
Scrolling231	QUIT258
Output Modes232	RATE258
Lowercase233	REFRESH259
Hard-Copy Output233	RESET259
Program-Function Keys234	ROLL260
Terminating the Session234	SCROLLING260
Appendix A: IBM 3278 Device Commands236	SNS261
ACTIVITY238	TABS262
ATTN240	THAW263
BLANK240	TOP263
BRIGHT241	WAIT264
BUFFER241	WB, WF, WL, WR264
CBEGIN, CEND242	WINDOW265
COMMENT242	YOUTAKEIT265
DCC243	ZIP266
DC?243	?266
DEQUEUE244	Appendix B: IBM 3278 Character Codes267
DUPLEX244	Appendix C: Binary Input/Output on the IBM 3278	269
EMPTY245	Appendix D: IBM 3278 Routines Return Codes272
ENTER245	Appendix E: IBM 3278 Carriage-Control Characters	.273
		Appendix F: The Lee Data Terminal274

July 1988

THE ONTEL TERMINAL

This section describes the use of the Ontel terminal. There are two different models of the Ontel available: the older model OP-1/R (gray and black) and the newer model 1503 (beige and brown). The differences between these terminals are minimal and are outlined in the description that follows.

The Ontel terminal consists of a cathode-ray-tube (CRT) screen and a separate keyboard. The screen is arranged in a 24-line by 80-character format.

The Ontel terminal has an ON/OFF switch, a brightness control, and a contrast control. These controls are located to the right of the screen at the bottom of the enclosure just above the keyboard. On the model OP-1/R, these controls are located on the front of the console; on the model 1503, these controls are located at the rear of the console. The CRT takes about 15 seconds to warm up when the terminal is powered on. The brightness control varies the intensity of the picture. Note that an overly bright picture may damage the CRT and is also uncomfortable to view for extended periods. The contrast control adjusts the contrast between normal- and high-intensity output and usually is set to about two thirds of the highest contrast position.

The cursor on the Ontel is a blinking box which alternates with the character at the same position. The cursor indicates the position in which the next character will be placed.

Information can be entered onto the screen from either the keyboard or the computer. Data entered at the end of a line normally continues onto the next line. Characters typed from the keyboard remain in a hardware buffer and are not transmitted to the computer until the RETURN key is pressed. Thus, changes can be made and errors can be corrected on the screen before the computer reads the information.

THE ONTEL KEYBOARD

The Ontel keyboard is arranged in four sections: a standard keyboard with an ASCII character set, a control pad, a numeric pad, and a function pad. All keys are typematic; that is, they will repeat their function at a rate of about 15 operations per second if held down for at least one second. The layouts for the Ontel keyboard are given below.

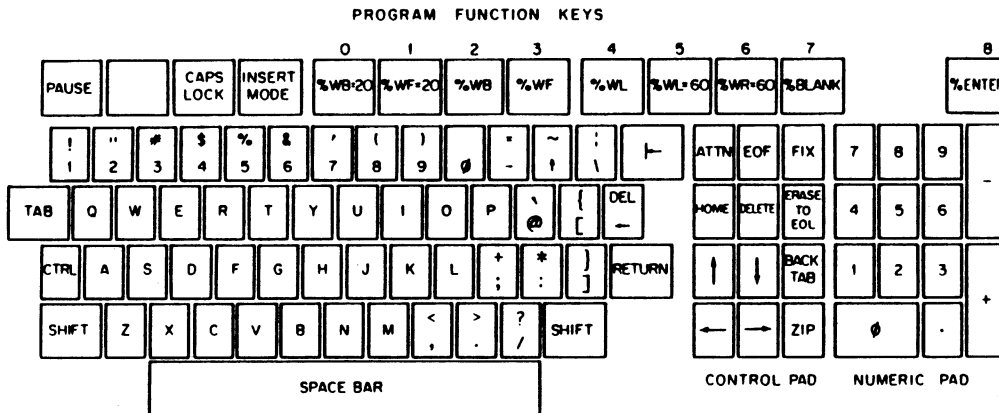


Figure 1: The Ontel Keyboard - Model OP-1/R

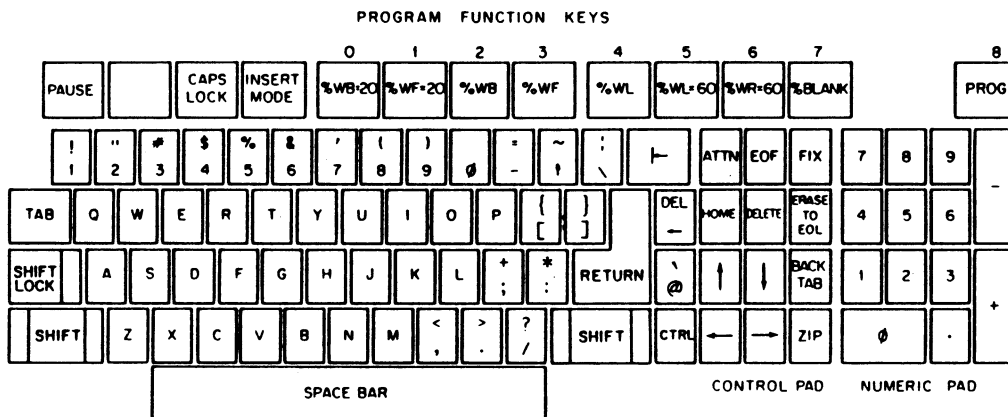


Figure 2: The Ontel Keyboard - Model 1503

The standard keyboard contains the entire upper- and lowercase alphabet, the number set, special characters, and standard controls such as SHIFT, TAB, CTRL, and RETURN.

Ordinary character keys enter the indicated character or symbol onto the screen at the current cursor position. Entry of a character advances the cursor to the next character location. A character on the

July 1988

screen can be replaced by simply overstriking it with another character. Note that the key in the standard keyboard with DEL over a left arrow produces an underscore character when pressed without SHIFT engaged.

Pressing the SHIFT key and another key together enters the uppercase character or symbol of the other key. Pressing the CTRL (control) key and another key together enters the control character associated with the other key. The control characters are displayed as graphic symbols. Some control characters perform special functions on the terminal: CTRL-C (end-of-file), CTRL-E (attention), CTRL-H (backspace), CTRL-I (TAB), and CTRL-M (RETURN).

The RETURN key transmits input to the computer. When the RETURN key is pressed, the entire line on which the cursor is positioned is transmitted to the computer. The position of the cursor on the line is irrelevant. Trailing blanks which were entered from the keyboard by pressing the SPACE BAR are also transmitted. Characters which are part of the "prefix" are not transmitted. A zero-length line can be entered by pressing the RETURN key without having typed any characters. See the section "Input/Output Procedures" for more details.

The control pad has 12 control keys including cursor-positioning controls. The functions of these keys are described below.

The numeric pad has 13 keys arranged in calculator format and includes the numerals 0-9, period, plus, and minus. By default, pressing these keys writes the indicated characters onto the screen, whether SHIFT is engaged or not. However, the function of these keys can be altered by the use of the %PAD device command (in conjunction with the %DEFINE command), as described in the section "The Numeric Pad."

The function pad has 12 function control keys (four of which contain integral status lights) and the %ENTER key. The functions of these keys are described later.

Cursor-Positioning Keys

The SPACE BAR advances the cursor one character position and replaces the character over which it passes with a blank.

The remaining cursor-positioning keys fall into two groups: character-based keys and line-based keys. All of these keys, except BACKSPACE, position the cursor without affecting any of the information already on the screen.

The character-based keys include UP, DOWN, LEFT, RIGHT, and BACKSPACE keys, which move the cursor one character position at a time in the indicated direction. The UP, DOWN, LEFT, and RIGHT keys are in the control pad to the right of the standard keyboard and are labeled with

July 1988

arrows pointing in the appropriate direction. The cursor LEFT and RIGHT keys, when used with the SHIFT key, will move the cursor two positions left or right. The BACKSPACE key in the extreme upper-right corner of the standard keyboard is labeled with a left arrow pointing to a vertical bar. The BACKSPACE key will "backspace and blank" when not in INSERT mode; that is, any character spaced over will be replaced by a blank character. The use of the BACKSPACE key in INSERT mode is explained later.

The line-based keys are the TAB, BACKTAB, HOME, and ZIP keys. The BACKTAB key moves the cursor to the first position of the current line (or the previous line, if the cursor is already in the first position of the current line). The ZIP key moves the cursor to the first unoccupied position of the current line (or the next line, if the cursor is just past the last position). The HOME key moves the cursor to the first character position in the topmost line on the screen. When the HOME key is pressed with the SHIFT key, the cursor is moved to the first available position after the prefix in the last nonblank line on the screen. If TAB stops are not set, the TAB key will position the cursor to the first position of the next displayed line. If TAB stops are set and TABbing is "enabled," the cursor will be positioned at the next TAB stop to the right. When beyond the last stop, the cursor is moved to the first position of the next displayed line.

Special Editing Keys

A character can be deleted from the screen by pressing the DELETE key located in the control pad or the DEL key (the SHIFTed left-arrow key) in the standard keyboard. All characters to the right of the deleted character are shifted left by one position. The character after the deleted character is then in the cursor position.

A word can be deleted by pressing the DELETE key and the SHIFT key together. If the cursor is at a nonblank character, the word containing that character and one blank to the right of the word (if any) are deleted. If the cursor is on a blank, only the blank is deleted. Again, all characters to the right of the deleted ones are shifted to the left. If an attempt is made to delete a character from a line that is longer than the screen width, that line first will be moved down to the last line of the screen (if it is not already there).

Characters can be inserted by pressing the INSERT key to enter insertion mode (indicated by the light under the INSERT key). When in insertion mode, each character typed is inserted before the character in the cursor position. The cursor and the remainder of the line to its right are displaced one position to the right. If the insertion reaches the rightmost character position on the screen or if the line is longer than the screen width, the line is moved down to the last line on the screen (if it is not already there). The maximum line length is 255 characters minus the current prefix length. While the terminal is in

July 1988

insertion mode, the cursor can be moved and characters can be inserted at various places on the screen. The DEL key deletes the character pointed to by the cursor and moves the remainder of the line one position to the left. In insertion mode, the BACKSPACE key deletes the character to the left of the cursor and moves the remainder of the line one position to the left.

A deletion/insertion example follows (the underscore "_" represents the cursor):

change this_thing	Initial display
change thi_ thing	BACKSPACE key pressed
change th_ thing	INSERT key(on); BACKSPACE pressed
change th_thing	DEL key pressed
change tha_thing	'a' typed
change thas_thing	's' typed
change thas_ thing	CURSOR LEFT
change that_thing	INSERT key(off); 't' typed
change that_thing	SHIFT-DEL key pressed
change _	SHIFT-DEL key pressed again

Additional Control Keys

The FIX key is useful when portions of the screen have been accidentally altered. Pressing this key rewrites the current screen with a clear input area. (The input area is described below.) FIX also cancels the effect of the %BLANK device command.

Pressing the ERASE TO EOL key deletes all characters to the right of the cursor position. To delete an entire line, the BACKTAB key can be used to move the the cursor to the beginning of the line before the ERASE EOF key is pressed. The cursor remains in position for reentering the line.

Uppercase-only mode can be enabled or disabled with the CAPS LOCK key (in the leftmost group of the function pad). Uppercase-only mode is indicated by the light on the CAPS LOCK key. Only alphabetic characters are affected. When enabled, the alphabetic keys generate only uppercase characters regardless of the use of the SHIFT key.

The ATTN key or the CTRL-E key generates an attention interrupt and moves the line at the cursor to the last line of the screen.

The EOF key or the CTRL-C key generates an end-of-file signal. The screen is also rewritten as with the use of the FIX key.

The PAUSE key suspends the transmission of data to the terminal by the system. The key will light up while output is suspended. Pressing the PAUSE key again will resume output. Pressing the ATTN key simultaneously cancels a PAUSE and generates an attention interrupt.

CONVERSATIONAL OPERATIONThe Conversation Buffer, Input Area, and Window

The Ontel has a special hardware memory area called the conversation buffer, which is used to save the most recent portion of the terminal session dialogue. When the user submits a line from the terminal to the system, or when the system writes an output line to the terminal, that line is added to the end of the conversation buffer. As the terminal session progresses, the oldest lines of the session are discarded from the conversation buffer as new lines are added. Note that device commands are not recorded in the conversation buffer unless they are transmitted.

The Ontel screen is divided into two memory areas, the input area and the window.

The input area is always displayed as the last nonblank line on the screen. This area acts as a holding buffer between the user and the system. Input lines are sent from the input area to the system. Note that when a user types characters in the input area, those characters are not sent to the system until the RETURN key is pressed. Thus changes may be made to the line before it is sent to the system.

The lines displayed above the input area are called the window. At any point in a terminal session, the window contains a copy of the first 80 characters of each of the most recent lines in the conversation buffer. Up to 22 lines can be displayed at one time. Thus this part of the Ontel screen is termed a "window" through which portions of the conversation buffer may be viewed. The section "Moving the Window" describes how the window can be moved to the right or left to view the undisplayed parts of long lines, and how it can be moved forward or backward to view all parts of the conversation buffer.

Input/Output Procedures

After an Ontel terminal has been turned on and it has warmed up, a percent sign (%) followed by a blinking box (the cursor) will appear in the upper lefthand corner of the screen (the underscore "_" is used to represent the cursor):

July 1988

```
%_
```

At this point the input area contains only the percent sign (%), which is the default device command character for the Ontel.

If the terminals are connected to the system through UMnet (e.g., the Ontels at UNYN and NUBS), the user should press the ATTN key on the control pad. A line similar to the following will be displayed:

```
%Merit:Hermes (NUB128:MP13:OP/VIS)

Which Host?um
  MTS Ann Arbor (AE47-00172)

#_
```

The above sequence shows that the terminal is connected to the system via Secondary Communications Processor (SCP) 1 at NUBS (port 13). The MTS device port is AE47 and the job number is 00172. If the terminal is connected to MTS by way of a modem and telephone lines, information similar to that shown above will be displayed as soon as the connection is made.

The last line displayed, the input area, contains the MTS input prefix, a pound sign (#). Each input and output line has a prefix associated with it which is used to identify the system component (e.g., MTS command, file editor command) that issued the output line or expects an input line. This prefix is usually a single character. When the screen is windowed right or left, each displayed line always begins with the prefix associated with the line, followed by the windowed portion of the line. The input area always appears at the bottom of the screen, regardless of any windowing. Note that when a line is sent to the system, the prefix is not sent.

Next the SIGNON command is typed:

```
%Merit:Hermes (NUB128:MP13:OP/VIS)

Which Host?um
    MTS Ann Arbor (AE47-00172)

#signon wxyz_
```

The RETURN key has not yet been pressed. The text "signon wxyz" exists only in the input area and not in the conversation buffer. Next the RETURN key is pressed causing "signon wxyz" to be transmitted to the system. The text is then written to the end of the conversation buffer.

Next the system will transmit a line requesting the user's password followed by line with a question mark (?) prefix:

```
%Merit:Hermes (NUB128:MP13:OP/VIS)

Which Host?um
    MTS Ann Arbor (AE47-00172)

#signon wxyz
#Enter password.
?_
```

At this point the input area contains "?". Next the user should enter the password. If the terminal is connected to the system through UMnet, the password will not be displayed on the terminal as it is typed.

Next MTS will print the SIGNON statistics and another pound sign prefix prompting for a command:

July 1988

```

%Merit:Hermes (NUB128:MP13:OP/VIS)

Which Host?um
    MTS Ann Arbor (AE47-00172)

#signon wxyz
#Enter password.
?
#Terminal,Normal,Univ/Gov't
#Last signon was at 12:32:43, Tue Jan 31/84
#User WXYZ signed on at 13:41:15, Mon Jan 30/84
#_

```

Subsequent lines may then be entered as with the SIGNON command.

When the terminal receives an output line, it places the prefix and the line in the conversation buffer. The prefix and the line are then written on the next available screen line. If the output line is longer than 80 characters (with prefix), only the first 80 characters are initially displayed. In ROLL output mode (the default), lines are added until the 22nd screen line has been used. Subsequent output lines cause the screen to roll up one line and the new line is placed in the 22nd slot.

When the system is requesting input, the terminal will display the prefix and leave the cursor at the beginning of a null input area. Normally the length of the input area is 255 minus the prefix length. However, if the input area begins at the 22nd or 23rd screen line, it will extend only to the end of the screen and thus may be as short as 160 characters. Typing will be suppressed if the user attempts to generate longer lines.

Input can be entered asynchronously; that is, lines can be entered even though the system has not yet requested input. There is always an input area on the screen and the prefix will indicate whether or not the system is expecting input (except when the prefix is the same as the device command character). Whenever the device command character is displayed, input can be typed in and transmitted to the the system by pressing the RETURN key. If an output line is received as the user is typing, the input area is saved before any output lines are written to the screen and is then rewritten to the new input area.

The editing facilities of the terminal can be used to make corrections to an input line before the line is entered. Note that although the cursor can be moved into the prefix positions, any attempt to enter characters in such a position will be suppressed. Certain control characters such as DEL and ERASE EOF will not work in these positions.

July 1988

The cursor must be moved to a nonprefix position before typing can proceed.

Suspending Output

The PAUSE key, located in the function pad, can be used to suspend the transmission of output to the terminal. When output is suspended, the PAUSE key will light up. The PAUSE key can be pressed again to resume the transmission of output. If the terminal is in PAGE mode (see description of PAGE mode), output will be suspended when a full screen has been received and displayed. Pressing the PAUSE key allows the next screen to be received and displayed. Pressing the ATTN key will cancel the effect of the PAUSE key and simultaneously generate an attention interrupt.

Line Reentry

In addition to typing in the input area, a user may enter a line already displayed in the window by moving the cursor to the desired line and pressing the RETURN key. All characters in the line containing the cursor will be written in the input area and then transmitted to the system. Note that any characters already in the input area will be lost when this is done. If the line to be reentered is not entirely displayed on the screen, the terminal will automatically append undisplayed parts before the line is moved to the input area and sent to the system.

Any line on the screen can also be edited before it is transmitted. If an attempt is made to insert characters in a line that is longer than 80 characters, the line will be written in the input area where insertion can be continued. These features allow the user to easily correct an input line that was rejected by the system. For example, consider the following portion of a terminal session:

```
#edit mylib
:atler@@nv /f 'ABC'123'
:Unrecognizable EDIT command
: _
```

July 1988

Rather than retyping the correct version of the entire line into the input area, the user may simply move the cursor to the invalid line and use the special editing keys to make the necessary changes.

```
#edit mylib
:alter@@nv /f 'ABC'123'
:Unrecognizable EDIT command
:
```

The user may then reenter the corrected line by pressing the RETURN key. Note that when the screen is rewritten, the original invalid line will be displayed, and the corrected version of that line will be the newest input line in the conversation buffer, as shown here:

```
#edit mylib
:atler@@nv /f 'ABC'123'
:Unrecognizable EDIT command
:alter@@nv /f 'ABC'123'
: _
```

This happens because the window contains only a copy of the conversation buffer. Thus changes made to lines in the window are not made to the corresponding lines in the conversation buffer, and when the screen is rewritten the window again contains a copy of the original lines in the conversation buffer.

Automatic Blank Fill

Blanks are not appended to the end of a line when the cursor is moved beyond the end of the line (unless it is moved by pressing the SPACE BAR). Thus, when the cursor is positioned beyond the end of a line and then the RETURN key is pressed, trailing blanks are not transmitted. However, if a character is typed when the cursor is beyond the end of the line, the intervening space is automatically filled with blanks. No visible sign is given that this has occurred except for normal movement of the cursor to the right as the character is typed.

Device Commands

Device commands can be used to control certain functions of the terminal such as windowing, tabbing, and the translation of characters to uppercase on input. Device commands may be entered whenever the system is expecting input.

There are two levels of device commands. The first level is the set of Ontel device commands that are processed within the Ontel terminal. These commands are described in this section with the complete list given in Appendix A. The second level is the set of UMnet device commands that are processed by the UMnet Computer Network. These device commands are described in the section "The UMnet Computer Network" in this volume.

All device commands must begin with the device-command character, which is initially the percent sign (%). Lines that begin with a single device-command character are examined by the Ontel terminal when they are entered to determine if they are valid Ontel device commands. If they are, the command is processed and erased from the screen. If the command is not a valid Ontel device command, it is passed on to UMnet as a command line (not a data line) for device-command processing. If the command is valid, it is acted upon by UMnet; if it is invalid, it is rejected (an error message will be printed). If user wishes to send a data line that begins with the device-command character, the device-command character must be doubled (or the %DCC command may be used to change the current device-command character).

| The %RESET, %LCI, and %UCI device commands are first processed by the Ontel and then by UMnet.

The acknowledgement of Ontel device commands depends on the type of command entered. If the command calls for an overt action, such as scrolling, its execution constitutes the acknowledgment. If the command merely sets a parameter, the user must assume the command had the desired effect. In the case of an invalid parameter, the command is ignored and no error message is generated. Normally, the device command is not recorded in the conversation buffer.

Only the MTS \$CONTROL command or the CONTROL subroutine can be used to send Ontel device commands to the terminal from the system (e.g., from a sigfile using the \$CONTROL command or from a program using the CONTROL subroutine). Otherwise, all lines sent to the terminal from the system are treated as data lines even if they begin with the device-command character.

July 1988

Moving the Window

The Ontel provides several ways for a user to "move" the window, so that any part of the conversation buffer can be displayed on the screen. The window can be moved right or left, to display all parts of lines that are longer than 80 characters, or it can be moved forward and backward, to display all parts of the terminal session. Moving the window is also called "windowing" or "scrolling."

One way to move the window is to enter one of the four device commands

```
%WR=n (window right)
%WL=n (window left)
%WB=n (window backward)
%WF=n (window forward)
```

The %WR=n command displaces the window to the right by "n" characters. This is convenient for viewing long lines. The %WL=n command moves the window to the left. The prefix characters are not affected by these commands and remain displayed. The %WB=n command moves the window backward in the conversation buffer by "n" lines. This is useful for viewing previous parts of the conversation. %WF=n moves the window forward by "n" lines. There is no lateral displacement involved in the window-backward and window-forward operations. If "n" is omitted on a window command, the extreme movement is assumed. For example, %WL moves the window to the extreme left of the conversation buffer, and %WB causes the first window in the conversation buffer to be displayed. The equal sign may be omitted from the command.

Another way to move the window is to use the program function keys which provide windowing facilities. For example, the window may be moved forward 20 lines by pressing the %WF=20 key. Similarly, pressing the %WL key moves the window to the extreme left portion of the conversation buffer, and pressing the %WB key moves the window to the first set of lines in the buffer. The use of the program function keys is further described below, in the section "Program Function Keys."

There are also three control keys which may be used to scroll the window to any part of the conversation buffer. The UP and DOWN cursor-positioning keys, when pressed with the SHIFT key engaged, will move the text in the window one line up or down. These keys are located in the control pad and are labelled with an arrow pointing in the given direction. (Note that it is the text, not the window, which moves in the given direction.) The ERASE TO EOL key, when pressed with the SHIFT key engaged, moves the window such that the last set of lines in the conversation buffer is displayed on the screen. The number of lines displayed in this window is the same as the last time the last line in the conversation buffer was displayed on the screen, i.e., %WF.

Each time the conversation buffer is updated, i.e., after a complete line is sent to or received from the system, the window is forced to the

July 1988

most recent line, overriding any backward displacement. However, lateral displacement is not affected. Note that any attempt to move the window beyond the boundaries of the conversation buffer will be ignored.

When the terminal is first powered on, or when the conversation buffer is emptied via the %EMPTY command, the three-digit window number is initialized to one. This number is displayed in the lower righthand corner of the screen (leading zeros are suppressed). If the user has used a control key or issued a command which moves the window to the right, the window number will be followed by a space and a horizontal displacement value. No displacement value will be displayed if the displacement to the right is zero.

For the purpose of determining window numbers, all lines in the conversation buffer are assumed to be numbered consecutively, starting with one. If the last line displayed on the screen is "m", the window number will be $\text{ceiling}(m/22)$, i.e., the smallest integer equal to or greater than $m/22$.

Output Modes

Four output modes are provided: ROLL, LINE, PAGE, and FAST. These modes may be specified by entering the device commands

```
%LINE
%ROLL
%PAGE
%FAST
```

In ROLL mode, each new input or output line added to the conversation buffer removes the top line from the screen and adds the new line to the bottom of the screen. Thus, the output appears to be rolling up the screen. ROLL mode is equivalent to LINE mode with %HIST=22 in effect. ROLL mode is the default.

In LINE mode, output lines appear in the next available screen line as they are produced by the system. When the 22nd screen line has been used, the screen is erased, the most recent few lines in the conversation are displayed at the top of the screen, and the input/output process continues at the next available screen line. By default 7 most recent lines are displayed when the screen is rewritten. The number of most recent lines displayed may be changed by entering the %HIST=n device command, where "n" may be any integer from 1 to 22.

In PAGE mode, output lines from the system are written into the conversation buffer, but are not displayed on the screen until 19 new lines have been received or until MTS requests an input line. At this time the last 21 lines in the conversation buffer are displayed and the system is told to stop the generation of output so that the new screenful can be viewed. Pressing the PAUSE key will resume the output of up to another 19 lines.

July 1988

In FAST mode, output lines from the system are written into the conversation buffer, but are not displayed on the screen until MTS requests an input line. At this time the last 21 lines in the conversation buffer are displayed. In the case of large amounts of output, some of it may be lost from the top of the conversation buffer.

Program Function Keys

Across the top of the keyboard is the program function pad. The nine rightmost keys are called program function keys and are labelled with Ontel device commands. They are arranged in the following format:

%WB=20	%WF=20	%WB	%WF	%WL	%WL=60	%WR=60	%BLANK	%ENTER
PF0	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8

On the Ontel Model 1503, the %ENTER key is labelled as PROG. By default, the program function keys execute the device commands that are inscribed on the key tops, both in the unshifted and shifted positions.

Each program function key has a program function associated with it. These program functions are named PFA0 through PFA8 (for the unshifted keys) and PFB0 through PFB8 (for the shifted keys).

PFA0	PFA1	PFA2	PFA3	PFA4	PFA5	PFA6	PFA7	PFA8
Unshifted								
PFB0	PFB1	PFB2	PFB3	PFB4	PFB5	PFB6	PFB7	PFB8
Shifted								

Each program function is assigned a character string which is transmitted to the system when the corresponding program function key is pressed. By default, the characters strings assigned to the above program functions are the device commands that are inscribed on the top of the keys. For example, pressing either the unshifted or shifted PF7 key transmits the character string "%BLANK" to the system which executes the %BLANK device command.

New strings can be assigned to the program functions by issuing the %PFAn and %PFBn device commands. These commands are given in the form

```
%PFAn=string
%PFBn=string
```

July 1988

where "n" is an integer from 0 to 8, and "string" may be any string of characters. For example, the following sequence of device commands will reconfigure the shifted program functions keys to transmit different device commands, MTS commands, and other replies to the system:

```
%PFB0=%PAGE
%PFB1=%LINE
%PFB2=%ROLL
%PFB3=%PF?
%PFB4=$RUN *FORUM
%PFB5=$MESS RETRIEVE NEW HEADER
%PFB6=$CONTROL *PRINT* PRINTER=PAGE
%PFB7=$RELEASE *PRINT*
%PFB8=CANCEL
```

Note that if the program-function string is to be used as a device command, that string must always be specified with an initial percent sign (%), even when the device-command character has been changed.

The %PF? device command may be used to display the current status of all the program function strings.

Note that whenever a program function key is pressed, if the string associated with that key is a valid nontransmitted device command, the line at which the cursor is positioned and the lateral position of the cursor are saved; that is, the line, including any undisplayed parts, is moved to the input area after the command is processed. If the cursor is in the input area when the PF key is pressed, the contents of the input area will remain unchanged. The cursor is also positioned so that its relative lateral position remains unchanged.

The FUNCTION Device Command

Each key on the Ontel keyboard has a function code associated with it. When the key is pressed, the function code causes one of the following operations to occur:

- (1) A character or sequence of characters is inserted into the input area (e.g., pressing the SHIFT-A key inserts the character A into the input area).
- (2) A line-editing or terminal operation is performed (e.g., pressing the DELETE key deletes the character pointed to by the cursor and pressing the ATTN key generates an attention interrupt).
- (3) A character or sequence of characters is transmitted to the system (e.g., pressing the PF7 key transmits (by default) the %BLANK device command to the system).

The %FUNCTION device command may be used to change the function code that is assigned to each key by default, thus allowing the user to

July 1988

redefine the operation of a key. The %FUNCTION command is given in the form

```
%FUNCTION n m
```

where "n" is the hexadecimal value used to identify the key being changed and "m" is the function code that is to be assigned to the key.

Table 1 of Appendix B to this section lists the hexadecimal values used to identify each key along with the default function code that is associated with the key. Table 2 of Appendix B lists the definitions of the function codes. For example from Table 1, the unshifted PAUSE key is identified by the hexadecimal value E0 and has a default function code of 20. From Table 2, the function code 20 definition is "Enter/Exit Pause Mode". The shifted PAUSE key is identified by the hexadecimal value F0 and has the same default function code of 20. Hence, both the unshifted and shifted PAUSE keys have the same effect by default. The PAUSE key could be redefined by the user so that the shifted PAUSE key would have a different function code from the unshifted PAUSE key; for example, the shifted PAUSE key could be redefined to generate an attention interrupt by giving the command

```
%FUNCTION F0 21
```

Note in Table 1 that the hexadecimal identifiers for CTRL-PAUSE and CTRL-SHIFT-PAUSE are the same as for PAUSE and SHIFT-PAUSE, respectively. This means that, for example, the PAUSE and CTRL-PAUSE keys cannot be assigned different function codes. Most of the keys in the function pad, the control pad, and the numeric pad are hardware designed so that the CTRL key will not produce a different function.

The %FUNCTION command is most useful for redefining keys to use either program functions or define functions. Since there are 32 available program functions (PFA0-15 and PFB0-15) and only 18 combinations of program function keys (unshifted and shifted PF0-8), other keys must be used to access the other 14 program functions. By default, the program function keys use program functions PFA0-8 and PFB0-8. Also by default, the numeric keys in the standard keyboard are defined as program functions (PFA1-PFA12) when used with the CTRL key (e.g., CTRL-1 is defined as program function PFA1 and CTRL-0 is defined as PFA10). The following example illustrates how some of the numeric keys could be redefined to the program functions PFA9-15 and PFB9-15.

```
%FUNCTION B1 89    (redefine CTRL-1 as PFA9)
%FUNCTION B2 8A    (redefine CTRL-2 as PFA10)
...
%FUNCTION B7 8F    (redefine CTRL-7 as PFA15)
%FUNCTION C1 99    (redefine CTRL-SHIFT-1 as PFB9)
%FUNCTION C2 9A    (redefine CTRL-SHIFT-2 as PFB10)
...
%FUNCTION C7 9F    (redefine CTRL-SHIFT-7 as PFB15)
```

July 1988

The %PFAn and %PFBn device commands can then be used to assign definitions to the these program functions as shown in the section "Program Function Keys" above.

Likewise, there are 32 available define functions (DEFINE 0-31), but only the first 26 (DEFINE 0-25) are assigned default definitions for use with the numeric pad in define mode (see the section "The Numeric Pad" below). The remaining 6 define functions may be assigned to other keys on the standard keyboard pad in the same manner as illustrated above for the program functions. For example, Define Function 26 through 31 could be assigned to the CTRL-numeric keys as follows:

```
%FUNCTION B8 CA    (redefine CTRL-8 as Define 26)
%FUNCTION B9 CB    (redefine CTRL-9 as Define 27)
%FUNCTION BA CC    (redefine CTRL-0 as Define 28)
%FUNCTION AD CD    (redefine CTRL-- as Define 29)
%FUNCTION 1E CE    (redefine CTRL-↑ as Define 30)
%FUNCTION 1C CF    (redefine CTRL-\ as Define 31)
```

The %DEFINE device command can then be used to assign definitions to these define functions as shown in the section "The Numeric Pad" below.

All the keys on the Ontel keyboard may have their functions redefined, except for the CTRL-SHIFT-PF8 key which is always defined as "Reset Terminal". The user may not redefine character keys to enter characters other than the character that is inscribed on the top of the key. For example, the A key cannot be redefined to enter the character B, although the A key could be redefined to generate an attention interrupt. However, if this were done, then the character could not be entered from the keyboard.

To restore a key to its original function, the command

```
%FUNCTION n
```

can be used. For example, if the lowercase "a" key (value 61) had been redefined, then the command

```
%FUNCTION 61
```

will restore it to its original lowercase "a" function.

The %RESET device command may be used to reset all of the keys, program functions, and define functions to their default status.

The Numeric Pad

To the far right of the keyboard is the numeric pad. This pad is labelled with the numerals 0 through 9, period, plus, and minus. The numeric pad may be used in three different modes: numeric mode, define

July 1988

mode, and PF-key mode. The %PAD device command which is given in the form

```
%PAD={NUMERIC|DEFINE|PFKEYS}
```

controls which mode is in effect.

Numeric mode is the normal (default) mode of operation. In numeric mode, each key enters into the input area the character that is inscribed on the top the of key, i.e., the numeral 0 through 9, period, plus, or minus. Thus, the numeric pad works much like a desk calculator pad.

7	8	9	
4	5	6	-
1	2	3	
0	.		+

Unshifted/Shifted

In define mode, each key may be redefined to enter a different character or sequence of characters into the input area. Each key has a define function associated with it. There are 32 define functions, numbered from 0 through 31. By default, the define functions are organized in the following format:

Df 7	Df 8	Df 9	
Df 4	Df 5	Df 6	Df 22
Df 1	Df 2	Df 3	
Df 0	Df 20	Df 21	

Unshifted

Df 17	Df 18	Df 19	
Df 14	Df 15	Df 16	Df 25
Df 11	Df 12	Df 13	
Df 10	Df 23	Df 24	

Shifted

This arrangement of define functions may be changed by the %FUNCTION device command but is not normally done. What is useful to change is the definition assigned to each define function. By default, the characters assigned to the define functions are the characters that would normally be used by the numeric pad in numeric mode, e.g., the character 7 is assigned to the Define 7 and 17 functions.

The %DEFINE device command is used to assign new character strings to the define functions. This device command is given in the form

```
%DEFINE n=string
```

where "n" is an integer from 0 to 31. Only the first 25 define functions have default values (the values that correspond to the numeric pad); the Define Functions 26 through 31 are undefined initially. The following example illustrates how the numeric pad may be reconfigured:

```
%DEFINE 7=$RUN *PAGEPR SCARDS=          PAR=PORTRAIT
%DEFINE 8=$RUN *PAGEPR SCARDS=          PAR=LANDSCAPE
%DEFINE 9=$RELEASE *PRINT*
%DEFINE 4=$COPY          *PRINT*
%DEFINE 5=$LIST          *PRINT*
%DEFINE 6=$DISPLAY RECEIPTS
%DEFINE 1=$DISPLAY *PRINT*
%DEFINE 2=$CONTROL *PRINT* ROUTE=UNYN
%DEFINE 3=$CONTROL *PRINT* ROUTE=NUBS
%DEFINE 0=$MESSAGE RETREIVE NEW HEADER
%DEFINE 20=$EDIT
%DEFINE 21=$SY Q
%DEFINE 22=$DISPLAY TIMESPELLEDOUT
%PAD=DEFINE
```

This device-command sequence reconfigures the unshifted numeric pad to insert MTS commands into the input area. The shifted numeric pad is unchanged and hence may be used for normal numeric calculations. By placing this device-command sequence into a sigfile, a user may reconfigure the terminal for every session. The %PAD=NUMERIC command may be given subsequently to restore the numeric pad for numeric use. The %DEF? command may be used to display the current definitions of the define functions.

In PF-key mode, the keys may be redefined to transmit different characters or sequences of characters to the system. Note that this differs from define mode where the characters are only entered into the input area instead of being transmitted directly to the system. Each key has a program function associated with it. There are 32 program functions available, labelled PFA0 through PFA15 and PFB0 through PFB15. By default, the program functions for the numeric pad are organized in the following format:

PFA13	PFA14	PFA15	
PFB0	PFB1	PFB2	PFA0
PFB3	PFB4	PFB5	
PFB6		PFB7	PFB8

Unshifted

PFA1	PFA2	PFA3	
PFA4	PFA5	PFA6	PFA0
PFA7	PFA8	PFA9	
PFA10		PFA11	PFA12

Shifted

This particular arrangement of program functions was chosen so that the numeric pad could be used to simulate the program function pad of an IBM 3278 Display Station when used in visual mode with the MTS file editor (see MTS Volume 18, The MTS File Editor, for further details). Some of

July 1988

these program functions are also the same as those defined for the program function pad across the top of the Ontel keyboard (notably PFA0 through PFA8 and PFB0 through PFB8). By default, the initial definitions of the program functions are Ontel device commands:

%WL 1	%WR 1	%PAGE	
%WB 20	%WF 20	%WB	%WB 20
%WF	%WL	%WL 60	
%WR 60		%BLANK	%ENTER

Unshifted

%WF 20	%WB	%WF	
%WL	%WL 60	%WR 60	%WB 20
%BLANK	%ENTER	%WB 1	
%WF 1		%WF	%THAW

Shifted

The arrangement of program functions for the numeric pad may be changed by the %FUNCTION device command as described above. This allows the user to reconfigure the terminal so that the numeric-pad keys invoke different program functions than the program-function keys. The %PFAn and PFBn device commands are used to assign new definitions to the program functions. These device commands are given in the form

```
%PFAn=string
%PFBn=string
```

where "n" is an integer from 0 to 15.

The following example illustrates how the numeric pad may be reconfigured:

```
%FUNCTION C0 90 (assign unshifted 0 to PFB0)
...
%FUNCTION C9 99 (assign unshifted 9 to PFB9)
%FUNCTION CB 9A (assign unshifted . to PFB10)
%FUNCTION D0 9B (assign shifted 0 to PFB11)
...
%FUNCTION D4 9F (assign shifted 4 to PFB15)
%FUNCTION D5 89 (assign shifted 5 to PFA9)
...
%FUNCTION D9 8D (assign shifted 9 to PFA13)
%FUNCTION DB 8E (assign shifted . to PFA14)
%FUNCTION DD 81 (assign shifted + to PFA1)
%FUNCTION CD 81 (assign unshifted + to PFA1)
```

The above sequence of %FUNCTION device commands reassigns the program functions for the numeric pad into the following format. Note that both the unshifted and shifted minus (-) keys are not reassigned.

PFB7	PFB8	PFB9	
PFB4	PFB5	PFB6	PFA0
PFB1	PFB2	PFB3	
PFB0		PFB10	PFA1

Unshifted

PFA11	PFA12	PFA13	
PFB15	PFA9	PFA10	PFA0
PFB12	PFB13	PFB14	
PFB11		PFA14	PFA1

Shifted

The following sequence of %PFAn and %PBBn device commands assigns new definitions to the program functions. The unshifted keys are configured as an MTS command pad and the shifted keys are configured as a device command pad.

```

%PFB7=$SOURCE DATASETUP
%PFB8=$RUN DATAPROG SCARDS=DATA1 SPRINT=-DATA
%PFB9=$EDIT -DATA
%PFB4=$RUN *PAGEPR SCARDS=-DATA PAR=PORTRAIT,ONESIDED
%PFB5=$RELEASE *PRINT*
%PFB6=$DISPLAY RECEIPTS
%PFB1=$DISPLAY *PRINT*
%PFB2=$CONTROL *PRINT* ROUTE=UNYN
%PFB3=$CONTROL *PRINT* ROUTE=NUBS
%PFB0=$MESSAGE RETREIVE NEW HEADER
%PFB10=$SIG $
%PFA0=%PAGE
%PFA1=%LINE
%PFA11=%WB 20
%PFA12=%WR 60
%PFA13=%EMPTY
%PFB15=%WF 20
%PFA9=%WL 60
%PFA10=%FREEZE=5
%PFB12=%WF
%PFB13=%WL
%PFB14=%THAW
%PFB11=%PAD=NUMERIC
%PFA14=%PAD=DEFINE
%PAD=PFKEYS
    
```

Again, this device-command sequence can be inserted into a sigfile to reconfigure an Ontel terminal at sign-on time.

July 1988

APPENDIX A: ONTEL DEVICE COMMANDS

Ontel device commands must begin with the current device-command character, initially a percent sign (%), immediately followed by the command name or a permissible abbreviation. The minimum abbreviation for each command is underlined in the command descriptions. The command can be followed by zero or more parameters, depending on the particular command. Delimiters such as equal signs (=) and commas are ignored by the command processor and can be replaced with blanks.

This list only includes Ontel device commands; it is also possible to issue many of the UMnet device commands (see the section "The UMnet Computer Network" in this volume).

The following notation is used to describe commands:

UPPERCASE	must appear as shown,
lowercase	replace appropriately,
	separates alternative forms,
{ }	encloses alternative forms,
[]	encloses optional parameters,
_	indicates command abbreviations,
...	indicates repeatable fields.

Summary of Device Commands

<u>BAUD</u> =n	set baud rate
<u>BELL</u> =[n]	specify the right-margin warning bell
<u>BLANK</u>	set the blanking switch
<u>CBEGIN</u>	specify first line written by %DUMP
<u>CEND</u>	specify last line written by %DUMP
<u>DCC</u> ="x"	define device command character
<u>DEFINE</u> n=string	define replacement string
<u>DEF?</u>	display replacement string definitions
<u>DUMP</u>	write conversation buffer to *MSOURCE*
<u>EMPTY</u>	empty conversation buffer
<u>ENTER</u>	write line at cursor to conv. buffer
<u>FAST</u>	set fast mode
<u>FREEZE</u> =[n]	freeze "n" lines on the screen
<u>FUNCTION</u> n1 m1 [n2 m2 ...]	redefine a key function
<u>HIST</u> =[n]	set number of lines of history
<u>LCI</u> =[{ON OFF}]	enable lowercase input
<u>LINE</u>	set line mode
<u>PAD</u> =[{DEFINE NUMERIC PFKEYS}]	define numeric pad usage
<u>PAGE</u>	set page mode
<u>PFAn</u> =definition	define program function strings
<u>PFBn</u> =definition	define program function strings
<u>PF?</u>	display PF definitions
<u>RESET</u>	reset device command parameters
<u>ROLL</u>	set roll mode
<u>TABI</u> =[{ON OFF}] ["x"] [t,...]	set input tab stops
<u>THAW</u>	thaw frozen lines
<u>UCI</u> =[{ON OFF}]	set uppercase switch
<u>WB</u> =[n]	window back "n" lines
<u>WF</u> =[n]	window forward "n" lines
<u>WL</u> =[n]	window left "n" positions
<u>WR</u> =[n]	window right "n" positions

July 1988

BAUD

Command: BAUD=n

where "n" is any of the legal baud rates: 50, 76, 110, 135, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, and 19200. The baud rate 76 actually corresponds to a baud rate of 75 which cannot be specified because of the table structure used to store the valid baud rates. The baud rate 135 corresponds to a baud rate of 134.5 which cannot be specified because it is fractional.

Purpose: To change the baud rate for communication with the system. The initial baud rate is determined by the setting of hardware switches within the terminal. The baud rate can be changed at any time.

Examples: %BAUD=300
%B 1200

BELL

Command: BELL=[n]

Purpose: To set the right-margin warning bell. A warning noise will be emitted from the terminal when the cursor reaches the column specified by "n". This is analogous to the ring given by a typewriter when the carriage nears the right margin. If the %BELL command is given without a column specification, the warning bell is disabled.

Example: %BELL=40
%BE 60

July 1988

BLANK

Command: BLANK

Purpose: To specify that the next input line should not be displayed on the screen. For example, this is useful for maintaining security when entering confidential information.

This command sets a switch which causes characters to be translated into blanks before they are entered into the conversation buffer. After the input line is entered, the switch is reset.

Pressing the FIX key will clear the input line and reset the switch, except when issued from MTS via a \$CONTROL command or the CONTROL subroutine while connected through UMnet.

Examples: %BLANK
%BL

CBEGIN, CEND

Command: CBEGIN
CEND

Purpose: To specify a portion of the conversation buffer to be printed by the %DUMP command. After the %CBEGIN command is executed, the first line currently displayed on the screen becomes the first line of the conversation buffer to be printed; after the %CEND command is executed, the last line currently displayed on the screen becomes the last line of the conversation buffer to be printed. The first and last lines may be adjusted by executing the %WF and %WB commands before the %CBEGIN and %CEND commands are executed.

If either the %CBEGIN or %CEND commands are omitted before the %DUMP command is used, the beginning and/or the end of the conversation buffer will be used.

To print the conversation buffer, the following sequence should be used:

- (1) Execute %CBEGIN and %CEND command (if desired).

July 1988

- (2) Issue the command

```
COPY *MSOURCE* FDname
```

where "FDname" is the file or device to which the conversation buffer will be written, e.g., *PRINT*.

- (3) Execute the %DUMP command.
- (4) Press the EOF key.

Example: %CBEGIN
 %CEND
 COPY *MSOURCE* *PRINT*
 %DUMP
 {end-of-file}

DCC

Command: DCC="x"

where "x" is any character.

Purpose: To redefine the device-command character. This character must be prefixed to all device commands. It is initially set to a percent sign (%). The quotes are required.

Examples: %DCC="*"
 %D "\$"

DEFINE

Command: DEFINE n=string

where "n" is an integer from 0 to 31.

Purpose: Define functions may be used to enter a single character or a sequence of characters into the input area. By default, the single characters that are inscribed on the numeric pad keys are assigned to Define Functions 0 through 25. These default assignments are detailed in the section "The Numeric Pad." Define Functions 26 through 31 do not have default assignments.

The %FUNCTION device command may be used to assign a define function to any key on the Ontel keyboard. By default, Define Functions 0 through 25 are assigned to the numeric pad keys as outlined in the section "The Numeric Pad" and are invoked when the %PAD=DEFINE device command is in effect. Define Functions 26 through 31 are not assigned to any keys. The use of the %FUNCTION command is described in the section "The FUNCTION Device Command."

For example, if the %FUNCTION command were used to assign Define Function 26 to the CTRL-8 key and the string "copy" was assigned to Define Function 26 as follows:

```
%DEFINE 26=copy
```

then pressing the CTRL-8 key would be exactly equivalent to typing the four keys c, o, p, and y in succession. If CAPS LOCK is on, "COPY" is entered; if INSERT mode is on, "copy" is inserted.

The storage available for all 32 replacement strings is limited to a total of 255 characters. If a DEFINE command that would cause the maximum to be exceeded is entered, it will be ignored, and no error message will be given.

Examples: %DEFINE 0=\$RUN PROGRAM SCARDS=
 %DEF 10=CANCEL
 %DEF 31=COPY *PRINT*

July 1988

DEF?

Command: DEF?
Purpose: To display the current definitions of the string assignment that were specified by the %DEFINE command.
Example: %DEF?

DUMP

| Command: DUMP
Purpose: To print the conversation buffer. The %DUMP command writes a portion of the conversation buffer to *MSOURCE*. The beginning and end of the portion written may be specified by the %CBEGIN and %CEND commands; if either of these commands is omitted, the beginning and/or the end of the conversation buffer will be used as the boundary.

To print the conversation buffer, the following sequence should be used:

- (1) Execute %CBEGIN and %CEND command (if desired).
- (2) Issue the command

COPY *MSOURCE* FDname

where "FDname" is the file or device to which the conversation buffer will be written, e.g., *PRINT*.

- (3) Execute the %DUMP command.
- (4) Press the EOF key.

Example: %CBEGIN
%CEND
COPY *MSOURCE* *PRINT*
%DUMP
{end-of-file}

July 1988

EMPTY

Command: EMPTY
Purpose: To empty (clear) the conversation buffer of all lines.
Examples: %EMPTY
 %E

ENTER

Command: ENTER
Purpose: To add the current contents of the input area to the conversation buffer without sending the line to the system. This is useful for saving a long input line so that it can be reentered later, perhaps after the user realizes that another input line must be entered first.

This command can only be used when entered via a program function key (by default, the %ENTER command is assigned to the PF8 key which is marked PROG or %ENTER on most Ontels). Because the processing of PF keys causes the line at the cursor to be moved to the input area before commands are processed, any line on the screen can be edited and then added to the conversation buffer. The line is added with a one-character prefix which is the current device command character. The input area is cleared after this command.

July 1988

FAST

Command: FAST

Purpose: To set the output mode of the terminal to FAST mode. In FAST mode, output lines are recorded in the conversation buffer, but are not written to the screen until an input line is requested by MTS. Only the last full screen is displayed and in the case of large amounts of output, some of it may be lost off the top of the conversation buffer.

Examples: %FAST
%F

FREEZE

Command: FREEZE=[n]

where "n" is the number of lines to be frozen.

Purpose: To cause the top "n" lines currently on the screen to remain on the display in spite of any vertical scrolling operations, further input, output, etc. The first characters of all frozen lines are displayed in inverse video, that is, the immediate area surrounding the character is darkened instead of the character. "n" must be greater than zero; if it is greater than the number of lines on the screen (not including the input area), only the lines on the screen will be frozen. The frozen lines can be edited and returned as input, windowed horizontally, etc., in the normal manner. If the frozen lines leave the conversation buffer (i.e., they have been forced out of the buffer by more recent lines), the freeze is deactivated. Note that this command does not alter any PF key definitions to reflect the smaller effective screen size. The user may wish to do this explicitly if using %FREEZE for an extended period.

Examples: %FREEZE=4
%FR 10

FUNCTION

| Command: FUNCTION n1 m1 [n2 m2 ...]

where "n" is a hexadecimal value identifying the key to be redefined and "m" is the function code specifying the function the key is to assume. Appendix B gives the value identifying each key, the default function code for each key, and the meaning of each function code.

Purpose: To redefine the function of a key. This command is most useful for assigning program functions and define functions to keys not normally used. For example, the unshifted CTRL-1 through CTRL-7 keys can be assigned the program functions PFA9 through PFA15, and the shifted CTRL-1 through CTRL-7 keys can be assigned the program functions PFB9 through PFB15. Likewise, the %FUNCTION command may be used to assign Define Functions 26 through 31 to other keys on the keyboard. See the section "The FUNCTION Device Command" for further details.

For example, program function PFA10 may be assigned to the CTRL-Q key by giving the command

 %FUNCTION 11 8A

In this case, 11 is the hexadecimal value identifying the CTRL-Q key, and 8A is the function code for PFA10 (i.e., the hexadecimal base value 80 for PFA0, plus 0A, the hexadecimal displacement for 10).

Any key listed in Table 1 of Appendix B can be assigned to any function code listed in Table 2. The command arguments are similar to those given in the above examples, except there is no hexadecimal offset to add to the function code as required for program functions and define functions. For example:

 %FUNCTION B3 1B

assigns the Delete Word function to the CTRL-3 key.

If "m" is omitted, the function of the key is restored to its original default value.

One must be cautious when using this command as a mistake in defining a key may cause a useful key to become unavailable until the user signs off or enters the %RESET device command. The command provides no error checking-

July 1988

HIST

Command: HIST=[n]

where "n" is any integer from 1 to 22.

Purpose: To specify the number of lines of conversation history which are redisplayed at the top of the screen when the 22nd line has been used in LINE mode. The default value is 7.

Examples: %HIST=5
%H 17

| LCI

| Command: LCI=[{ON|OFF}]

| Purpose: To disable automatic uppercase translation of characters transmitted to the system. The initial state of the terminal has translation disabled (LCI=ON). By issuing this command, lowercase alphabetic data can be entered. If ON or OFF is omitted, ON is assumed.

| Examples: %LCI=ON
| %LC

LINE

Command: LINE

Purpose: To set the output mode of the terminal to LINE mode. This command is equivalent to the %HIST=7 device command.

Examples: %LINE
%LI

PAD

Command: `PAD= [{NUMERIC|DEFINE|PFKEYS}]`

Purpose: To alter the function of the numeric-pad keys. If NUMERIC is specified (the default), the numeric pad keys will generate the characters shown on the keys. If DEFINE is specified, the numeric-pad keys will generate the character strings given via the %DEFINE command. If PFKEYS is specified, the numeric-pad keys will execute the program functions specified by the %PFAn and %PFBn commands. If no parameter is specified for the %PAD command, NUMERIC is assumed.

Examples: `%PAD=DEFINE`
`%P NUMERIC`

PAGE

Command: `PAGE`

Purpose: To set the output mode of the terminal to PAGE mode. In PAGE mode, output lines are recorded in the conversation buffer, but are not displayed until 19 new lines are received or until MTS requests an input line. The terminal also suspends output after receiving 19 contiguous lines of output. Transmission of output will resume when the user presses either the PAUSE or the ATTN key (the latter simultaneously generates an attention-interrupt).

Example: `%PAGE`

July 1988

PFA_n, PFB_n

Commands: PFA_n=string
PFB_n=string

where "n" is an integer from 0 to 15.

Purpose: To assign a string to a program function. There are two series of program functions, PFA0 through PFA15 and PFB0 through PFB15. Program functions PFA0-8 and PFB0-8 are initially assigned to the unshifted and shifted PF0 through PF8 keys as outlined in the section "Program Function Keys."

The initial definitions of all the program functions are device commands as follows:

<u>PFA and PFB</u>	<u>Value</u>
0	%WB 20
1	%WF 20
2	%WB
3	%WF
4	%WL
5	%WL 60
6	%WR 60
7	%BLANK
8	%ENTER
9	%WB 1
10	%WF 1
11	%WF
12	%THAW
13	%WL 1
14	%WR 1
15	%PAGE

The %PFA_n and %PFB_n commands may be used to reassign the program functions to other device commands or to other character strings such as MTS commands or program prompting replies.

When a key that is defined as a program function is pressed, the character string assigned to the program function is transmitted directly to the system instead of being placed in the input area (as in the case with define functions).

Program function PFA9-15 and PFB9-15 initially are not assigned to PF keys. However, these program functions, as well as those initially assigned to the PF keys, may

July 1988

be assigned to any key on the keyboard by issuing the FUNCTION device command. See the section "The FUNCTION Device Command" for more details.

Examples: %PFA6=%LINE
 %PFB2=\$CONTROL *PRINT* PRINT=PAGE
 %PFB8=CANCEL

PF?

Command: PF?
Purpose: To display the current string definitions of the program functions. They are displayed in such a way that reentering the lines will restore the definitions as displayed.
Example: %PF?

RESET

Command: RESET
Purpose: To restore to their default state all parameters and functions that can be changed by device commands except for the baud rate (%BAUD), the number of frozen lines (%FREEZE), and the "pause switch" (PAUSE). The CAPS LOCK and INSERT mode keys are reset to the off state.
Examples: %RESET

July 1988

ROLL

Command: ROLL

Purpose: To redefine the output mode of the terminal to ROLL mode. In ROLL mode each new input or output line added to the conversation buffer causes the screen to be erased and the latest screenful from the buffer to be displayed. The output appears to be rolling up the screen. This is the default mode.

Examples: %ROLL
%RO

TABI

Command: TABI=[{ON|OFF}] ["x"] [t,...]

where "n" is an integer value between 1 and 255. Up to 10 "n" values can be specified, in ascending order. If the ON or OFF parameter is omitted, the status will be changed to ON. If no tab stops are given, the previous tab stops will be used.

Purpose: To enable or disable input tabulation editing and to specify tab stops. Input tab editing is initially off with no tab stops set. Each character "x" given will become a tab character. The TAB key, by default, is always a tab character even if "x" is omitted.

Examples: %TABI=ON 10,20,30
%TAB ON 10 16 35
%T

THAW

Command: THAW

Purpose: To undo the effect of the %FREEZE command. All frozen lines will disappear from the screen. They will be replaced with lines from the conversation buffer above the first such line already on the screen.

Examples: %THAW
%TH

| UCI

| Command: UCI=[{ON|OFF}]

Purpose: To enable automatic uppercase translation of characters transmitted to the system. If ON or OFF is not given, ON will be assumed.

| Examples: %UCI=ON
| %UC

July 1988

WB, WF, WL, WR

Commands: WB= [n]
 WF= [n]
 WL= [n]
 WR= [n]

where "n" is any integer between 1 and 255.

Purpose: To allow the display window to be "scrolled" to any part of the conversation buffer. An attempt to move the window beyond the boundaries of the conversation buffer is not valid. The WR command causes the window to be displaced to the right by "n" characters and the WL command moves the window left. The WF command moves the window forward by "n" lines and the WB command moves the window backwards. If "n" is omitted, the extreme movement is assumed.

Examples: %WR=60
 %WF 20
 %WL
 %W

APPENDIX B: ONTEL KEYBOARD VALUES AND FUNCTION CODES

Table 1 gives the ASCII hexadecimal values for keys on the Ontel keyboard. This table is divided into four sections, one section for each part of the Ontel keyboard (i.e., the function pad, the standard keyboard, the control pad, and the numeric pad). For each Ontel key, two numbers are given in the form "n/m". The first number "n" is the hexadecimal value for the Ontel key. The second number "m" is the default function code that is executed by the terminal.

Table 2 gives the definitions of the function codes "m" that may be assigned to Ontel keys. The numbers from these two tables are used in the %FUNCTION device command which is given in the form

%FUNCTION n m

to redefine the function of Ontel keys (see the section "The FUNCTION Device Command" for details).

Table 1: Ontel Keyboard Values

The Function Pad Keys

<u>Key</u>	<u>Hex Value</u>	<u>Value with SHIFT</u>	<u>Value with CTRL</u>	<u>Value with CTRL and SHIFT</u>
PAUSE	E0/20	F0/20	E0/20	F0/20
Unlabeled key	E1/07	F1/07	E1/07	F1/07
CAPS LOCK	E2/19	F2/19	E2/19	F2/19
INSERT MODE	E3/18	F3/18	E3/18	F3/18
PF0	E4/80	F4/90	E4/80	F4/90
PF1	E5/81	F5/91	E5/81	F5/91
PF2	E6/82	F6/92	E6/82	F6/92
PF3	E7/83	F7/93	E7/83	F7/93
PF4	CE/84	DE/94	CE/84	DE/94
PF5	CA/85	DA/95	CA/85	DA/95
PF6	CC/86	DC/96	CC/86	DC/96
PF7	EE/87	FE/97	EE/87	FE/97
PF8	EF/88	FF/98	ED/88	Reset

July 1988

The Standard Keyboard Keys

<u>Key</u>	<u>Hex Value</u>	<u>Value with SHIFT</u>	<u>Value with CRTL</u>	<u>Value with CTRL and SHIFT</u>
1	31/03	21/05	B1/81	A1/07
2	32/03	22/04	B2/82	A2/07
3	33/03	23/04	B3/83	A3/07
4	34/03	24/04	B4/84	A4/07
5	35/03	25/04	B5/85	A5/07
6	36/03	26/04	B6/86	A6/07
7	37/03	27/04	B7/87	A7/07
8	38/03	28/04	B8/88	A8/07
9	39/03	29/04	B9/89	A9/07
0	30/03	30/03	B0/8A	B0/8A
-	2D/04	3D/04	AD/8B	BD/07
Up arrow ↑	5E/04	7E/04	1E/8C	1E/8C
Rev. slant \	5C/04	7C/DA	1C/06	1C/06
@	40/04	60/04	00/06	00/06
[5B/04	7B/04	1B/06	1B/06
Underscore	5F/DB	7F/1A	1F/06	1F/06
;	3B/04	2B/04	BB/07	AB/07
:	3A/04	2A/04	BA/07	AA/07
]	5D/04	7D/04	1D/06	1D/06
Comma	2C/04	3C/04	AC/07	BC/07
Period	2E/05	3E/04	AE/07	BE/07
/	2F/04	3F/05	AF/07	BF/07
Space Bar	20/00	20/00	A0/07	A0/07
A	61/02	41/01	01/06	01/06
B	62/02	42/01	02/06	02/06
C	63/02	43/01	03/1D	03/1D
D	64/02	44/01	04/06	04/06
E	65/02	45/01	05/21	05/21
F	66/02	46/01	06/06	06/06
G	67/02	47/01	07/06	07/06
H	68/02	48/01	08/1C	08/1C
I	69/02	49/01	09/14	09/14
J	6A/02	4A/01	0A/06	0A/06
K	6B/02	4B/01	0B/06	0B/06
L	6C/02	4C/01	0C/06	0C/06
M	6D/02	4D/01	0D/23	0D/23
N	6E/02	4E/01	0E/06	0E/06
O	6F/02	4F/01	0F/06	0F/06
P	70/02	50/01	10/29	10/29
Q	71/02	51/01	11/06	11/06
R	72/02	52/01	12/06	12/06
S	73/02	53/01	13/06	13/06
T	74/02	54/01	14/06	14/06
U	75/02	55/01	15/06	15/06
V	76/02	56/01	16/06	16/06
W	77/02	57/01	17/06	17/06

X	78/02	58/01	18/06	18/06
Y	79/02	59/01	19/06	19/06
Z	7A/02	5A/01	1A/06	1A/06

The Control Pad Keys

<u>Key</u>	<u>Hex Value</u>	<u>Value with SHIFT</u>	<u>Value with CRTL</u>	<u>Value with CTRL and SHIFT</u>
ATTN	87/21	97/21	87/21	97/21
EOF	88/1D	98/1D	88/1D	98/1D
FIX	89/2A	99/22	89/2A	99/22
HOME	84/13	94/28	84/13	94/28
DELETE	85/1A	95/1B	85/1A	95/1B
ERASE TO EOL	86/17	96/AB	86/17	96/AB
Up arrow ↑	81/10	91/AA	81/10	91/AA
Down arrow ↓	82/12	92/A9	82/12	92/A9
BACK TAB	83/15	93/15	83/15	93/15
Left arrow ←	8A/0F	9A/26	8A/0F	9A/26
Right arrow →	80/11	90/27	80/11	90/27
ZIP	8B/16	9B/16	8B/16	9B/16

The Numeric Pad Keys

<u>Key</u>	<u>Hex Value</u>	<u>Value with SHIFT</u>	<u>Value with CRTL</u>	<u>Value with CTRL and SHIFT</u>
0	C0/B0	D0/BA	C0/B0	D0/BA
1	C1/B1	D1/BB	C1/B1	D1/BB
2	C2/B2	D2/BC	C2/B2	D2/BC
3	C3/B3	D3/BD	C3/B3	D3/BD
4	C4/B4	D4/BE	C4/B4	D4/BE
5	C5/B5	D5/BF	C5/B5	D5/BF
6	C6/B6	D6/C0	C6/B6	D6/C0
7	C7/B7	D7/C1	C7/B7	D7/C1
8	C8/B8	D8/C2	C8/B8	D8/C2
9	C9/B9	D9/C3	C9/B9	D9/C3
Minus	CF/C6	DF/C9	CF/C6	DF/C9
Plus	CD/C5	DD/C8	CD/C5	DD/C8
Period	CB/C4	DB/C7	CB/C4	DB/C7

July 1988

Table 2: Ontel Function Codes

<u>Function</u>	<u>Value</u>
Blank	00
Uppercase alphabetic	01
Lowercase alphabetic	02
Numeric	03
Punctuation	04
End of Sentence	05
Control	06
Ignore	07
Cursor Left	0F
Cursor Up	10
Cursor Right	11
Cursor Down	12
Home	13
Tab	14
Back Tab	15
Zip	16
Erase EOL	17
Enter/Exit Insert Mode	18
Enter/Exit CAPS Lock Mode	19
Delete Character	1A
Delete Word	1B
Backspace and Blank	1C
Send EOF	1D
Move Cursor to Next Line	1E
Enter/Exit Pause Mode	20
Attention	21
Fix	22
Send line+CR	23
Double Left	26
Double Right	27
Move Cursor to Input Area	28
Literal Next	29
PFA _n (0≤n≤15)	80+hex(n)
PFB _n (0≤n≤15)	90+hex(n)
Default PF n (0≤n≤15)	A0+hex(n)
DEFINE n (0≤n≤15)	B0+hex(n)
DEFINE n (16≤n≤31)	C0+hex(n-15)
Default DEFINE n (0≤n≤15)	D0+hex(n)

Note: The function code of a PFA_n, PFB_n, or DEFINE n function is a sum of a hex code base value which specifies the type of the function, and a hex displacement value specifying the particular function "n".

Examples:

<u>Function</u>	<u>Base</u>	<u>+</u>	<u>Displacement</u>	<u>=></u>	<u>Code</u>
PFA8	80		08		88
PFB12	90		0C		9C
DEFINE 9	B0		09		B9
DEFINE 26	C0		0A		CA

July 1988

MERIT/UMNET

The Merit Computer Network is a packet-switching, data communications network which grew out of a joint venture among the University of Michigan, Michigan State University, and Wayne State University. Subsequently other state-supported universities in Michigan have joined this group. The primary purpose of Merit is the development and operation of a computer network which links together the computing resources of those universities.

Although all of the Merit universities have large, self-sufficient, combined time-sharing and batch-processing systems and numerous resources available on their own campus-wide networks, they find it advantageous to participate in the Merit network. The network allows students, faculty, and research staff at any Merit university to access and share the computing resources of the other members. A long-range Merit goal is to expand the network to other Michigan universities.

There are too many hosts on Merit, and those hosts change too frequently, to list here. For a list of Merit hosts, get to Merit's "Which Host?" prompt as usual (described in the section "Terminal Connection Procedures") and enter the HELP command.

The Merit Computer Network's hardware and software are designed, built, written, and maintained by the staff of Merit and the University of Michigan. The principal components of Merit are its Primary and Secondary Communications Processors, called PCPs and SCPs. PCPs are customized packet switching nodes which use DEC minicomputers and special interface equipment to provide the requisite networking functions. The SCPs are smaller nodes built from DEC microcomputers. Both PCPs and SCPs provide the same network terminal support, called Hermes.

The University of Michigan uses the Merit hardware and software for its UMnet campus network. UMnet is connected to Merit; in fact, since they use the same technology, the two networks appear to the user as one network. UMnet can be considered a subset of Merit, although the two are funded separately. The term "Merit/UMnet" is sometimes used to describe the part of Merit which is on the U-M campus.

The Merit network is also connected to two commercial networks, GTE Telenet and ADP Autonet. This allows access to the Merit hosts from any national or international location served by one of these networks or by the foreign public data networks connected to them.

TERMINAL CONNECTION PROCEDURESAccess Into Merit

The public microcomputers and terminals provided by the Computing Center are hardwired (directly connected by a high-speed line) into UMnet.

A user who has a terminal that is not hardwired into the network may use the dial-up telephone ports. The telephone number to use to make a connection depends on the data set and the data rate being used. The telephone numbers are given in the file MNET:HermesNos on MTS.

Before dialing the appropriate telephone number, the user must properly configure the terminal or microcomputer and modem. Use 7 data bits, EVEN parity, and 1 stop bit. Other parameters that may have to be set and their correct values are listed in the file MNET:Terminal.Set on MTS.

If a busy signal is heard, all dial-up lines at this data rate are busy. If there is no answer, there may be a problem with one of the modems in the dial-up sequence. In either case, try dialing a short while later. If there is a problem, call a Computing Center consultant at 764-HELP.

If there is a line available, the telephone is answered with a high-pitched carrier tone. After the connection is established, a terminal identification (answerback) procedure occurs. Some terminals provide terminal identification automatically. For terminals which are automatically identified, UMnet prints a message of the form:

```
%ttnn:answ
```

where "tt" is the port type, "nn" is the port number, and "answ" is the terminal identification (answerback) code. "%" is the prefix character used for all messages from UMnet. Familiarity with the above codes may be necessary if problems occur during the terminal session. If the terminal does not automatically identify itself, UMnet prompts the user to enter the terminal type by printing the message

```
%Terminal=
```

The user should respond by entering the terminal identification code for the type of terminal followed by a RETURN. The valid terminal types are listed in Appendix B. UMnet establishes the correct operating characteristics for the terminal including the carriage width and the delays for carriage-return, horizontal tab, and line-feed operations. If the user fails to enter a terminal type within 60 seconds, UMnet selects a terminal type which is based on the transmission speed:

July 1988

<u>Transmission Speed</u>	<u>Terminal Type</u>
110	TTY
300	TI700
1200	CRT
2400	CRT

The user can subsequently change or correct the terminal type by using the TERMINAL device command (see Appendix A). The available terminal types and their associated operating characteristics are given in Appendix B.

Following the terminal identification, UMnet prints at the terminal:

```
%Merit:Hermes (xxxx,ttnn:type:EDIT=UM+)
Which Host?
```

where¹

```
xxxx is the PCP or SCP port name,
ttnn is the port type and UMnet port number,
type is the terminal type (see Appendix B),
```

In response to "Which Host?", the user should type "UM" to establish a connection to UM-MTS or "UB" to establish a connection to UB-MTS. If MTS is operating, it prints the message

```
MTS Ann Arbor (ddyy-zzzzz)
```

where¹

```
ddyy is the MTS port name
zzzzz is the MTS job number associated with user's task.
```

At this point, MTS prompts with a pound sign "#" prefix character. The user should sign on to an MTS account with the SIGNON command; if no command is entered within 5 minutes, the terminal is automatically disconnected. After a valid SIGNON command is entered, the system prompts for the user's password by printing the message

```
Enter password.
?
```

After the password is accepted, MTS prints

¹This information is not normally needed by the user; however, if the user experiences difficulty during a terminal session, some of this information may be needed by Computing Center staff in order to investigate the problem.

July 1988

```
#mode,rate-group,class
#Last signon was at time, date
#User ccid signed on at time, date
```

The first line indicates the mode of access (Terminal or Batch), the rate-group for the job (Normal, Low, Deferred, or Minimum), and the classification of the signon ID (Univ/Gov't, Commercial, Exchange, Research, or Non-Univ). The remaining lines give the time and date of the last sign-on and the time and date of the current sign-on. After this information is printed, another "#" prompt is printed indicating that the user may begin the terminal session.

When MTS is unavailable, UMnet replies "UM Host is down" when "UM" is entered in response to the "Which Host?" prompt. When this occurs, the user can try accessing the host a short while later.

The following example illustrates the terminal-access procedure (input from the user is underlined):

```
%Enter terminal type: vt100

%Merit:Hermes (AE1C:LF14:DTC302:EDIT=UM+)

Which Host?um

    MTS Ann Arbor (AN21-00488)
    #sig wxyz
    #Enter user password.
    ?
    #Terminal,Normal,Univ/Gov'T
    #Last signon was at 12:32:43, Mon May 02/83
    #User WXYZ signed on at 13:41:15, Mon May 02/83
    #
```

| Access Through Another Network

The Merit Network is interfaced to two commercial networks, ADP Autonet and GTE Telenet. Both of these networks provide access to Merit hosts from hundreds of locations across the United States. Access from Canada and from several other international locations is also possible. Documentation for access from within the United States, overseas, and Canada is provided in Merit User's Memos 12, 19, and 20, respectively.

July 1988

ARRANGING AND PAYING FOR NETWORK USE

Individuals planning to use computing resources from any of the hosts on the network must have separate accounts for each computer they expect to access. There is no such thing as a Merit ID or password.

For University students, faculty, and staff who have an existing account at any Merit host, the procedure for obtaining a remote host account is:

- (1) fill out a Network Application form (available from each computing center);
- (2) obtain any necessary authorization from a department head, project director, instructor, or unit leader; and
- (3) take this Network Application form to the local computing center business office.

The local business office of each computing center obtains and issues the appropriate credentials for computer use at the remote computer(s) selected. This process typically takes about one day to complete.

In other words, all arrangements are made at the user's own university. All computing charges resulting from the use of these remote accounts are billed against the user's original, local-host account. Rebate requests are also initiated through one's local university.

Non-University account holders must make individual arrangements with the Computing Center of each host they plan to use.

The address and telephone number for the University of Michigan Computing Center Business Office is:

The University of Michigan Computing Center
535 W. William St.
Ann Arbor, MI 48109-4943
Attn: Business Office
(313) 764-8000

A list of addresses and telephone numbers of the computing center business offices of the other Merit members is given in the file MNET:Host.BusOfcs on MTS.

GETTING HELP WITH NETWORK USE

Help takes three basic forms: from documentation located on-line for immediate reference; from printed documentation available for purchase, free distribution, or reference; and from staff available at each host site and the Merit Central Office who can answer questions, in person or by telephone.

July 1988

Network Documentation

Merit provides documentation in several ways. First, the Merit staff maintains on-line help files on MTS hosts. These files contain information such as Telenet and Autonet access numbers, brief guides to network facilities, help in locating further assistance, and quick summaries of how to use various network services. For a directory of the available Merit help files, type

\$COPY MNET:INDEX

Second, Merit publishes Merit User's Memos that describe and explain the network and its use. A list of these memos is maintained on-line at the MTS hosts and is updated as new documents and new revisions of old documentation are issued. Users should type

\$COPY MNET:NETDOC

to see this file.

All network documentation is available from the Merit Central Office as well as from the user consultation services at each of the Merit hosts. Merit User's Memos are available free of charge.

Host Documentation

Merit publishes its User's Memo 11 to help users find the appropriate host and network documentation. This memo describes how to access the on-line help and obtain the printed documentation for each major Merit host.

At the University of Michigan, the information in help files is retrieved by using the "\$COPY filename" command, where "filename" is the name of an MTS file. Useful files to look at include *CCHOURS, *CCMEMOS, *CCPHONES, *CCPUBLICATIONS, *CONSULTINGHOURS, *RATES, and *WORKSTATIONS.

Consultation

For general consultation about Merit, its hosts, or the Merit-Telenet or Merit-Autonet links:

- (1) Contact the Merit Central Office at (313) 764-9430.
- (2) Users with an account at the University of Michigan may send a message via \$MESSAGE to Merit_Computer_Network containing their question or a description of their problem.

July 1988

| For a problem specific to UM, call the U-M Computing consultants at
| (313) 764-HELP.

| For a problem specific to another university or network, see the list
| of contact persons and telephone numbers in the file MNET:Help on MTS.

LINE-EDITING AND CONTROL FUNCTIONS

Special control characters are reserved by UMnet to provide line-editing facilities at the terminal. There are five main line-editing functions provided:

- (1) End-of-Line Character: RETURN is an end-of-line character that indicates that the line just typed is complete and should be passed on to the system. After the line is transmitted, the carriage is positioned at the beginning of the next line. Line editing for the deletion of characters (see (2) below) takes place before the line is transmitted over the network.
- (2) Delete-Previous Character: CTRL-H or BACKSPACE can be used as a backspace to delete the last character typed in the input line. To delete several characters, consecutive backspaces can be used. The character echoed at the terminal in response to the entering of a BACKSPACE or CTRL-H is controlled by the BACKSPECIALLECHO parameter. For most terminals, this parameter defaults OFF which means that a backspace is echoed and the carriage is moved one position to the left. See the description of the BACKSPECIALLECHO device command in Appendix A for further details.
- (3) Delete-Line Character: RUBOUT or DELETE can be used to delete an entire line; all characters entered in the line up to that point are deleted. The line-delete function has immediate effect; a "#" prefix is printed at the end of the deleted line and the carriage is positioned at the beginning of the next line.
- (4) Literal-Next Character: CTRL-P indicates that the next character should not be interpreted to have any special line-editing function, but should be processed literally as a data character. CTRL-P can be used before any character.
- (5) End-of-File Character: CTRL-C generates an end-of-file condition which is sent to MTS or to the executing user program. CTRL-C has immediate effect; a reverse slant "\" character is printed at the end of the current input line and the carriage is positioned at the beginning of the next line. If an input line is terminated by a CTRL-C instead of a RETURN, the input line is transmitted just as if it were an ordinary input line terminated with RETURN, but the next input request receives the end-of-file condition.

July 1988

The CTRL-E key can be used to generate an attention interrupt. The attention interrupt has immediate effect. During input, the line being entered (if any) is terminated by an exclamation point and the carriage is positioned at the beginning of the next line. All input lines still queued in the network are deleted. During output, the remainder of the output line is not printed and any output lines that are queued for printing are purged. The ECHO and EDIT parameters are reset to their default values (ECHO=ON and EDIT=ON). If INLEN is less than 240, it is reset to 240; if it is greater than 240, it remains unchanged.

Output printing at the terminal can be suspended by entering CTRL-T; in this case, all output including the printer and paper-tape punch is suspended. Output can be resumed by entering CTRL-R.

The EDIT device command can be used to design and control the keyboard editing characteristics of the terminal including character-delete (CTRL-H), line-delete (RUBOUT), and attention interrupt (CTRL-E). See the description of the EDIT device command in Appendix A.

Following is the default keyboard editing table for UMnet.

<u>Function</u>	<u>Character</u>
End-of-line	RETURN or CTRL-M
Character-delete	BACKSPACE or CTRL-H
Line-delete	DELETE, RUBOUT, CTRL-X, or CTRL-N
Literal-next	CTRL-P
End-of-file	CTRL-C
Attention	CTRL-E or BREAK
Halt output	CTRL-T
Pause	CTRL-S
Resume output	CTRL-Q or CTRL-R
Echo	ESCAPE
End-of-tape	CTRL-D
Ignore	NUL, CTRL-F, or CTRL-J
Repeat	CTRL-W

DEVICE COMMANDS

The terminal user may issue commands directly to UMnet. Device commands deal with device-oriented functions as opposed to system operations. They enable the user to control the behavior of the terminal. UMnet inspects each input line entered from the terminal for a device command.

There are two levels of device-command processing. A resident command-language interpreter within UMnet handles commands for line-editing, while functions such as tabulation, output carriage control, and record-lengths are handled by commands to the MTS Device-Support

July 1988

Routines (DSR) for UMnet. Since UMnet passes commands along from one level to the next, users need not be concerned about where the commands are processed.

Device commands can be entered on any new input line. Each device command must be preceded by the "device-command character" which by default is the percent sign "%". The device-command character can be changed by the DCC device command. A line that does not begin with a single device-command character is passed directly to MTS as a data line.

Some of the more commonly used device commands are described briefly below. All device commands are listed and described in more detail in Appendix A.

BLANK - This command causes the echoing of the of next input line to be suppressed so that confidential information may be typed. It is entered as

%BLANK

OUTLEN - This command sets the length of output lines. Any output line exceeding the value of OUTLEN is truncated at that value. Each type of terminal has its own default value for OUTLEN which is determined by its carriage width. If the user sets OUTLEN to be greater than the value set for the carriage width, output records longer than WIDTH will fold onto subsequent output line(s). The command is entered as

%OUTLEN=n

RESET - All parameters that have been changed by device commands are reset to their default values by this command. The command is entered as

%RESET

TAB - This command enables the user to control tabbing and tab settings while entering input. When this command is entered to turn tab editing on, the user may specify the tab stops and the tab character or use the default settings. While entering input, if the current position is to the left of the next tab stop, entering the tab character causes blanks to be inserted up to the next tab stop. If the last tab stop has already been passed, one blank is inserted. The tab command is entered as

TAB[={ON|OFF}] ["x"] [t,...]

where "t,..." are the tab positions, e.g.,

%TAB=ON 10,20,30,40

- | On most terminals, the tab character is CONTROL-I.
- UCI - This command can be used to control the conversion of lowercase input to uppercase. The default is OFF (uppercase-conversion disabled). The command is entered as
- %UCI={ON|OFF}
- UCO - This command is similar to the UCI device command except that it controls the conversion of lowercase output to uppercase. The default is OFF (uppercase-conversion disabled). The command is entered as
- %UCO={ON|OFF}
- WIDTH - This command specifies the carriage width (or screen width) to be used for output printing. The default value depends on the terminal type; for the DTC 302, the value of WIDTH is 130. If the length of the output exceeds the value of the WIDTH parameter, the output will fold, that is, continue to printed on subsequent lines. The output will fold at the value set for WIDTH. The command is entered as
- %WIDTH=n
- where "n" is the carriage width to be used, e.g., WIDTH=60.

USING NETWORK BATCH

| Network batch service allows users to submit batch jobs (either via the card reader or *BATCH*) for execution at the UM, WU, or WS hosts. The output from these jobs can be routed to a printer or card punch located at the UM or WU hosts. Independent of a network execution job, MTS users can interactively route print jobs to Merit hosts using the \$CONTROL *PRINT* command. In addition, network batch can be used for copying files between Merit hosts. For large files, batch file transfer is usually faster, can be more reliable, and is more economical. Batch file copying is explained later in this section.

Network Execution Jobs

To route a batch job submitted through a local card reader or copied to *BATCH* to another host for execution, the file that contains the job set-up must have as its first record the command

```
#NET EX=host1(options) [PROUTE=host2(options)]
      [CROUTE=host2(options)] [ROUTE=host2(options)]
```


July 1988

| where "host1" is the host (either WS or WU) that will execute the batch
| job, and "host2" is the host (UM or WU) that will receive the printed or
punched output from the job. The MTS line-continuation character (-)
may be used to extend the #NET command to another line. "options" are
the host-dependent options which are valid for the receiving host (see
below). PROUTE is used to route printed output, CROUTE is used to route
punched output, and ROUTE is used to route both printed and punched
output.

For example, the following command may be used to route a batch job
for execution on MTS at Wayne State with all output to be returned to
the University of Michigan:

```
#NET EX=WU ROUTE=UM(ID=umid,PW=umpw,ROUTE=UNYN,PRINTER=PAGE)
```

If no output routing information is given, all output will be returned
to the UM local station from which the job was submitted (or to CNTR for
jobs submitted via *BATCH*). Routing information is required for host
UM only when the user wants to change the default settings of any of the
host-dependent options. In the above example, the ROUTE option was
explicitly needed since the values specified for the ROUTE and PRINTER
options (within the parentheses) are not the default settings. Punched
output routed to stations without card punches will be rerouted to CNTR.

| Since the only host that currently requires host-dependent options
| following "host1" is the WS host, for the UM and WU hosts this option is
simply written

```
EX=host1
```

After the #NET record will follow the actual batch job to be executed
at the execution host which was designated on the #NET record by
"EX=host1". This batch job will consist of commands to sign or log on
to the execution host, issue a password, and perform the tasks included
in the job. All commands intended for the execution host must conform
to the standards for that host.

The file containing the #NET record and batch job set-up can be
copied to *BATCH* on MTS. MTS will issue a network receipt number which
should be used to retrieve the output at the host designated on the #NET
record.

When submitting jobs through a local card reader, the receipt number
(S-8) card is used to pick up a confirmation printout that will be
printed at the station where the job was submitted. If the #NET command
contains errors, the confirmation printout will have an error comment.
If the #NET command is not in error, this printout will show the network
receipt number assigned to the job to be executed. This number is used
to pick up the output from the job.

"\$CONTINUE WITH" and "\$ENDFILE" lines in batch jobs submitted through
local card readers receive no special treatment and will be passed
unchanged to the remote host. The treatment of "\$CONTINUE WITH" and

July 1988

"\$ENDFILE" lines copied to *BATCH*, *PRINT*, or *PUNCH* will depend on the setting of the ENDFILE and IC options and the ENDFILE and IC I/O modifiers; i.e., if these options or modifiers are enabled, the "\$CONTINUE WITH" and "\$ENDFILE" lines will be effective.

The PRINT and COPIES options of the \$CONTROL and \$SET commands are ignored for network batch jobs.

Routing Output from MTS Batch Jobs

Another way to route output from an MTS batch job that executes locally to another Merit host is to append the ROUTE, PROUTE, and/or CROUTE options to the \$SIGNON command for the batch job. For example, the command

```
$SIGNON umid PROUTE=WU (ID=wuid,PW=wupw,PRINTER=LINE)
```

will route the printed output from a job executed at host UM to host WU for printing. When the batch job is copied to *BATCH* (or submitted via a card reader), MTS will issue a network receipt number which should be used to retrieve the output at Wayne State University. Any other output, e.g., punched output, will remain at the UM host unless it is routed elsewhere by the CROUTE option.

Network Print Jobs

Users may also route a file from MTS to a Merit host for printing by using the \$CONTROL *PRINT* command during a terminal session. For example, the commands

```
| $CONTROL *PRINT* PROUTE=WU (ID=wuid,PW=wupw,PRINTER=LINE)  
| $COPY printfile *PRINT*  
| $RELEASE *PRINT*
```

```
| can be used to route "printfile" to a line printer at WSU. When the  
| print job is released, a network receipt number will be issued which  
| should be used to retrieve the output at Wayne State University.
```

Routing Output to Xerox Page Printers

Often when routing output to a remote host, users will want the output to print on a Xerox page printer. This can be done for MTS hosts by specifying PRINTER=PAGE as a host-dependent option. At UM, this will produce landscape, two-sided output in the standard font on the page

July 1988

printer. At WSU, this will produce landscape, two-sided, three-hole punched output in the standard font. At WSU, three other types of page-printer output (all in landscape mode) can be selected as host-dependent options (see "Host-Dependent Options" below). To produce output on the page printer that is in a different format (e.g., portrait) or in a nonstandard font, users must copy their files to the remote MTS host and then run the *PAGEPR program. Files can be copied to the remote host interactively (.COPY) or via batch. Interactive and batch file copying are described below.

Batch File Copying

Merit provides a utility program, MNET:BLOCK, for copying files in batch mode between Merit's two MTS hosts (UM and WU). This utility blocks files with lines of any length into lines of a uniform length equal to the network's packet size, and deblocks the files to their original condition at the other end of the network connection.

To use MNET:BLOCK, construct a file that contains the #NET record that will route the job to the specified host and follow it with an MTS batch job. The only difference between this job and one used for regular batch file copying is that a special MNET:BLOCK "/COPY" statement is used instead of the \$COPY command. /COPY uses the standard \$COPY command syntax including line ranges and FDname modifiers.

When the file of batch commands is completed, instead of \$COPYing it to *BATCH*, run the program MNET:BLOCK assigning the UM file to SCARDS and *BATCH* to SPUNCH. For example:

```
$RUN MNET:BLOCK SCARDS=batchfile SPUNCH=*BATCH*
```

where "batchfile" contains the following lines:

```
#NET XROUTE=WU PROUTE=UM(host-dependent options)
$SIGNON wuid T=nn
wupw
$CREATE wufile
/COPY umfile(*F) wufile@I
$SIGNOFF
```

This job blocks an existing file at UM into 240-character packets, transfers it across the network to Wayne State, and copies it into the file "wufile" (which is \$CREATED for the purpose in this job but could have been pre-existing) in its original form. The T (TIME) option on the \$SIGNON card is optional, but if "umfile" is very large it may be needed to keep the job from exceeding the default global time limit.

The print produced by this job will indicate whether the job executed successfully or not. It will not contain a listing of the transferred file unless this is specifically requested with an additional command following the /COPY command (such as "\$COPY wufile" or "\$LIST wufile").

July 1988

This same job structure can be used to copy files in the other direction (from WU to UM). The host identifiers on the #NET command must be changed and a UM user ID and password substituted for the WU user ID and password (and of course MNET:BLOCK must be run at Wayne State).

As indicated above, the job need not end after the /COPY command; any other MTS commands may be issued, including commands to \$RUN the copied file or use it as input to the *PAGEPR program which will produce a copy of the file on the Xerox page printer.

Host-Dependent Options

The following is a list of the host-dependent options that may appear following the host name (UM, WS, or WU) on the #NET command, the \$SIGNON command, or the \$CONTROL *PRINT* command. These options must be enclosed in parentheses and separated by commas. Each host has its own set of options, some of which are required.

Wayne State University MTS Execute Jobs (EX=WU)

No execute host-dependent options are allowed; the necessary information is specified on the \$SIGNON command which immediately follows the #NET command.

Wayne State University Print and Punch Jobs

ID=wsid	Wayne State user id (required).
PW=wspw	Wayne State password (required).
PROUTE=station	Local routing information for print output; see WSU MTS Volume 1 (optional; defaults to CNTR, if omitted).
CROUTE=station	Local routing information for punch output; see WSU Volume 1 (optional; defaults to CNTR, if omitted).
ROUTE=station	Local routing information for all output, print and punch.
PRINTER={PAGE LINE ANY}	Type of printer (optional; defaults to PAGE, if omitted). If PAGE is selected, by default landscape, two-sided, three-hole punched output is produced on the Xerox page printer. Four types of page-printer output (all landscape) can be specified as host-dependent options for WU:

July 1988

ROUTE=CNTR (punched, two-sided) (default)
 ROUTE=PONE (punched, one-sided)
 ROUTE=UTWO (unpunched, two-sided)
 ROUTE=UONE (unpunched, one-sided)

PRINT={ANY|TN|PN|UC|LC|MC}
 Print train, see WSU MTS Volume 1 (optional; defaults to ANY, if omitted).

COPIES=n Number of copies (optional; defaults to 1, if omitted).

DELIVERY=d where "d" is a Wayne State delivery code used for delivering output on the WSU campus, see WSU MTS Volume 1 (optional, defaults to the CSC distribution window).

Wayne State University Non-MTS Execute Jobs (EX=WS(options))

SYS=MVS Specifies the MVS Operating System on the Wayne State University administrative Amdahl 470V/6 computer. Printed output from the job may be routed to the Merit hosts UM or WU by specifying PROUTE=host(options).

University of Michigan Execute Jobs (EX=UM)

No execute host-dependent options are allowed; the necessary information is specified on the \$SIGNON command which immediately follows the #NET command.

University of Michigan Print and Punch Jobs

ID=umid University of Michigan user id (required).

PW=umpw University of Michigan password (required).

PROUTE=station Local routing information for print output (optional; defaults to local input station, or to CNTR for *BATCH*, if omitted).

CROUTE=station Local routing information for punch output (optional; defaults to local input station, or to CNTR for *BATCH* or if input station lacks card-punching equipment).

ROUTE=station Local routing information for all output, print and punch.

PRINTER={PAGE|LINE|ANY}
 Type of printer (optional; defaults to ANY, if omitted). If PAGE is selected, by default lands-

July 1988

cape, two-sided output is produced on the Xerox page printer.

PRINT={ANY|PN|TN|UC|LC|MC}
 Print train (optional; defaults to ANY, if omitted).

COPIES=n Number of copies (optional; defaults to 1, if omitted).

DELIVERY={station|NONE}
 "station" specifies the UM campus batch station to which output will be delivered by courier service; see MTS Volume 1.

DATA TRANSMISSION

Data lines entered by the user are transmitted by the workstation to UMnet where they are examined for errors. There are three types of errors that can occur: parity errors, overrun errors, and buffer overflow errors.

Data transmission between the terminal and UMnet takes place in a coded format that is known as the American Standard Code for Information Interchange (ASCII). This is a seven-bit character code with an eighth bit generated for the parity convention selected. The parity bit is used as a check bit for message reliability. Whenever a character is sent or received, the parity bit is set to either zero or one, depending on the type of parity and the number of 1-bits contained in the character representation. Even parity means that together with the parity bit the total number of 1-bits is an even number; odd parity means there is an odd number of 1-bits for each character. For example, the letter A is represented in ASCII as 1000001. With even parity, the parity bit is set to 0 so that the final ASCII representation is 01000001; with odd parity the parity bit is set to 1 and the final ASCII representation is 11000001.

UMnet normally checks the parity of each character as it is received to ensure that the character was transmitted correctly. If the character was transmitted incorrectly, i.e., some of the bits in the character were altered, a parity error may be detected. If this case, the message

%Parity error.

will be printed on the terminal; the line must then be reentered by the user. Parity checking can be disabled by setting the PARITY option to OFF (see the description of the PARITY device command in Appendix A).

July 1988

Overrun errors occur when an input character has not been processed by the system before the next character is received. If overrun occurs, the message

```
%Data lost.
```

is printed on the terminal, and the line must be reentered.

Buffer overflow errors occur when no storage is available in UMnet to store an input character. Buffer storage is assigned to a terminal in small blocks from a large pool available to all terminals. Buffer overflow can occur if there is a particularly heavy load and the user has entered more input than may be processed for one terminal or if a few terminals are using large amounts of storage. If buffer overflow occurs, the entire record is discarded and when buffer space becomes available, the message

```
%Buffer overflow.
```

is printed on the terminal. The input line can then be reentered. Buffer overflow also occurs if the user attempts to enter an input line that is longer than the maximum allowable input record length.

Under unusual circumstances, every character entered may cause an error message to be printed on the terminal. This may be due to a terminal malfunction or to a buffer overflow condition in UMnet itself. If buffer overflow is suspected, the user should press the BREAK button which will cause UMnet to reset certain error flags and reclaim buffer space allocated to the terminal. If the trouble persists, the user should contact the Computing Center.

If the character entered is not discarded due to an error, it is examined for one of the special line-editing characters (see the section "Line-Editing and Control Functions" above). If the character is not one of these special characters, it is stored in a buffer as part of a record to be passed on to MTS.

Normally UMnet buffers characters received from a terminal and sends the record to MTS when a end-of-line character is received (e.g., RETURN). UMnet will also send a record to MTS when the number of characters entered exceeds the value of the INLEN parameter. Users with input record-lengths greater than 240 characters should read the description of the INLEN device command in Appendix A. Until the end-of-line character is received or the value of INLEN is exceeded, the partial record can be edited, i.e., backspaced over or deleted.

After performing any line-editing and either receiving an end-of-line character or reaching the value of INLEN, UMnet translates the record from ASCII to EBCDIC and sends it to MTS. Translation to EBCDIC can be disabled by setting the BINARY option to ON (see the description of the BINARY device command in Appendix A).

MTS FDNAME MODIFIERS

MTS FDname modifiers can be used to affect input and output operations (see Appendix A to "Files and Devices" in MTS Volume 1, The Michigan Terminal System). Modifiers are appended to file or device names by specifying the FDname followed immediately by an at-sign (@) and the modifier. Modifiers can also be specified in the call to MTS I/O subroutines (READ, WRITE, etc.).

The modifiers that are special to UMnet are described below.

BIN - The binary modifiers @BIN and @-BIN (the default) control the translation of characters to or from the ASCII code used by terminals to the EBCDIC code used by MTS. In addition to disabling the translation, the @BIN modifier also disables carriage-control processing, prefixing, command recognition, tabulation, hexadecimal conversion, line-number peeling, output or input line truncation, output carriage-length processing, and any other operations which affect the input or output data. The use of @BIN does not affect the recognition or translation of device commands.

CC - The logical carriage-control modifiers @CC (the default) and @-CC affect the application of output carriage-control processing. A carriage-control character is the first character of an output line, not including the prefix character, if any. The valid carriage-control characters are:

blank	single space (skip 0 lines)
0	double space (skip 1 line)
-	triple space (skip 2 lines)
+	single space (skip 0 lines)
&	no space after printing
9	single space
1	triple space
2	triple space
4	triple space
6	triple space
8	triple space

Valid carriage-control characters are deleted from the output line unless the CC=OFF device command has been given (see the CC device command description in Appendix A). A complete description of the carriage-control function is given in Appendix H to the section "Files and Devices" in MTS Volume 1, The Michigan Terminal System.

SP - The special modifiers @SP and @-SP (the default) control the processing of MTS prefix characters on both input and output operations for UMnet. The @SP modifier applied to an I/O operation prohibits the use of prefixing on that operation,

July 1988

i.e., no prefix characters are printed at the beginning of input or output lines.

MODES OF OPERATION

The MODE device command allows users to simultaneously set a group of associated terminal characteristics. Four different modes are available, each representing a different combination of options. To select a particular mode, issue the command

```
MODE=mode
```

where "mode" is FDX, HDX, TAPE, or BINARY. Each mode is described below.

FDX (Full-duplex) Mode

Full-duplex is the default mode. UMnet echoes input character typed at the terminal back to the terminal providing a visual indication that the character was properly received. In full-duplex mode, input to UMnet may be entered while output is in progress since it is UMnet which controls the echoing of input at the terminal. The line-editing control characters produce the effect described in the section "Line-Editing and Control Functions"; all other characters are echoed exactly as received. The terminal should be set for full-duplex operation when FDX mode is used.

The following parameters are set to the values indicated:

```
BINARY=OFF DUPLEX=ON ECHO=ON EDIT=ON READER=OFF
```

HDX (Half-duplex) Mode

In half-duplex mode, the terminal performs the echoing process and UMnet echoes only the special sequences for line-editing control characters. When the terminal is performing the echoing, it is typing directly on the printer in response to input being typed on the keyboard. Therefore in HDX mode, it is not permissible to type while output is in progress since input echoing would interrupt the printing of the output. The terminal itself should be set for half-duplex operation when HDX mode is used.

The following parameters are set when HDX mode is used:

```
BINARY=OFF DUPLEX=OFF ECHO=ON EDIT=ON READER=OFF
```

TAPE Mode

TAPE mode is intended for paper-tape or magnetic-tape cassette input in symbolic format and is useful for the reading of tapes prepared either off-line or by a terminal connected to MTS. Tapes can be read reliably only from terminals equipped with a tape reader which responds to the control characters CTRL-Q (X-ON) and CTRL-S (X-OFF) (used to start and stop the tape reader, respectively). The terminal must be capable of full-duplex operation. To provide input flow control and prevent loss of data, UMnet sends a CTRL-S to stop terminal transmission whenever UMnet or MTS is unable to receive further input. When UMnet or MTS again is ready to receive input, a CTRL-Q is sent and transmission resumes. Even if a CTRL-S has not been sent, a CTRL-Q will always be sent after every record received (when UMnet is ready for another record). In most cases, the rate at which UMnet and MTS can receive input is sufficiently high to keep file transfer continuous. Equipment which does not respond to CTRL-S and CTRL-Q can still be used with UMnet, but the user should be warned that a fair probability exists that data will be lost if the total number of input characters exceeds 250. In such cases, the ECHO parameter should be set to OFF.

The following parameters are set when TAPE mode is used:

```
BINARY=OFF ECHO=ON EDIT=OFF READER=ON
```

Note that the EDIT option defaults OFF in TAPE mode. This means that CTRL-H (backspace), CTRL-N (shift-out), CTRL-O (shift-in), and CTRL-SHIFT-K (escape) are transmitted as data characters and do not perform their regular editing functions. RUBOUT (line-delete), CTRL-I (horizontal tab), CTRL-E (attention interrupt) are ignored. Note also that the PARITY control option is not affected by any of the mode operations. Therefore, if the tape is not punched with even parity, the PARITY=OFF option should also be used.

Once the command %MODE=TAPE has been issued, the \$COPY command should be used to copy a tape to a file or device. MTS prefixing should be disabled either by the @SP FDname modifier, the device command PFX=OFF, or the MTS command \$SET PFX=OFF. For example,

```
$COPY *SOURCE*@SP FDname
```

will copy the tape to "FDname", the name of a file or device. This command can also be given after entering TAPE mode.

When transmission is concluded, a CTRL-D should be entered either from the keyboard or from the tape itself. CTRL-D transmits an end-of-file to MTS, stops the reader, turns EDIT on, and returns the terminal to either FDX or HDX mode, whichever was in effect prior to issuing the MODE=TAPE command.

July 1988

BINARY Mode

BINARY mode is intended for arbitrarily coded tape input and is useful for the input of binary tapes punched by peripheral equipment. All of the information given under TAPE mode, concerning control of the tape reader, also applies to BINARY mode. In this mode, the normal ASCII-to-EBCDIC translation is not performed. All input is accepted and transmitted to MTS with the parity bit (channel 8) being placed in the high-order bit position in each byte in the central system. Characters are accumulated until the record size (default 240 characters) has been reached. The record is then automatically ended and transmitted to MTS. The INLEN device command can be used to change the default record size.

The following parameters are set when BINARY mode is in effect:

BINARY=ON ECHO=OFF READER=ON

To copy binary data to a file or device, the following MTS command can be used:

\$COPY *SOURCE*@BIN FDname

The BREAK key is used to exit from BINARY mode and return to either FDX or HDX mode, depending on which of these modes preceded BINARY mode. The BREAK will not cause an attention interrupt in this case, but instead will send an end-of-file, end-of-line, and turn BINARY and READER off.

The four modes available are mutually exclusive. The table below summarizes the options set with each mode.

<u>Option</u>	<u>FDX</u>	<u>HDX</u>	<u>TAPE</u>	<u>BINARY</u>
BINARY	OFF	OFF	OFF	ON
DUPLEX	ON	OFF		
ECHO	ON	ON	ON	OFF
EDIT	ON	ON	OFF	
READER	OFF	OFF	ON	ON
SIZE	128	128	128	255

| APPENDIX A: MERIT/UMNET DEVICE COMMANDS

Any input line beginning with a device-command character (DCC) is treated as a device command; all other input lines are passed on to MTS as ordinary input. By default, the DCC is a percent sign (%). It may be altered to any other character except a lowercase letter using the DCC command. The actual processing of device commands is done by UMnet, by MTS, or both. The table below indicates where the processing is done. In general, the user need not be concerned about where the processing occurs.

If the user wishes to enter an ordinary (non-device command) input line which begins with the current DCC, one extra DCC must be typed at the beginning of the line. UMnet will remove the first DCC and transmit the remainder of the input line to MTS as ordinary input.

Commands to UMnet may be issued at any time, whether or not MTS is expecting input. Commands that are processed only by UMnet are acted on as soon as they are issued. However, commands that are performed by MTS or by UMnet and MTS in tandem are queued until MTS is ready for input.

The DCC should not be used in commands issued through the MTS \$CONTROL command or the CONTROL (or CNTRL) subroutine, e.g.,

```

From a terminal:  %UCI=OFF

From MTS:        $CONTROL *MSINK* UCI=OFF

From FORTRAN:    INTEGER*2 LEN/7/
                  CALL CNTRL('UCI=OFF',LEN,6)
    
```

In the descriptions of the device commands that follow, the following syntax conventions are used:

- {..|..} braces indicate a choice to be made from the enclosed alternative forms (each separated by a vertical bar).
- [..] brackets indicate optional parameters.
- ... ellipses indicate parameters that may be repeated.
- ___ underlining indicates minimum abbreviations.

July 1988

| Summary of Merit/UMnet Device Commands

BACKSPECIALECHO={ON|OFF}
BINARY={OFF|ON}
BLANK [=n]
BROADCAST={ON|OFF}
CC={ON|OFF|MOD}
CLOCK=n
CONNECTIONS
CRDELAY=n
DCC={x|"x"}
DEQUEUE [=n]
DONT [= {ON|OFF}]
DUPLEX={ON|OFF}
ECHO={ON|OFF}
EDIT={?|ON|OFF| [table] [key=function]}
FLIP [{+|-}]
FORMFEED [= {ON|OFF}]
GRAB
HELLO
HEX [= {ON|OFF}] ["x"]
INLEN=n
INTERLOCK={OFF|ON}
JOB
LCI [= {ON|OFF}]
LINES=n
MIX={ON|OFF}
MODE={FDX|HDX|TAPE|BINARY}
NPC={ON|OFF| {x|"x"} }
OPERATOR messagetext
OUTLEN=n
PAGEWAIT={ON|OFF|n}
PARITY={OFF|ON}
PFX={ON|OFF|MOD}
QUIT
READER={OFF|ON}
RESET
SCROLL={ON|OFF}
SENSE
STATUS
TABI [= {ON|OFF}] ["x"] [t,...]
TABSPECIALECHO={ON|OFF}
TBDELAY=n
TERMINAL=termtype
TEST [length]
UCI={ON|OFF}
UCO={ON|OFF}
WAIT
WIDTH={n|ON|OFF}
?
n±n

Where Device Commands are Performed

| Device Commands Performed Entirely by Merit/UMnet

BACKSPECIALECHO	FLIP	SCROLL
BINARY	GRAB	SENSE
BLANK	HELLO	STATUS
BROADCAST	INTERLOCK	TABSPECIALECHO
CLOCK	LINES	TBDELAY
CONNECTIONS	MIX	TEST
CRDELAY	MODE	UCB
DCC	NPC	UCO
DEQUEUE	OPERATOR	WIDTH
DISPLAY	PAGEWAIT	?
DUPLEX	PARITY	n±n
ECHO	QUIT	
EDIT	READER	

| Device Commands Performed by Merit/UMnet and MTS

INLEN	RESET
OUTLEN	TERMINAL

Device Commands Performed by MTS Only

CC	JOB	TABI
DONT	LCI	UCI
FORMFEED	PFX	WAIT
HEX	SNS	

July 1988

Functional Classification of Device Commands

Commands Affecting the Behavior of the Terminal

Physical Behavior

CLOCK
CRDELAY
TBDELAY
TERMINAL

Terminal I/O

Input	Output	Echo
BINARY	CC	BACKSPECIALECHO
DEQUEUE	FORMFEED	BLANK
EDIT	LINES	DUPLEX
HEX	NPC	ECHO
INLEN	OUTLEN	INTERLOCK
LCI	PAGEWAIT	MODE
PARITY	PFX	NPC
TABI	SCROLL	TABSPECIALECHO
TERMINAL	TERMINAL	TERMINAL
UCI	UCO	WIDTH
WIDTH	WIDTH	

Commands that Control Peripheral I/O Devices

BINARY	MODE	READER
--------	------	--------

Commands that Control Multiple Concurrent Connections

CONNECTIONS	GRAB	QUIT
FLIP	MIX	

Commands that Display Information

CONNECTIONS	JOB	UCB
DISPLAY	SENSE	?
EDIT=?	SNS	
HELLO	STATUS	

Commands that Control Device Commands

DCC	RESET	TERMINAL
-----	-------	----------

Miscellaneous Commands

BROADCAST	OPERATOR	WAIT
DON'T	TEST	n+n

July 1988

BACKSPECIALECHO

Purpose: To print an underscore () when BACKSPACE is entered on full-duplex terminals without a physical backspacing capability.

Prototype: %BACKSPECIALECHO={ON|OFF}

Description: The BACKSPECIALECHO command controls the character echoed by the system in response to the entering of BACKSPACE or CTRL-H on full-duplex terminals. With BACKSPECIALECHO set to ON, an underscore () (left-arrow on some terminals) is echoed when BACKSPACE is entered. With BACKSPECIALECHO set to OFF, a backspace is echoed in response to BACKSPACE or CTRL-H and the print head (or CRT cursor) is moved one position to the left.

| BACKSPECIALECHO defaults ON for ASCII terminals that are
| recognized as CRTs and OFF for all other terminals. See
the table of terminal types in the file *TERMTYPES or in
Appendix B.

Example: %BACKSPECIALECHO=ON

Whenever the user enters BACKSPACE or CTRL-H, an underscore will be printed at the terminal.

Comment: See also the description of the TABSPECIALECHO device command.

July 1988

BINARY

Purpose: To disable character translation on input.

Prototype: %BINARY={ON|OFF}

Description: When input is sent to MTS with BINARY set to ON, the normal translation of input characters from ASCII (the code used by terminals) into EBCDIC (the code used by MTS) is disabled. The data is passed as 8-bit bytes with no tab editing, hex processing, uppercase conversion, or line-number peeling performed. BINARY defaults OFF.

Any setting of the EDIT switch is overridden with BINARY set to ON, and only the BREAK key has special significance; all other input is treated as data. Pressing the BREAK key ends the current input record, sends an end-of-file to MTS, and resets BINARY to OFF and ECHO to ON. This use of BREAK does not send an attention interrupt to MTS.

Another way of setting BINARY to OFF is to issue the MTS \$CONTROL command or to call the CONTROL subroutine from a program. For example:

```
$CONTROL *MSOURCE* BINARY=OFF
```

```
or  INTEGER*2 LEN/10/
     CALL CNTRL('BINARY=OFF',LEN,6)
```

The maximum length of binary input records is determined by the setting of the INLEN device command. Binary input records are never processed as device commands. The BINARY setting defaults to OFF for all terminal types.

Example: %BINARY=ON

Comment: The BINARY command applies only to input operations and is a subset of binary mode (see the description of the MODE device command).

July 1988

BLANK

Purpose: To conceal an input line for password protection or other privacy.

Prototype: %BLANK[=n]

Description: The BLANK command causes the first "n" characters of the next input line to be blanked out to conceal the text typed by the user. The default value for "n" is twelve characters. The maximum value is 15.

On full-duplex terminals, typed characters will not be echoed for the entire input line following the issuing of a BLANK command, regardless of the value of "n" given on the command.

On half-duplex terminals, the first "n" characters will be multiply overprinted to form a blackened region to conceal the input.

Example: %BLANK=15

The first fifteen characters of the next input line will be concealed if the terminal is operating in half-duplex mode. If it is operating in full-duplex mode, the entire line is concealed.

Comment: Terminals operating in full-duplex mode may also use whatever key is currently defined as the ECHO character to suppress printing (see the description of the EDIT device command).

July 1988

BROADCAST

Purpose: To inhibit the printing of broadcast messages from the system operator.

Prototype: %BROADCAST={ON|OFF}

Description: If the broadcast option is OFF, broadcast messages sent by the system operator to the terminal will be inhibited. If the broadcast option is ON, broadcast messages will be allowed. Furthermore if the BROADCAST option is OFF, the message generated by the FLIP command will also be inhibited. BROADCAST defaults ON.

Example: %BR=OFF

Broadcast messages will not be sent to the terminal.

Comment: The BROADCAST command is useful when printing a "final copy" of a document at a terminal. Otherwise, it is not recommended that users set BROADCAST to OFF since important messages from the system operator may be missed.

CC

Purpose: To enable or disable output carriage-control processing.

Prototype: %CC={ON|OFF|MOD}

Description: The CC command controls carriage-control processing for terminal output. If simultaneous connections exist from the same terminal (see the GRAB command), the user may set CC differently for each connection.

The various parameters result in the following actions:

- (1) CC=ON means the first character of each output record will be checked to see if it is a valid MTS carriage-control character. If it is a valid carriage-control character, the carriage-control will be interpreted; otherwise, the record is single-spaced and then printed in its entirety. When CC is ON, the MTS @CC modifier is ignored.
- (2) CC=OFF means that all lines will be single-spaced and printed as is, including any carriage-control characters. When CC is OFF, the MTS @CC modifier is ignored.
- (3) CC=MOD (the default) means that carriage-control is controlled by the MTS @CC FDname modifier (the standard MTS practice). If the MTS @CC modifier is specified when an output record is written, the record is treated as though CC were ON, i.e., the first character is interpreted as carriage-control. If the MTS @¬CC modifier is specified, the record will be single-spaced as though CC were OFF.

For further details on carriage-control processing, see Appendix F to this section and the section "Files and Devices" in MTS Volume 1, The Michigan Terminal System.

Example: %CC=OFF

All lines will be single-spaced; any carriage-control characters will be printed as text and otherwise ignored.

July 1988

CLOCK

Purpose: To change the data transmission rate of UMnet.

Prototype: %CLOCK=n

Description: The CLOCK command changes the data transmission (clock) rate for terminals that can operate at various speeds. The parameter "n" is the transmission rate in bits per second (bps). Values of "n" applicable are dependent upon the terminal type and the type of terminal port through which the connection is made. Most standard data rates from 50 bps to 19200 bps are supported.

UMnet sets the default data transmission rate for a particular terminal according to the data transmission rate detected at connection time.

Note that using the CLOCK command to change the data transmission rate for UMnet does not change the data rate for the terminal. This must be done separately.

Example: %CLOCK=300

This example resets the data transmission rate to 300 bps.

July 1988

CONNECTIONS

Purpose: To display a list of all UMnet connections open for a given terminal.

Prototype: %CONNECTIONS

Description: A line of output will be displayed for each open connection, including the initial connection and all grabbed connections.

Most of the information displayed is useful only to the Computing Center communications staff.

July 1988

CRDELAY

Purpose: To control the length of time that UMnet will delay before allowing echoing or printing to resume after a carriage return, line-feed, or new-line sequence is sent to the terminal.

Prototype: %CRDELAY=n

Description: The CRDELAY command (carriage-return delay) sets the amount of time that UMnet will delay before allowing the resumption of echoing or printing after performing a carriage return, line-feed, or new-line sequence thus allowing the terminal's carriage time to return to the left margin. The delay is specified by the parameter "n", where $1 \leq n \leq 255$ and "n" is in units of 10 milliseconds, e.g., a CRDELAY of 10 is equal to 100 milliseconds.

UMnet defaults CRDELAY at connection time according to the terminal type. For the default values for various terminal types, see the table of terminal types in the file *TERMTYPES or in Appendix B.

Example: %CRDELAY=20

This example establishes a delay of 200 milliseconds after a carriage return is performed before UMnet allows echoing or printing to resume.

Comment: See also the description of the TBDELAY device command.

DCC

Purpose: To change the character used to prefix device commands.

Prototype: %DCC={x|"x"}

Description: The DCC command changes the current device-command character (the character that precedes and identifies a device command) to any character. A lowercase letter assigned as the device-command character will be mapped to uppercase. The default is the percent sign "%".

Example: %DCC="/"

Any line beginning with a slash (/) will be interpreted as a device command.

Comment: Device-command characters are detected on input only. Nevertheless, device commands can be issued from MTS by using the MTS \$CONTROL command or CONTROL subroutine. For example, to issue a device command from an MTS sigfile, the user should put into the sigfile a command of the form

\$CONTROL *MSINK* command

A specific example might be

\$CONTROL *MSINK* DCC="/"

July 1988

DEQUEUE

Purpose: To delete input lines that have been queued ahead.

Prototype: %DEQUEUE [=n]
 %DQ [=n]

Description: The DEQUEUE command deletes queued input lines on a last-in-first-out basis to a level specified by "n", where "n" is an integer indicating the number of input lines to be deleted. If "n" is omitted, the entire input queue is deleted. This command is useful for deleting input lines that have been typed ahead without first having to interrupt a pending operation.

The DEQUEUE command has no effect when INTERLOCK has been enabled; in this case, an attention interrupt must be given to delete the input queue.

Example: %DEQUEUE=3

 This example deletes the last three input lines that have been queued.

July 1988

DONT

Purpose: To prevent MTS from breaking the connection after signoff.

Prototype: %DONT[={ON|OFF}]
%DON'T[={ON|OFF}]

Description: When the user signs off after issuing a DONT=ON command, MTS will not disconnect the terminal, but will instead print the MTS herald, thus offering another MTS connection. If no parameter is given, DONT=ON is assumed.

The DONT command must be issued individually for each connection to which the user wants it to apply.

Example: %DONT

The user will be offered another connection immediately upon signing off.

July 1988

DUPLEX

Purpose: To configure a terminal for full-duplex or half-duplex operation.

Prototype: %DUPLEX={ON|OFF}

Description: When DUPLEX is ON, UMnet assumes that the user's terminal is configured for full-duplex operation. This means that the keyboard and the printing mechanism of the terminal are independent. When a character is typed, it is transmitted to UMnet which queues the character for input to MTS and prints ("echoes") it at the terminal. In contrast, when a terminal operates in half-duplex mode, its keyboard and printer are connected, so that each time a key is pressed, the character is both transmitted to UMnet and immediately printed at the terminal.

When DUPLEX is OFF, UMnet does not echo characters typed by the user as they are received. The only echoing done in this case is for the special control-character sequences that it performs: those for end-of-line, end-of-file, line-delete, and attention. That is, when the user ends a line, UMnet echoes a line-feed; when the user signals an end-of-file condition, UMnet echoes a reverse slant (\) followed by a carriage return and a line-feed; when the user signals a line-delete function, UMnet echoes a pound-sign (#) plus carriage return and line-feed; and when the user issues an attention interrupt UMnet echoes exclamation point (!) plus carriage return and line-feed.

DUPLEX defaults ON for most terminal types. The MTS file *TERMTYPES and Appendix B show the default for each terminal type.

Example: %DUPLEX=OFF

The terminal will be assumed to be operating in a half-duplex manner, that is, doing its own echoing of text input.

Comments: The DUPLEX setting is a subset of FDX and HDX modes. Issuing the MODE=FDX command sets DUPLEX to ON, while the MODE=HDX command sets it to OFF.

See also the description of the ECHO device command which controls whether or not UMnet does any echoing at all.

ECHO

Purpose: To enable or disable the echoing of input characters and special echoing sequences.

Prototype: %ECHO={ON|OFF}

Description: The ECHO command controls whether or not UMnet does any echoing at all, either of input characters or of special control-character sequences. When ECHO is OFF, UMnet performs neither input echoing nor special echoing until

- (1) a ECHO=ON command is issued,
- (2) an attention interrupt is issued, or
- (3) a character with bad parity is received from the terminal.

See the description of DUPLEX device command for a list of the special-echo sequences performed by UMnet which ECHO=OFF suppresses. Note that for suppression of normal typed input only, the appropriate command is DUPLEX=OFF. Setting ECHO to OFF is useful mostly when reading binary data or input from an auxiliary data-storage device.

ECHO defaults to ON at the time of connection.

Example: %ECHO=OFF

Comments: MODE=BINARY sets ECHO to OFF.

Do not confuse the ECHO command with the ECHO keyboard editing function which refers to the suppression of printing on full-duplex terminals for only the current input line. See the description of the EDIT device command for further details.

July 1988

EDIT

Purpose: To control the keyboard editing characteristics of the terminal.

Prototype: %EDIT={?|ON|OFF| [table] [key=function]}

Description: The EDIT command allows the user to design and control the keyboard editing behavior of the terminal. The parameters that EDIT may take function as follows:

EDIT=?

The "?" parameter displays the current settings for all of keyboard editing functions. Initially, EDIT=? will display the default table. If the user sets EDIT to a different table, EDIT=? will display that table. Any functions that have been set individually will also be displayed.

EDIT={ON|OFF}

EDIT=OFF invokes the TAPE edit mode used for reading input from auxiliary data-storage devices attached to terminals. In this mode, the following control characters have their normal functions:

ETX (CTRL-C)	DC1 (CTRL-Q)
EOT (CTRL-D)	DC2 (CTRL-R)
CR (CTRL-M)	DC3 (CTRL-S)
DLE (CTRL-P)	DC4 (CTRL-T)

Several other control characters are ignored entirely:

NUL	SO (CTRL-N)
ENQ (CTRL-E)	CN (CTRL-X)
ACK (CTRL-F)	SUB (CTRL-Z)
LF (CTRL-J)	DEL

All alphanumeric characters and special graphics, as well as other control characters not listed above, are treated as text.

The user will not ordinarily need to issue the EDIT=OFF command, since the MODE=TAPE and MODE=BINARY commands automatically set EDIT to OFF.

The BINARY=ON device command ignores any setting of EDIT to produce the BINARY editing mode. This mode is

July 1988

for reading unedited input; when it is in effect, all characters are treated as text and only the BREAK key has special significance. To exit from BINARY editing mode, the BREAK key should be pressed which resets BINARY to OFF, ECHO to ON, and issues an end-of-line and an end-of-file condition. The previous setting of EDIT will be in effect. Pressing BREAK in this context does not send an attention interrupt to the computer. Another method of exiting from BINARY editing mode is to use the MTS \$CONTROL command or the CONTROL subroutine from the program that is reading the binary input.

After EDIT has been set to OFF, an EDIT=ON command returns the setting of EDIT to whatever had been in effect previously and also restores any user-programmed settings. The RESET device command sets EDIT to ON, but also restores the edit settings to the most recently invoked complete predefined table and clears any user-set alterations.

EDIT=table

Two keyboard editing tables are available for users at the University of Michigan host. These are called the exact table (UM) and the superset table (UM+). Other tables are available at other hosts.

<u>Function</u>	<u>UM+</u>	<u>UM</u>
ATTN	CTRL-E/BREAK	CTRL-E/BREAK
DATA	Everything else	Everything else
DISCARD	CTRL-Z	CTRL-Z
DL	RUBOUT/CTRL-X/CTRL-N	RUBOUT/CTRL-N
DPC	BACKSPACE/CTRL-H	BACKSPACE/CTRL-H
ECHO	ESCAPE	ESCAPE
EOF	CTRL-C	CTRL-C
EOL	RETURN	RETURN
EOT	CTRL-D	CTRL-D
HALTOUT	CTRL-S	CTRL-S
IGNORE	NUL/CTRL-F/CTRL-J	NUL/CTRL-F/CTRL-J
LITERAL	CTRL-P	CTRL-P
PAUSE	CTRL-T	-
REPEAT	CTRL-W	-
RESUME	CTRL-Q/CTRL-R	CTRL-Q/CTRL-R

EDIT key=function ...

The EDIT command allows the user to program a personalized keyboard editing function by assigning any key or control-key pair to any of the fifteen UMnet keyboard editing functions.

July 1988

Several key assignments may be entered on the same line along with an entire table assignment (if desired), i.e.,

```
EDIT=table key=function ...
```

The key assignments will override the designated parts of the table. At any one time, all or none of the fifteen UMnet editing functions may be specifically assigned. The user may program an individual table so that many different keys or control-key pairs invoke the same function. However, multiple functions may not be assigned to the same key or control-key pair; each key or control-key pair has one and only one function which is the one assigned to it last.

"key" may be assigned in several ways. "key" may be any ASCII character including an alphabetic character, a number, or a control-key pair. The simplest way to assign "key" is to type it. This will always work if that key is DATA, that is, if it does not already have a keyboard editing function assigned to it. The reassignment of a key that already has a non-DATA function requires one of the alternate methods given below, since UMnet will perform the function assigned to that key whenever it reads it. These alternate methods may be used at any time, even if the key to be assigned is DATA.

- (a) The typing of the key to be assigned can be preceded by a literal-next indication, that is, the LITERAL character (defaults to CTRL-P).
- (b) If "key" is a control-key pair, "key" may be designated by spelling out the name of the control-key pair. For example, the pair of CONTROL and B may be entered as

```
CONTROL B   or CONTROL-B
CTRL B      or CTRL-B
CNTL B      or CNTL-B
CTL B       or CTL-B
CTRLB
```

All of these may be entered in either uppercase or lowercase.

- (c) If "key" is a control-key pair, "key" may be designated by giving the standard ASCII mnemonic for it. Appendix C gives the ASCII mnemonics, ASCII function names, and ASCII numeric values for the ASCII control functions.
- (d) Any "key" assignment may be specified as the numeric ASCII equivalent. Appendix D gives the coding for standard 7-bit ASCII. Any number

July 1988

appearing as "key" in a EDIT command is assumed to be a decimal value unless the modifier "X" (for hex) or "O" (for octal) is given and the number is enclosed in primes, e.g., X'1C' or O'34'. A decimal value may be explicitly indicated with a "D" modifier in the same way.

Restriction: Only four keys that are not control-key pairs may have keyboard editing functions assigned to them at any one time. That is, if "a", "b", "c", and "d" have been assigned to invoke editing functions, one of them will have to be reassigned back to DATA before UMnet will allow the assigning of anything other than a control-key pair to an editing function. If four keys that are not control-key pairs are already assigned to editing functions and a user tries to assign a fifth key, UMnet will print the message

```
%Edit overflow!
```

and will not allow the assignment.

Notice that DATA is not a function in the same sense as the others. The DATA function is merely the state of being no other function. Everything that is not otherwise assigned starts out as DATA; the way to unassign a key is assign it to DATA.

The fifteen UMnet keyboard editing functions are as follows:

ATTN	Attention interrupt
DATA	Transmit as data
DISCARD	Discard current output record
DL	Delete line
DPC	Delete previous character
ECHO	Disable/enable local echoing
EOF	End-of-file
EOL	End-of-line (return)
EOT	End-of-tape
HALTOUT	Hold output immediately
IGNORE	Do not process (ignore character)
LITERAL	Literal next
PAUSE	Stop output at end of current record
REPEAT	Echo current line locally
RESUME	Continue output

Examples: %EDIT=ON

The normal keyboard editing mode is restored.

```
EDIT DEL=IGNORE CTRL-Y=DL
```


July 1988

The DELETE key (often labeled RUBOUT) will now be completely ignored by UMnet while the CTRL-Y key-pair will cause a line-delete whenever it is typed.

Comment: See the descriptions for the BINARY, MODE, and READER device commands.

July 1988

FLIP

Purpose: To control which of several concurrent MTS connections is active.

Prototype: %FLIP [{+|-}]

Description: The FLIP command, in conjunction with the GRAB command, allows a user to have several (currently up to four) concurrent MTS connections from a single terminal. Only one of these connections can be "active" at a given time, and input may be given only to the active connection. Other grabbed connections are open but inactive. The FLIP command allows the user to activate any one of these open connections. Inactive connections may still send output to the terminal, and output from all open connections is intermixed at the terminal on a record-by-record basis unless the MIX option is set to OFF.

If the FLIP command is given without a parameter, UMnet reactivates the inactive connection that was most recently active.

The "+" and "-" parameters allow the user to flip forwards and backwards through the currently open connections. For example, if the active connection is the second one grabbed of three connections, a FLIP + command will take the user to the third connection while FLIP - will take the user to the first connection. The connections are assumed to "wrap around", that is, a FLIP + from the most recently acquired connection will activate the first connection.

Example: %FLIP

This will reactivate the most recently active connection.

Comment: See also the command descriptions for the GRAB, MIX, and QUIT device commands.

July 1988

FORMFEED

Purpose: To cause the terminal to perform its form-feed function at a top-of-page indication on output.

Prototype: `%FORMFEED[={ON|OFF}]`

Description: The FORMFEED command is used for copying files with embedded logical carriage controls to a printing terminal. When FORMFEED is ON, all MTS logical carriage-control "1" characters (top of logical page) are translated into ASCII form-feed characters (CTRL-L) so that any terminal that is capable of form-feeding will position the paper at the top of the next sheet.

When using the FORMFEED command, users should be sure to identify the terminal correctly to UMnet so that a sufficient TBDELAY is set up (see Appendix B), or set up a sufficiently large TBDELAY themselves, since the TBDELAY setting controls form-feed delays as well as horizontal tabbing delays.

The FORMFEED command defaults OFF for all terminals.

Example: `%FORMFEED=ON`

July 1988

GRAB

Purpose: To open multiple concurrent MTS connections from a single terminal.

Prototype: %GRAB

Description: The GRAB command allows the user to have open several MTS connections at one time from a single terminal. Currently, the maximum number of concurrent connections allowed is four.

The GRAB command establishes another connection from the user's terminal to MTS. This new connection becomes the "active" connection, i.e., the only connection to which input can be given. Previous connections that are still open become the "inactive" connections. Output can still be received from an inactive connection if MIX is ON, but no input can be sent to one until it is reactivated either by the FLIP command or by signing off the active connection.

If UMnet cannot establish another connection from the terminal to MTS or if the user already has the maximum number of connections, it will respond with

%No path available!

and return the user to the active connection.

Example: %GRAB

This will open a new connection from the user's terminal to MTS.

Comment: See also the descriptions for FLIP, MIX, and QUIT device commands.

July 1988

HELLO

Purpose: To display the time and system information.

Prototype: %HELLO

Description: The HELLO command displays the the time, date, and control-unit system information in the following format:

%hh:mm:ss mm-dd-yy XXnn PDP-11 MINOS (Version xxxxx)

where

"hh:mm:ss" is the current time of day in hours, minutes, and seconds,

"mm-dd-yy" is the current date,

"XXnn" is the source device name (the device from which the HELLO command was issued), and

the remainder identifies the control-unit computer, the operating system, and the current system version.

Example: %HELLO
%11:34:22 12-10-82 DT34 PDP-11 MINOS (Version EC062)

HEX

Purpose: To control hexadecimal input conversion.

Prototype: `%HEX=[{ON|OFF}] ["x"]`

Description: Hexadecimal input processing converts input consisting of an even number of hexadecimal digits (0-9 and A-F), delimited by the current hex-processing delimiter, into a string of half as many characters having the binary value represented by the hexadecimal number. Hexadecimal strings may appear anywhere on an input line (which may also contain normal text input).

Hexadecimal conversion defaults OFF and the delimiter defaults to a prime ('). If the user chooses to specify a hexadecimal delimiter in the command, that delimiter must be enclosed in quotation marks (not primes) as shown in the prototype.

The RESET device command resets HEX to OFF and the delimiter to the prime.

When HEX is ON, all input lines are inspected for the hexadecimal delimiter character. All characters between pairs of delimiters are converted to hexadecimal before being transmitted to MTS. For example,

```
ABC'00'DEF'00'GHI
```

will result in the line

```
ABC*DEF*GHI
```

where each "*" represents an ASCII NUL character. When HEX is OFF, input lines are not inspected for the delimiter character.

Hexadecimal processing is useful for entering characters that are not available on the terminal keyboard.

This command is specific to a particular connection, i.e., HEX may be set differently for each concurrent MTS connection.

Example: `%HEX=ON "&"`

This enables hexadecimal input processing and sets the delimiter character to the ampersand.

July 1988

INLEN

Purpose: To set the logical input record length.

Prototype: %INLEN=n

Description: The INLEN command specifies the maximum input record length, i.e., the maximum length line that can be entered from the terminal. The length must lie in the range $1 \leq n \leq 32767$.

An input record is terminated and sent to MTS whenever the user enters an end-of-line or an end-of-file character, or whenever the number of characters specified by the setting of INLEN is reached, whichever comes first. (Of course, most input records are terminated by pressing the carriage-return key.) When an input record is terminated by reaching the value of INLEN, UMnet will normally indicate this by echoing a carriage-return and line-feed. However, if ECHO is OFF or BINARY is ON (or MODE=BINARY, which sets ECHO to OFF and BINARY to ON), UMnet will terminate the record and pass it to MTS without additional indication.

The default value for INLEN is 240 for all terminal types. If the value of INLEN is less than 240, issuing an attention interrupt will reset the value to the default. If the value of INLEN is greater than 240, an attention interrupt will not reset its value.

Example: %INLEN=120

Whenever the current input record reaches 120 characters in length, it will be terminated and sent to MTS.

INTERLOCK

Purpose: To control when the echoing of input records occurs.

Prototype: %INTERLOCK={OFF|ON}

Description: When INTERLOCK is ON, the echoing of user input is inhibited until MTS reads the input.

Moreover, device commands will not be processed immediately as is the case when INTERLOCK is OFF. Thus, the DEQUEUE device command has no effect and queued input must be purged by issuing an attention interrupt.

INTERLOCK defaults OFF for all terminals for which this is a meaningful option.

Example: %INTERLOCK=ON

Comments: Some users like this feature because it causes input and output to be printed in the correct order even if the user has typed ahead on a full-duplex terminal.

July 1988

JOB

Purpose: To display job status information from MTS.

Prototype: %JOB

Description: The JOB command displays job status information in the following format:

```
*MNETRTN (MODEL xxxxx) HOST=UM JOB#=nnnnn DEV=MNnn
USER=ccid
```

where

"xxxxx" gives the version number of the current MTS Device Support Routine (DSR),

"nnnnn" is the user's MTS task number,

"MNnn" is the MTS device name for this connection, and

"ccid" is the user's MTS signon ID.

Example: %JOB
*MNETRTN (MODEL EC092) HOST=UM JOB#=00248 DEV=MN56
USER=WXYZ

LCI

Purpose: To control case conversion on input to MTS.

Prototype: %LCI [= {ON|OFF}]

Description: The LCI command controls the conversion of lowercase and mixed-case input to uppercase. When LCI is OFF, all alphabetic input is converted to uppercase. When LCI is ON (the default), no case conversion is done. When the command is given as %LCI (without a righthand side), %LCI=ON is assumed.

For terminals which do not themselves process this command, uppercase input conversion may be set independently for each grabbed connection to MTS. However, for some intelligent terminals such as the Ontel, the terminal itself does the uppercase input conversion, and thus the conversion will be the same for both connections.

LCI defaults to ON (no case conversion).

Example: %LCI=OFF

Lowercase and mixed-case input will be converted to uppercase.

Comments: Note that the LCI and UCI commands both perform the same function, namely to control the conversion of lowercase and mixed-case input to uppercase. The command LCI=OFF is equivalent to UCI=ON; LCI=ON is equivalent to UCI=OFF.

July 1988

LINES

Purpose: To define the video-terminal screen size in order to allow a periodic hold on video output.

Prototype: %LINES=n

Description: When the PAGEWAIT option is enabled, the number of lines that are written onto the video screen before waiting is determined by the setting of LINES. Initially, UMnet assigns a value to LINES that corresponds to the number of lines that fit onto the screen for storage-tube terminals. For scrolling terminals, UMnet writes one line less than the value of LINES before pausing, in order to leave a "context" line from the previous screenful of output.

LINES may be set to any value in the range $1 \leq n \leq 255$. Thereafter, if PAGEWAIT is enabled, UMnet will always write "n" lines before "freezing" the display.

Example: %LINES=16

If PAGEWAIT is enabled, UMnet will freeze the display each time 16 lines of output have been written on the screen.

Comment: See also the descriptions of the PAGEWAIT and SCROLL device commands.

MIX

Purpose: To control the mixing or separation of MTS output from two concurrent connections.

Prototype: %MIX={ON|OFF}

Description: When a second MTS connection has been grabbed, nonbinary data output from both connections is intermixed on a line-by-line basis by default (MIX=ON) as it is sent to the terminal.

When MIX is OFF, only output from the "active" connection (the connection last grabbed or flipped to) will appear at the terminal. To receive output from another connection, that connection must be flipped to. Output is held until its connection is made active.

Example: %MIX=OFF

Comments: If MIX is set to OFF and then set back to ON, all waiting output from all concurrent connections will be printed immediately, intermixed line-by-line.

See also the descriptions of the FLIP and GRAB device commands.

July 1988

MODE

Purpose: To enable simultaneously a specified group of terminal options.

Prototype: %MODE={FDX|HDX|TAPE|BINARY}

Description: The MODE command allows the simultaneous setting of a group of associated terminal characteristics. The default setting at connection time is MODE=FDX (full-duplex) for all terminals capable of full-duplex operation. To override this default at connection time, a user may type "go" when UMnet prompts

%Terminal=

This will identify the terminal as half-duplex.

The parameters of the MODE command have the following effects:

FDX: In FDX (full-duplex) mode, all input entered at the terminal is echoed by UMnet, including the special echoing sequences for editing control characters (see the EDIT command). For a discussion of full-duplex and half-duplex operation, see the DUPLEX command. Issuing a MODE=FDX command is equivalent to simultaneously issuing the following device commands:

```
BINARY=OFF      EDIT=ON
DUPLEX=ON       READER=OFF
ECHO=ON
```

HDX: In HDX (half-duplex) mode, the terminal itself performs all the printing of normal input characters and UMnet echoes only the special sequences for editing control characters. Typing ahead during output while operating in half-duplex mode is not recommended since the typed-ahead input will be intermixed with the output. Issuing a MODE=HDX command is equivalent to simultaneously issuing the following device commands:

```
BINARY=OFF      EDIT=ON
DUPLEX=OFF      READER=OFF
ECHO=ON
```

TAPE: TAPE mode is useful for reading paper-tape, magnetic-tape cassette, and floppy-disk text (nonbinary) input containing embedded line editing characters, such

July 1988

as BACKSPACE and DELETE, which are intended as normal data characters. (Note that text that is formatted as normal keyboard input may be read by simply issuing a READER=ON command.) Issuing a MODE=TAPE command is equivalent to simultaneously issuing the following device commands:

```
BINARY=OFF      EDIT=OFF
ECHO=ON         READER=ON
```

To read text input, the user should ready the device and issue a MODE=TAPE command. The input should end with a CTRL-D (ASCII EOT); Hermes will turn READER OFF when it reads the CTRL-D.

BINARY: BINARY mode is useful for reading binary data from paper-tape, magnetic-tape cassette, and floppy-disk devices. Issuing a MODE=BINARY command is equivalent to simultaneously issuing the following commands:

```
BINARY=ON      READER=ON
ECHO=OFF
```

To read binary input, ready the device and issue a MODE=BINARY command.

To exit from BINARY mode, press the BREAK key. This will stop the input, send an end-of-line and an end-of-file to MTS, and reset BINARY to OFF, ECHO to ON, and READER to OFF. This use of BREAK does not send an attention interrupt to MTS. Another way of exiting from BINARY mode is to use the MTS \$CONTROL command or CONTROL subroutine from an MTS program that is reading binary input.

Example: %MODE=HDX

The user's terminal will now operate in half-duplex mode.

Comment: See the READER device command for more information about input from auxiliary data-storage devices.

July 1988

NPC

Purpose: To control the echoing/printing of nonprinting characters.

Prototype: `%NPC={ON|OFF|{x|"x"}}`

Description: By default, Hermes echoes on input (for full-duplex terminals) and prints on output (for all terminals) a question mark "?" in place of all nonprinting characters. Except for lowercase letters, any character may be specified as the NPC replacement character in place of the question mark. The NPC replacement character may be specified as the character itself or as the character enclosed in question marks. The use of the replacement character may be disabled entirely by setting NPC to OFF.

If the user sets a new NPC character and then sets NPC to OFF, UMnet will remember the new character and will use it rather than the default question mark if NPC is subsequently set to ON.

Both the RESET and TERMINAL device commands reset NPC to ON. Users who set NPC to OFF for graphics output should do so after issuing any RESET and TERMINAL commands.

For all terminals identified as half-duplex, the NPC substitution occurs on output only.

Examples: `%NPC=" "`

All nonprinting characters will be printed or echoed as a blank (one horizontal space). This is useful with output that contains characters not in the terminal's character set. Blanks will appear in the appropriate places and can be filled in later.

`%NPC=OFF`

The presence of nonprinting input or output characters will not be indicated in any manner.

July 1988

OPERATOR

Purpose: To send a message to the UMnet control unit operator's console.

Prototype: %OPERATOR messagetext

Description: The OPERATOR command prints a message on the operator's console of the UMnet control unit along with the current time of day. The message should not exceed 100 characters in length; excess characters will be ignored.

The message is "announced" on the operator's console by the automatic ringing of several bell characters. The OPERATOR command is intended for alerting the operations staff to UMnet control unit problems encountered by users.

Example: %OPER This phone line is giving me data checks.

Comment: There is no guarantee that the message will be noticed and responded to immediately, especially outside of normal business hours.

July 1988

OUTLEN

Purpose: To set the terminal's output record length.

Prototype: %OUTLEN=n

Description: The OUTLEN command controls the maximum length of output records. OUTLEN defaults to a value appropriate for each terminal type; see the MTS file *TERMTYPES or Appendix B for the values for each terminal type. The value of OUTLEN must be in the range $20 \leq n \leq 32767$.

Example: %OUTLEN=132

Records being send from MTS to the terminal for output will be truncated at 132 characters.

Comment: See also the description of the WIDTH device command.

PAGEWAIT

Purpose: To provide for a periodic hold on video-terminal output.

Prototype: %PAGEWAIT={ON|OFF|n}

Description: When PAGEWAIT is set to ON, output is automatically halted and held on the screen each time the bottom of a logical "page" is reached. The size of the logical page is initially set to equal the size of the video screen, as determined by UMnet from the terminal type. The size may be set to a different value by the LINES device command.

The PAGEWAIT command defaults to OFF for all terminals. Once it has been set to ON, either explicitly or with a %PAGEWAIT=n command (where "n" specifies the duration of the hold), UMnet counts the number of lines of output sent to the screen. Whenever this number reaches the value of LINES, UMnet "freezes" the output on the screen.

If PAGEWAIT=ON has been specified, the hold on output will last until the user restarts output by entering a RESUME keyboard-editing character or by entering other keyboard input. Users with storage-tube video terminals should first clear the screen to keep the new output from overwriting the old. Users with scrolling terminals may clear the screen before restarting output or may simply allow the new output to scroll up from the bottom. In either case, the line counter is reset and output will again "freeze" when the value of LINES is reached. For differences in the resetting of the line counter for storage-tube and scrolling video terminals, see the description of the SCROLL device command.

The user may also specify PAGEWAIT=n, where "n" gives the length of the holding period in seconds ("n" must be in the range 1≤n≤255). When this is done, UMnet will automatically restart output after waiting "n" seconds with output held on the screen. Output may be resumed before the delay period expires by entering a RESUME keyboard-editing character or by entering other keyboard input. This settable-delay feature is useful primarily for scrolling video terminals.

The RESET device command resets PAGEWAIT to OFF.

The PAGEWAIT feature has no effect during binary output-

July 1988

Examples: %PAGEWAIT=10

Each time the number of output lines reaches the value LINES, the screen will "freeze" for 10 seconds before another screenful is written.

%PAGEWAIT=ON

Each screenful of output will be held until it is released by the user.

Comment: See also the descriptions of the LINES and SCROLL device commands.

PARITY

Purpose: To control parity checking on input records.

Prototype: `%PARITY={OFF|ON}`

Description: UMnet automatically sends all nonbinary output as even parity ASCII characters. When PARITY is ON, even parity is also checked on input characters. When a parity error is detected on input, UMnet deletes the record in which the error occurred and prints a DATA CHECK message at the terminal.

When PARITY is OFF, all parity errors on input are ignored, and the received characters are processed as though they had correct parity.

For terminals with automatic answerback mechanisms, PARITY defaults ON when the answerback code has correct even parity and OFF when it does not. For other terminals, PARITY defaults OFF.

Example: `%PARITY=OFF`

Any parity errors on input will be ignored by UMnet.

Comment: UMnet does not support odd-parity input checking for terminals.

July 1988

PFX

Purpose: To enable or disable the printing of MTS prefix characters on input and output records from MTS.

Prototype: %PFX={ON|OFF|MOD}

Description: If PFX (prefix) is ON, the current MTS prefix characters, if any, are printed at the beginning of every line of input and output.

If PFX is OFF, prefix characters will not be printed .

If PFX is MOD (the default), the @SP MTS FDname I/O modifier controls the inclusion of the prefix characters; @SP causes the prefix characters to be omitted, and @¬SP causes them to be included.

Example: %PFX=OFF

Records from MTS will be printed at the terminal without the MTS prefix characters.

July 1988

QUIT

Purpose: To disconnect the terminal from the active MTS connection.

Prototype: %QUIT

Description: The QUIT command immediately disconnects the terminal from the active MTS connection. No log-out or sign-off statistics are generated, and any running program is aborted. If there is another connection (through use of the GRAB command), UMnet reactivates that connection.

The QUIT command will disconnect the user's terminal from UMnet if another connection does not exist.

Example: %QUIT

If there is only one MTS connection, UMnet will disconnect the user's terminal. If there is a grabbed connection, the currently active connection will be terminated and the inactive connection reactivated.

Comments: The QUIT command is useful for terminating an MTS connection when MTS is not responding.

See also the descriptions of the GRAB and FLIP device commands.

July 1988

READER

Purpose: To provide control over an auxiliary data-storage device attached to a terminal.

Prototype: %READER={OFF|ON}

Description: The READER command is used to control auxiliary devices at the user's terminal. It defaults OFF for all terminal types. Input from auxiliary devices can be read only from full-duplex terminals equipped with a device that responds to X-ON (CTRL-Q, ASCII DC1) and X-OFF (CTRL-S, ASCII DC3).

When READER is ON, UMnet generates the X-ON and X-OFF characters for input device control and the auxiliary device will be started. Both the MODE=TAPE and MODE=BINARY commands set READER to ON. Nonbinary input should end with an EOT (CTRL-D), which will stop the device.

Using the READER command alone will work properly only when reading text (nonbinary) input records that are formatted as normal keyboard input.

Examples: To read input from an auxiliary data-storage device attached to a terminal, the user first should ready the device and then

- (a) for text input formatted as normal keyboard input, type %READER=ON,
- (b) for text input with embedded control characters which are to be treated as ordinary data, type %MODE=TAPE, or
- (c) for binary input, type %MODE=BINARY.

Comment: When nonbinary data is being read and printing of the data is not required, a ECHO=OFF command may be issued. (MODE=BINARY turns ECHO to OFF automatically so that this is not necessary for binary input.)

RESET

Purpose: To reset user-settable options.

Prototype: %RESET

Description: The RESET command resets the command settings shown in the table below to their default or initial values. Those commands described as "Reset Globally" are reset for both connections if a second connection has been grabbed. Those described as "Reset Locally" are performed only for the connection on which the RESET command is issued.

Reset Locally

CC=MOD	OUTLEN=*
DON'T=OFF	PFX=MOD
FORMFEED=OFF	TABI=OFF "TAB" 10,16,35
HEX=OFF ""	UCI=*
INLEN=240	WIDTH=*
LCI=*	

Reset Globally

BACKSPECIALECHO=*	MIX=ON
BINARY=OFF	NPC=ON
BROADCAST=ON	PAGEWAIT=OFF
CRDELAY=*	READER=OFF
DCC="%"	SCROLL=*
ECHO=ON	TABSPECIALECHO=*
EDIT=ON	TBDELAY=*
LINES=*	

In the table, an asterisk (*) means that the value will be reset to the setting determined at connection time, or if the user has subsequently issued a TERMINAL command, to the setting established at that time. For the initial values of these settings, see the MTS file *TERMTYPES or Appendix B.

Example: %RESET

Comments: When a TERMINAL command is issued, the terminal characteristics are reinitialized to the default characteristics of the new terminal type. For more information on the relationship between RESET and TERMINAL, see the description of the TERMINAL device command.

July 1988

SCROLL

Purpose: To control the behavior of the line counter for video-terminal output when the PAGEWAIT feature is enabled.

Prototype: %SCROLL={ON|OFF}

Description: When the PAGEWAIT feature is set to ON or "n", the setting of SCROLL determines when the line counter is reset. UMnet initializes the setting of SCROLL at connection time according to whether the terminal is a scrolling or storage-tube video terminal.

For storage-tube terminals, SCROLL defaults to OFF. This means that the line counter is reset whenever a page-wait is terminated either by the user or by a time-out, or whenever a RESUME keyboard-editing character is entered whether or not the terminal is in page-wait. The screen is then allowed to refill up to the value of LINES, at which point a page-wait will again take effect. By default, the RESUME character is CTRL-Q or CTRL-R (in some cases both) in the predefined keyboard editing tables. RESUME may be set by the user to any other character (see the EDIT command description).

For scrolling terminals, SCROLL defaults to ON. This means that, in addition to being reset when a page-wait is terminated, the line counter is also reset each time that input is requested by the system (thereby indicating that the current output is at an end). In essence, this means that the line count starts afresh each time a new input-output sequence is begun, even when it begins in the middle of the screen, because the output will scroll upward when necessary. Output will continue only to the point at which the last input line (or the last line of the previous screenful of output) is at the top of the screen to provide context.

SCROLL=ON is not a useful setting for storage-tube terminals since the resetting of the line counter at points other than the page top would result in the overwriting of part of the screen. However, users of scrolling terminals may choose to set SCROLL to OFF.

Example: %SCROLL=OFF

The line counter will be reset only at the termination of a page-wait or receipt of a RESUME character.

SENSE

Purpose: To display information about the configuration of the user's terminal.

Prototype: %SENSE

Description: The SENSE command displays information about the user's terminal in the following format:

%xxnn:termtype ER=n ddinfo

where:

"xxnn" is the terminal device-type code and port number,

"termtype" is the terminal identifier,

"ER=n" represents the number of received errors (data check, data lost, and parity) since the connection was established, and

"ddinfo" is a variety of different parameters and their values (in hexadecimal digits) representing device-dependent information applicable to the specific connection.

Example: %SENSE

Comment: In the event that a user suspects that the UMnet control unit is malfunctioning, it is recommended that the information displayed by this command be shown to a member of the Computing Center communications staff.

July 1988

SNS

Purpose: To provide information about the terminal connection to an executing MTS program.

Description: The SNS command may only be issued by calling the MTS CONTROL subroutine (see MTS Volume 3, System Subroutine Descriptions), i.e., it may not be entered from the terminal or as an MTS \$CONTROL command.

The SNS command returns up to 56 bytes of information about the terminal. The calling program must provide an information area (the first argument of the call to CONTROL) which has "SNS" as the first 3 characters. The length (the second argument) should indicate the length (in bytes) of the information area (including the "SNS"). The format of the information returned to the calling program information area is as follows:

<u>Byte</u>	<u>Function</u>
0-2	"SNS" (supplied by calling program)
3	Binary zero (X'00')
4-7	MTS device name ("MNnn")
8-11	MTS device type ("MNET")
12-15	Control unit name ("MN1 ")
16-19	Control unit type ("MNET")
20-21	Connection type ("TL")
22-23	Remote host id ("UM")
24-47	Optional identifiers
48-55	Terminal type (8 characters)

Comment: This list is subject to change as the services provided by the UMnet control unit are expanded.

July 1988

STATUS

Purpose: To display various operating statistics for a particular UMnet control unit computer.

Prototype: %STATUS

Description: The STATUS command displays information about the system in the following format:

```
%m id=x ...  
%B=n,n,n,n GF=n OF=n FF=n OR=n NF=n NC=n
```

where:

"m" is the total number of current open connections into which the terminal is connected,

"id=x" represents a UMnet control unit computer ID and the number of currently open connections between it and the control unit computer into which the terminal is connected, and

"B", "GF", "OF", "FF", "OR", "NF", and "NC" represent various statistics describing available system space and space-acquisition failures. This information is of use only to the Computing Center communications staff.

Example: %ST

July 1988

TABI

Purpose: To control tabbing for input records to MTS.

Prototype: %TABI=[{ON|OFF}] ["x"] [t,...]

Description: The TAB command controls tabbing and tab settings for input records. The default setting of TABI is OFF; when turned ON, the tab character ("x") defaults to TAB (or CTRL-I) and the tab stops default to columns 10, 16, and 35. If the user chooses to specify a different tab character in the command, that character must be delimited with quotation marks (not primes) as shown in the prototype. This command is effective only for the active connection; inactive grabbed connections are not affected.

When TABI is turned OFF by the user, any user-set values for tab character and tab stops are retained and will become effective again when TABI is turned back ON.

A maximum of 20 tab stops may be set. They must be in the range of the current value for INLEN.

Example: %TABI=ON "*" 5,10,30,55

Tabbing for input records is enabled. Whenever the user types an asterisk, the input record will be positioned at the column indicated by the next tab stop. Tab stops are set at columns 5, 10, 30, and 55.

Comment: See also the description of the TABSPECIALECHO device command.

TABSPECIALECHO

Purpose: To echo a space when an input tabbing operation is performed on full-duplex terminals without a physical tabbing capability.

Prototype: `%TABSPECIALECHO={ON|OFF}`

Description: The TABSPECIALECHO command controls the character echoed by UMnet in response to TAB (or CTRL-I), vertical tab (CTRL-K), or form-feed (CTRL-L). When TABSPECIALECHO is ON, UMnet echoes a space in response to these characters. When TABSPECIALECHO is OFF, the actual character is echoed and will cause carriage motion if the terminal can do so.

See the MTS file *TERMTYPES or Appendix B for the default for specific terminals. TABSPECIALECHO defaults to ON for terminals with electronic tabbing, because if a user presses the tab key on such a terminal before explicitly setting up tabs, the print-head or cursor may shoot directly to the end of the carriage. Users with these terminals who want to set physical tabs should set TABSPECIALECHO to OFF.

Example: `%TABSPECIALECHO=ON`

Whenever the user types a tab character, UMnet will echo a space at the terminal.

Comment: See also the descriptions of the BACKSPECIALECHO and TABI device commands.

July 1988

TBDELAY

Purpose: To specify the amount of time that UMnet will delay before resuming printing or echoing after a tabbing operation.

Prototype: %TBDELAY=n

Description: The TBDELAY (tab delay) command allows the user to set the length of time UMnet waits after a TAB (CTRL-I), vertical tab (CTRL-K), or form-feed (CTRL-L) is printed, before the resumption of printing or echoing, to allow for mechanical delay in the positioning of the carriage. The delay is specified by the parameter "n", which must be in the the range 1≤n≤255. The unit of "n" is 10 milliseconds, e.g., a TBDELAY of 10 is equal to 100 milliseconds.

The setting of TBDELAY controls output tabbing delay times for all terminals. It has no effect on input tabbing delays for terminals operating in half-duplex mode.

UMnet sets TBDELAY at connection time according to the terminal type. See the MTS file *TERMTYPES or Appendix B for the default for specific terminal types.

Example: %TBDELAY=20

This example establishes a delay of 200 milliseconds after a TAB, vertical tab, or form-feed is printed before UMnet will resume echoing or printing.

Comment: See also the description of the CRDELAY device command.

July 1988

TERMINAL

Purpose: To change the identification of an already-connected terminal.

Prototype: %TERMINAL=termtype

Description: The TERMINAL command reconfigures the user's terminal as the type specified by the "termtype" parameter.

The TERMINAL command will change the following command settings to correspond to the default values associated with the specified termtype: BACKSPECIALECHO, CRDELAY, INLEN, OUTLEN, TABSPECIALECHO, and TBDELAY. It will also change the terminal type field returned by the SNS and SENSE commands.

The TERMINAL command is useful when a terminal has been incorrectly identified at connection time. The SENSE command can be used to display the current terminal type.

The MTS file *TERMTYPES and Appendix B list the configuration parameters for each terminal type supported by UMnet.

Example: %TERMINAL=LA36

The connected terminal will be identified and configured as a LA36 DECwriter.

Comments: Users with a terminal not listed in *TERMTYPES or Appendix B should choose the terminal type of a similar terminal. For wide-carriage printing terminals, using "LA36" will usually work well. Terminals with thermal printers can use "TI700" or "TI745" depending on whether the terminal requires a carriage-return delay or not, respectively. For scrolling video terminals, "CRT" is generally appropriate; for storage-tube video terminals "T4010" will frequently work. Users may also call the Computing Center communications staff for advice.

July 1988

TEST

Purpose: To generate a set of test lines to test the print mechanism of a terminal.

Prototype: %TEST [length]

"length" is a decimal number giving the output line length of the terminal. If omitted, it defaults to the value of INLEN.

Description: A set of test lines is output continuously to the terminal. To abort this command, the user must enter an attention interrupt by typing CTRL-E or BREAK. If the BACKSPECIALECHO option is not enabled, a "backspace test" consisting of four lines starts the test. Otherwise, the test starts with what is commonly referred to as a "ripple pattern." This pattern is the set of printable ASCII characters. Each successive line is shifted left one character position from the previous line. After 94 lines of the ripple pattern, the test is repeated.

The TEST command may only be issued from the terminal, i.e., it may not be issued from MTS through the CONTROL subroutine.

Example: %TEST 79

A set of test lines of length 79 is generated.

%TE

A set of test lines of length equal to the value of INLEN is generated.

July 1988

UCB

Purpose: To display the contents of part of the user's private control storage within the UMnet control unit.

Prototype: %UCB

Description: The UCB command causes the contents of the Unit Control Block (UCB) to be displayed at the terminal in hexadecimal digits. The UCB constitutes part of the private control storage associated with the particular port to which the terminal is connected.

Displaying the UCB is useful primarily when the user suspects a control unit malfunction. The information printed by the UCB command should be given to a Computing Center communications staff member in such a case.

Example: %UCB

July 1988

UCI

Purpose: To control case conversion on input to MTS.

Prototype: %UCI={ON|OFF}

Description: The UCI command controls the conversion of lowercase and mixed-case character input to uppercase. When UCI is ON, all alphabetic input is converted to uppercase. When UCI is OFF (the default), no case conversion is done.

For terminals which do not themselves process this command, uppercase input conversion may be set independently for each connection to MTS. However, on some intelligent terminals such as the Ontel, the terminal itself does the uppercase input conversion and thus the conversion will be the same for both connections.

Example: %UCI=ON

Lowercase and mixed-case input will be converted to uppercase.

Comments: Note that the UCI and LCI commands both perform the same function, namely to control the conversion of lowercase and mixed-case input to uppercase. The command UCI=ON is equivalent to LCI=OFF; UCI=OFF is equivalent to LCI=ON.

July 1988

UCO

Purpose: To control case conversion on output from MTS.

Prototype: %UCO={ON|OFF}

Description: The UCO command controls the conversion of lowercase and mixed-case character output from MTS to uppercase. When UCO is ON, all alphabetic output is converted to uppercase before being printed. When UCO is OFF (the default), no case conversion is done.

Example: %UCO=OFF

Lowercase output will be printed without change.

July 1988

WAIT

Purpose: To override the 30-minute inactivity timer.

Prototype: %WAIT

Description: Unless the WAIT command is issued, MTS starts a 30-minute timer running each time it awaits input from the terminal. If 30 minutes elapse without a completed input line being entered, the user is given a warning that if no input line is entered within two minutes, the user will be signed off from that connection. If there is an inactive grabbed connection, it will then become the active connection.

Those device commands which are processed totally by UMnet do not affect the 30-minute timer since they do not reach MTS.

Issuing the WAIT command overrides the inactivity timer, causing MTS to wait for input indefinitely. However, the WAIT command applies only to the current time-out. It must be repeated every time MTS is waiting for input, if the user expects to enter no input for the next 30 minutes and wants to prevent the automatic signoff.

Example: %WAIT

WIDTH

Purpose: To specify the width of output lines printed at the terminal.

Prototype: %WIDTH={n|ON|OFF}

Description: The WIDTH command allows the user to specify the terminal carriage or screen width. The value of WIDTH may be set to a maximum of 255.

The value of WIDTH is used for two purposes:

- (1) MTS commands and programs can obtain the current WIDTH value from the GDINFO subroutine and use the value to format output lines so that the information "fits" on carriage or screen lines in a reasonable way.
- (2) The numerical value of WIDTH can be used in conjunction with the ON and OFF parameters to control the "wrapping" of long output lines. If WIDTH is ON and the WIDTH value is less than the length of the output line, the line will be "wrapped around," i.e., printed as two or more lines, all but the last of which will be of length WIDTH. If WIDTH is OFF, wrapping is disabled.

The default WIDTH values for each terminal type are listed in the MTS file *TERMTYPES or Appendix B. WIDTH defaults to ON for all terminals except OP/VIS and OP/MTS.

Example: %WIDTH=25

Any output lines longer than 25 characters will be continued on a second and, if necessary, subsequent line(s) of not more than 25 characters each.

July 1988

?

Purpose: To display information about the configuration or the terminal and UMnet device-command settings.

Prototype: %?

Description: The ? command displays information about the terminal configuration and UMnet device-command settings.

Example: %?

July 1988

n±n

Purpose: To perform simple arithmetic operations and interbase conversions in three bases.

Prototype: %n±n

Description: In the prototype above, the arguments may appear in any of the following forms:

n - hexadecimal format
X'n' - hexadecimal format
D'n' - decimal format
O'n' - octal format

The two arguments need not be in the same base. Each is restricted to a maximum absolute value of 65535 (decimal).

The result of the addition or subtraction is printed in all three bases, for example

%D'10'+X'10'
%X'001A' D'00026' O'000032'

To perform simple interbase conversions, the user must enter the number to be converted and then add or subtract zero.

July 1988

APPENDIX B: TERMINAL CHARACTERISTICS

This appendix lists the valid terminal types that may be given in response to the "Enter terminal type" message that is printed when a terminal is connected to MTS and the terminal type cannot be determined automatically.

In the table below, UMnet will set the OUTLEN (maximum output line length) and WIDTH (right margin value) values to the indicated carriage width except as noted. The tab and carriage-return delays will be set to the indicated values (in units of 10 milliseconds) which have been found to be optimal for the terminal.

Additional feature codes are the following:

TABSP Typing a tab echoes a blank instead of tab.
 BA Typing a backspace echos an underscore.

See the public file *TERMTYPES for the most recent listing of the currently supported terminals.

<u>Terminal Name</u>	<u>Terminal Type</u>	<u>Screen Width</u>	<u>Delays TB,CR</u>	<u>Features (See above)</u>
Amigo (Ontel Amigo PC)	AMIGO	80	0,0	OUTLEN=255
Anderson-Jacobsen 630	AJ630	140	10,30	TABSP
Anderson-Jacobsen 830	AJ830	130	0,0	TABSP
Anderson-Jacobsen 832	AJ832	132	0,1	TABSP
Anderson-Jacobsen 833	AJ833	132	0,1	TABSP
Anderson-Jacobsen 860	AJ860	130	0,0	TABSP
Ann Arbor Terminals 24x80	AA2480	80	0,0	TABSP
Apple II	APPLE	39	0,0	TABSP
Chromatics CG series	CG1399	132	0,0	
CompuTek 400	C400	83	0,0	
Computer Devices Inc 1203	CD1203	79	0,25	TABSP
Control Data 713	or CDC713	80	0,0	TABSP
	CRT80	80	0,0	TABSP
CRT Display (generic)	CRT	79	0,0	TABSP
	or CRT80	80	0,0	TABSP
	or CRT132	132	0,0	TABSP
Data Products Portcom	PCOM	79	0,25	TABSP,BA
Data Terminals Corp 300	DTC300	130	0,30	TABSP
Data Terminals Corp 302	DTC302	130	0,30	TABSP
DEC GT40	DECGT40	72	0,0	
DEC LA34/LA36	LA36	130	0,0	TABSP
DEC LA120/LS120/TP1000	LA120	130	0,0	TABSP
DEC PDT11	PDT11	80	0,0	TABSP
DEC VT52	VT52	80	0,0	TABSP
	or CRT80	80	0,0	TABSP
DEC VT100	VT100	80	0,0	TABSP
(depending on	or CRT	79	0,0	TABSP
"setup mode"	or CRT80	80	0,0	TABSP

MTS 4: Terminals and Networks in MTS

July 1988

settings)	or CRT132	132	0,0	TABSP
Execuport 3000	TI700	79	0,25	TABSP
Gencom 300	GC300	130	0,0	TABSP
General Electric Terminet 30	GE30	132	20,40	TABSP,BA
General Electric Terminet 300	GE300	118	55,30	TABSP
Genisco 1000	G1000	73	0,0	
GSI 300	GSI300	130	0,30	TABSP
Hazeltine 2000	H2000	73	0,0	TABSP
Hewlett Packard 2621	HP2621	79	0,0	TABSP
	or CRT	79	0,0	TABSP
Hewlett Packard 2648	T4010	73	0,0	
IBM 3101	I3101	79	0,0	TABSP
IBM 5100	I5100	64	0,0	TABSP
IBM PC or XT	IBMPC	79	0,0	TABSP
Intelcolor 2400	I2400	80	0,0	TABSP
Macintosh	MAC	80	0,0	TABSP
Macintosh (with Versaterm)	VERSA	80	0,0	TABSP
Megatek	MEGATEK	79	0,0	TABSP
Ontel (OP/TTY or SUNY pgms)	ONTEL	79	0,0	TABSP
Ontel (MTS MCP pgm)	OP/MTS	80	0,0	OUTLEN=255
Ontel (MTS Visual Mode pgm)	OP/VIS	80	0,0	OUTLEN=255
Osborne	OSBORNE	52	0,0	TABSP
Process Technology SOL	SOL	63	0,0	TABSP,BA
Qume Sprint 5	QUMES5	130	0,30	TABSP
Radio Shack TRS80	TRS80	63	0,0	
Research Inc. Telray 33	RI33	80	0,0	TABSP,BA
Research Inc. Telray 39	CRT80	80	0,0	TABSP
Tektronix 4002	T4002	83	0,0	
Tektronix 4006	T4006	73	0,0	TABSP,BA
Tektronix 4010/4012/4013	T4010	73	0,0	
Tektronix 4014	T4014	73	0,0	
Tektronix 4015	T4015	73	0,0	
Tektronix 4023	T4023	80	0,0	TABSP
	or CRT80	80	0,0	TABSP
Tektronix 4025	T4025	73	0,0	
Tektronix 4025 (with GIN)	T4025G	73	0,0	
Tektronix 4027	T4027	73	0,0	
Tektronix 4051	T4051	73	0,0	
Tektronix 4112	T4112	73	0,0	BA
Tektronix 4113	T4113	73	0,0	BA
Teletype 33 or 35	TTY	72	20,0	TABSP,BA
Teletype 38	TTY38	132	20,7	TABSP,BA
Teletype 43	TTY43	130	0,0	TABSP
Teletype Dataspeed 40	TTY40	80	15,15	TABSP
Texas Instruments 700	TI700	79	0,25	TABSP
Texas Instruments 745/765	TI745	79	0,0	TABSP
Trendata 4000	TD4000	130	0,0	
Univac DCT 500	DCT500	132	30,60	TABSP,BA
Visual 550	V550	80	0,0	
Westinghouse 1600	W1600	80	0,0	TABSP
Xerox 800 (at half-duplex)	X800	130	0,40	TABSP
Xerox 1620	X1620	130	0,30	TABSP
Xerox 1750	X1750	130	0,30	TABSP

July 1988

APPENDIX C: ASCII SPECIAL CHARACTERS

This appendix lists the set of ASCII special characters, their corresponding key-stroke sequence for most ASCII terminals, and their corresponding ASCII function name. The EDIT device command controls which key-stroke sequences are associated with a specific function.

Name	Hex	Key Stroke	Function
NUL	00	CTRL-SHFT-P ¹	Null
SOH	01	CTRL-A	Start of header
STX	02	CTRL-B	Start of text
ETX	03	CTRL-C	End of text (end-of-file)
EOT	04	CTRL-D	End of transmission
ENQ	05	CTRL-E	Enquiry (attention interrupt)
ACK	06	CTRL-F	Acknowledge
BEL	07	CTRL-G	Bell
BS	08	CTRL-H (or BACKSPACE)	Backspace (character delete)
HT	09	CTRL-I	Horizontal tab
LF	0A	CTRL-J (or LINE FEED)	Line feed
VT	0B	CTRL-K	Vertical tab
FF	0C	CTRL-L	Form feed
CR	0D	CTRL-M (or RETURN)	Carriage return
SO	0E	CTRL-N	Shift out
SI	0F	CTRL-O	Shift in
DLE	10	CTRL-P	Data link escape
DC1	11	CTRL-Q	Device control 1
DC2	12	CTRL-R	Device control 2
DC3	13	CTRL-S	Device control 3
DC4	14	CTRL-T	Device control 4
NAK	15	CTRL-U	Negative acknowledge
SYN	16	CTRL-V	Synchronous idle
ETB	17	CTRL-W	End of transmission block
CAN	18	CTRL-X	Cancel
EM	19	CTRL-Y	End of medium
SUB	1A	CTRL-Z	Substitute
ESC	1B	CTRL-SHFT-K ¹ (or ESC)	Escape
FS	1C	CTRL-SHFT-L ¹	File separator
GS	1D	CTRL-SHFT-M ¹	Group separator
RS	1E	CTRL-SHFT-N ¹	Record separator
US	1F	CTRL-SHFT-O ¹	Unit separator
DEL	7F	DEL (or RUBOUT)	Delete (line delete)

¹The ASCII characters given in the table with CTRL-SHFT-key combinations are for typewriter-paired keyboards (keyboards with the left and right parentheses defined as SHIFT-9 and SHIFT-0). For bit-paired keyboards (keyboards with the left and right parentheses defined as SHIFT-8 and SHIFT-9), these ASCII characters will have different CTRL-key combinations:

July 1988

NUL - CTRL-@	GS - CTRL-]
ESC - CTRL-[RS - CTRL-^
FS - CTRL-\	US - CTRL-__

The ASCII-to-EBCDIC and EBCDIC-to-ASCII translation tables are given with the descriptions of the ASCEBC and EBCASC subroutines in MTS Volume 3, System Subroutine Descriptions.

July 1988

| APPENDIX D: MERIT/UMNET RETURN CODES

| The return codes listed below are generated by the Merit/UMnet terminal routines. Return codes marked with an asterisk indicate an error condition; they will cause user programs not supplying the @ERRRTN I/O modifier to be interrupted.

Input	0	Successful return
	4	End-of-file read from network. This does not necessarily mean that there is no more data to be read from the network, only that the remote terminal or host has sent an end-of-file.
	8*	Read not allowed; must do a write. This means that the remote host is requesting input from the network connection and, to avoid a deadlock, the local program must not read from the network. The prompting characters, if any, sent by the remote host when it did the read are returned to the user.
	12*	Should not occur
	16*	Connection is closed: no I/O may be done
	20*	Should not occur
	24*	Attention interrupt received from MOUNTed network connection
	28*	Same as return code 8 except that the remote host has requested that the input area be blanked for "n" characters, where "n" is returned as a 2-digit decimal number followed by the prompting characters. A value of "00" means that no specific number of characters has been specified.
	44*	Read on a MOUNTed network connection was terminated by an attention interrupt from the user's terminal. No data is returned.
	48*	Read on a MOUNTed network connection was terminated because no data was received from the network by MTS within the time specified by the "timeout" network device command. No data was returned. (Note: This will not occur unless a "timeout" device command was issued since, by default, input operations are not timed.)
	Output	0
4*		Should not occur
8*		Write not allowed; must do a read. This means that the remote host has issued a write on the network connection and, to avoid a deadlock, the local program must not write on the network.
12*		Should not occur
16*		Connection is closed: no I/O may be done
20*		Should not occur
24*		Attention interrupt received from MOUNTed network connection

July 1988

CONTROL	0	Successful return
	4*	Should not occur
	8*	Control command not allowed--the remote host is attempting to send a record
	12*	Successful command with returned text
	16*	Connection is closed: no I/O may be done
	20*	Invalid syntax or context for control command
	24*	Attention interrupt received from network
	64*	Internal network error

July 1988

| APPENDIX E: CARRIAGE-CONTROL CHARACTERS

| MTS recognizes the following logical carriage-control characters. This recognition is under the control of the @CC FDname I/O modifier and the %CC device command.

blank	single space (skip 0 lines)
0	double space (skip 1 line)
-	triple space (skip 2 lines)
+	single space
&	no space after printing
9	single space
1	triple space (see also the %FORMFEED device command)
2	triple space
4	triple space
6	triple space
8	triple space

All other logical carriage-control characters are treated as a request to single-space and print the entire line (including the carriage-control character).

Lines written using the @MCC I/O modifier (machine carriage control) are always single-spaced and the machine carriage-control character is ignored.

| APPENDIX F: INTERNAL FORMAT OF THE ANSWERBACK

The following table gives the internal format of the UMnet answerback as returned by the GUINFO subroutine (see MTS Volume 3, System Subroutine Descriptions, for details).

Bytes 1-9 are fixed-format; the remaining portion of the answerback is variable-format.

<u>Byte</u>	<u>Information</u>
1-4	MTS device name
5	"-"
6-7	Service type (e.g., "TL" for terminal, "IT" for Merit host-to-host, "X3" for Telenet or Autonet)
8-9	Distant host name (e.g., "H " for Hermes)
Variable	Name of host that initiated the connection, delimited by semicolons - ;host name; (e.g., ";H@AN;" for Hermes on the AN PCP or ";H@UNY1;" for Hermes on the UNY1 SCP).
2 bytes	Terminal port type
2 bytes	Terminal port number
1 byte	;"
Variable	Terminal type (up to 24 bytes)

July 1988

| APPENDIX G: FDNAME MODIFIERS FOR MTS CONNECTIONS

| Listed below are those MTS FDname modifiers that are given special
| treatment by Merit/UMnet.

@CC This modifier is ignored for output by the local host and input by the remote host.

Input: The carriage-control character is removed from lines to the local host if and only if the @~CC modifier is specified and the line is marked as having carriage control. A blank carriage-control character is inserted if and only if the @CC modifier is specified and the line is marked as not having carriage control. The line is left unchanged if and only if neither modifier is specified or @CC is specified for a marked line or @~CC is specified for an unmarked line.

Output: Every line from the remote host is marked as having or not having a carriage-control character which is determined by the setting of the @CC modifier and the line itself as described above in the description of the CC Merit device command.

@BIN Input: If specified, no HEX, TAB, or UC input processing will be performed. If @PEEL is also specified, the line-number parameter will be set to zero, but no line-number peeling will be performed. The @BIN modifier implies @SP.

Output: The output record is not translated to ASCII, and the record is designated as not being translated so that the receiving host will not translate it. This modifier may be overridden by the BIN device command. The @BIN modifier implies @SP.

@SP Input: If specified, this modifier suppresses the sending of the prefix character(s) to the remote host. (The prefix is sent only when the Merit device is *MSOURCE*.)

Output: If specified, this modifier suppresses the sending of the prefix character(s). (The prefix is sent only when the Merit device is *MSOURCE*.)

| APPENDIX H: NETWORK ERROR MESSAGES

Terminal users using or attempting to use the network may see either Network Standard Error Messages or Nonstandard Error Messages when problems arise. The nonstandard messages are local-host dependent.

| For information or help in an error situation beyond that given in the error message, contact the Merit Central Office, (313) 764-9430.

Network Standard Error Messages:

All {UM|WU} interactive ports are busy

Try again to open a network connection. This is analogous to getting a busy signal when a telephone call is placed.

{UM|WU} host is down

Check the appropriate network news file for status information.

All network paths to {UM|WU} are busy (n)

Try again to make a network connection. If the second try produces the same result, please call the Merit office. "n" represents an error code that the Merit office would like to be told about.

No network path to {UM|WU}

Either the remote Communications Computer (CC) has crashed or all the telephone lines to it are down. This is probably a temporary condition; check the appropriate network news file.

Network system error (n)

Try again to open a network connection. If that does not work, please call the Merit Central Office and relay the number "n".

Nonstandard Error Messages: User outbound from UM or WSU

MNnn: Attempt to READ out of turn

User attempted to read out of turn, that is, at the same time that the remote host was trying to read. This message is the result of a user error.

MNnn: Connection is not OPEN

User attempted to do a read, write, or control command on a closed connection.

July 1988

OPEN FAILED: message

The "message" will be one of the five Network Standard Error Messages listed above.

Network down. Try again later

The Communications Computer is not available for some reason.

MNnn: Attempt to WRITE out of turn

User attempted to write a message to the network (or send an end-of-file) while the network was sending a record to the user. This message is the result of a user error. The user should check the program.

No response to previous attention

The previously transmitted attention interrupt has not been responded to by the remote host. Possible causes are that the remote host is down or that the user has pressed the attention key again too soon after the first time.

MNnn: Network failure. Try again later

If this message again received upon retry, the user should call the Merit Central Office.

MNnn: Invalid device command

This is a response to illegal syntax in a command to the local device support routine (DSR). See Appendix G.

MNnn: ATTENTION received from network

The remote host has issued an attention to the local host.

Invalid hex input

This message indicates invalid hexadecimal input (e.g., ABC "F" instead of ABC "F1").

|

July 1988

THE NET COMMAND

The NET command allows a user (or a user program) at one host computer to become an interactive user of a remote host, through a connection initiated at a local host. The user may obtain services from either one of the two hosts. All lines of text entered from the terminal are processed first by the local host, before being passed to the remote host. Terminal input is handled by the local host terminal support, so the user need not learn the terminal support conventions of the remote host.

The resources at the remote host, made available by the host-to-host network connection, are the same as those available through a terminal connected directly to that host or available through a terminal connected to that system through Hermes. In addition, input to and output from the remote host may be taken from or written to files and devices located at the local host, and files may be transferred between hosts. The network also allows interprocess communication between programs running at each host, thus making distributed processing between Merit hosts possible.

There are several reasons for using the NET command. One of the most common applications is to copy data files between hosts interactively. Another is to couple data files, tapes, or printers on one host to a program running on another host.

One of the more advanced uses of the host-to-host service is to interconnect two user-written programs. It is possible for programs to interchange information on such a connection. This means that true distributed processing may be performed over the network.

The network host-to-host interactive connection is most conveniently viewed as a device, e.g., like a magnetic tape; the connection must be mounted and then a command language subsystem or user program can be invoked which reads lines from the connection or writes lines to it. It is this property of a connection in MTS that allows the host-to-host service to be used to copy files, couple files to devices between hosts, or perform interprocess communication. It also is possible to combine the mounting step with the invocation of the command language subsystem. This is preferred by most users.

After signing on to an MTS host, the first step in the general case is to allocate a network device and specify its name to MTS. The command to do this is the same at both MTS hosts. The command is:

```
$MOUNT MNET *pdn* DEST=host
```

July 1988

where "*pdn*" is some alphanumeric name written between asterisks. It is the pseudodevice name of the network device for use in other MTS commands. The last parameter specifies the host at which service is requested. Currently, connections may be mounted from MTS to three of the Merit Network hosts. Each of these hosts has a two-character identifier:

```

      UM  University of Michigan
|      UB  University of Michigan
      WU  Wayne State University

```

| Thus, an MTS user on UM may access WSU by typing:

```
|      $MOUNT MNET *QQUU* DEST=WS
```

though a more mnemonic pseudodevice name may be given.

| The assignment of this device in MTS has the effect of opening an interactive connection to WSU through the network. There the incoming network connection appears to the WU host just like a local terminal that has dialed in directly. The WU host therefore immediately sends some identification lines to be printed on the user's terminal.

| How is the output from the WU host copied to the user's terminal on the local MTS host? After all, the network appears to MTS to be a device, and mounting a magnetic tape does not cause it to write its contents on the user's terminal. There must be a method available which copies lines from the network to the terminal, and vice versa. Since there is a particular sequence of commands to follow, it is inconvenient to use \$COPY in each direction for every line. Therefore, a special network command language subsystem (CLS) called \$NET, is available for this purpose.

\$NET will normally take its input from *SOURCE* and produces output on *SINK*, but it must be told the name of the network connection. This is the pseudodevice name given to the MOUNT command which opened the connection. The step after the mount is, therefore, to type:

```
$NET *QQUU*
```

Thereafter, lines received from the network are printed on the terminal; lines typed on the terminal are sent over the network. Pressing the Break or Interrupt button will cause an interrupt signal to be sent over the network. Typing "\$ENDFILE" sends an end-of-file signal over the network (unless the user has previously "\$SET ENDFILE=NEVER" or ".EOF=OFF").

The two-step process of mounting a network connection and invoking the NET CLS can be combined into one command of the form:

```
$NET host
```

July 1988

| where "host" is UM, UB, or WU. For example,

| \$NET WU

This latter method is the recommended and preferred way to start for most interactive procedures.

| After either of these initial steps the network intrudes as little as possible, and the user proceeds by executing the appropriate commands required by the remote system. The user first has to sign on, just as a terminal user does on any time-sharing system. Thereafter one may do anything the system allows a terminal user to do. The NET CLS copies lines until the user signs off or logs out at the remote host. This causes the connection to be closed, and, inside MTS, the network device is dismantled or deallocated. Thereafter the user is again a local MTS terminal user.

To return to the local MTS host without losing the connection, either of two commands may be given to NET instructing it to stop. These are:

```
.STOP
.MTS
```

Note the period at the beginning. A period is an indication that the rest of the line is not to be copied to the remote host; the line is a command to NET. A complete list of available commands for NET is given in Appendix A. Should it be necessary to send a text line that actually begins with a period, the user must type two periods. NET strips off the first and then sends the remainder as usual. The network command .MTS returns the user to the local MTS host and allows the user to reenter the remote host simply by issuing the MTS command:

```
$NET
```

The network command .STOP unloads the NET CLS, but does not close the connection. Should the user wish to reenter the remote host the MTS command must be typed:

```
$NET *pdn*
```

or

```
$NET host
```

where "*pdn*" is the pseudodevice name given to the mount command which opened the connection. The pseudodevice or host name need not be typed when the remote host was left by way of the .MTS network command.

EXAMPLES OF USING THE NET COMMAND

The following examples illustrate how to establish and use a host-to-host connection. They were obtained from an interactive terminal session. The following notation is used:

uppercase - indicates output from the computer.
lowercase - indicates input from the user.

In this example, a user signs on to MTS at WSU, opens a connection to the UM host, and signs on.

```

MTS (MN51-00053)
#sig wuid
#ENTER PASSWORD.
?wupw
#MERIT,NORMAL,CSC
#LAST SIGNON WAS AT 10:04:08, THU FEB 9/84
#USER WUID SIGNED ON AT 14:05:16, THU FEB 9/84
# $net um

#*UM*:      MOUNTED ON MN02
) MTS : ANN ARBOR (AN18-00137)
)#sig umid
)#ENTER PASSWORD.
)?umpw
)#TERMINAL,NORMAL,UNIV/GOV'T
)#LAST SIGNON WAS AT 11:58:53, THU FEB 9/84
)#USER UMID SIGNED ON AT 14:05:40, THU FEB 9/84
)#

```

The example ends with the remote host (UM MTS) prompting for a command using the "#" prefix.

Copying Files Over the Network Interactively

Interactive file transfers over the network are easy. The .COPY command is used after the necessary interactive connection is opened. The following examples illustrate the appropriate procedures. For more information on the .COPY command, see Appendix B. Merit User's Memo 9 provides complete documentation.

These examples show how to copy files between two MTS systems. Although these examples are presented from the point of view of a user who initiates a host-to-host connection from Wayne State, they are equally valid for a user who initiates a connection from UM. As usual, the user signs on to the local host and then makes a connection to the remote MTS host and signs on there, too. In each example below there is a file OLDFILE at one host to be copied to the file NEWFILE at the other.

July 1988

```
| (1) Copying a file from Wayne State to UM:

    $net um
    *UM*: MOUNTED ON MN01
    ) MTS : ANN ARBOR (AN43-00153)
    )#sig umid
    )#ENTER PASSWORD.
    )?umpw
    )#TERMINAL,NORMAL,UNIV/GOV'T
    )#LAST SIGNON WAS AT 10:31:35, WED FEB 08/84
    )#USER UMID SIGNED ON AT 13:27:02, THU FEB 09/84
    )#.copy lcl oldfile rmt newfile
    ) L: 23 RECORD(S) PROCESSED.
    )#
```

```
| (2) Copying a file from UM to Wayne State:

    $net um
    *UM*: MOUNTED ON MN01
    ) MTS : ANN ARBOR (AN43-00153)
    )#sig umid
    )#ENTER PASSWORD.
    )?umpw
    )#TERMINAL,NORMAL,UNIV
    )#LAST SIGNON WAS AT 10:31:35, WED 08/84
    )#USER UMID SIGNED ON AT 13:27:02, THU FEB 09/84
    )#.copy rmt oldfile lcl newfile
    ) L: 23 RECORD(S) PROCESSED.
    )#
```

The .COPY command automatically provides for copying object files and sequential files. This applies only to copying between UM and WSU.

When copying a file over the network from one host to another, users should keep in mind the time involved in making this transaction. Currently, the maximum rate of network file copying for a line file is approximately ten lines per second between UM and WSU and five lines per second between UM or WSU and MSU. Users should consider transferring long files using the network batch service described below in this section.

Closing a Connection to a Remote Host

An open connection is closed when one signs off or logs out from the remote host.

For an MTS user accessing a remote MTS system, the signoff looks like:

```
| )#sig
    )#UMID 14:04:40-14:11:51 THU FEB 09/84
    )#TERMINAL,NORMAL,UNIV/GOV'T
```

July 1988

```
)#ELAPSED TIME          7.183 MIN.          $.30
)#CPU TIME USED         .293 SEC.          $.11
)#CPU STOR VMI          .016 PAGE-MIN.
)#WAIT STOR VMI         .483 PAGE-HR.
)#PAGE-INS              26
)#DISK I/O              185

)# APPROX. COST OF THIS RUN:          $.41
)#DISK STORAGE CHARGE    9 PAGE-HR.    $.03
)# APPROX. REMAINING BALANCE:        $123.46
*CONNECTION CLOSED
*MN00 RELEASED.
#
```

The user is left communicating with the local MTS. Further network use requires opening a new connection.

Returning Temporarily to the Local Host

It is not necessary to close a connection to return to the local system. The NET command

```
.mts
```

leaves the connection (and the remote job) suspended while control returns to the local MTS. The \$NET command will restore remote communication:

```
#$net
```

July 1988

APPENDIX A: NET CLS COMMANDS IN MTS

NET commands may be given as soon as a connection is established to a remote host and the NET CLS is invoked. Command lines are distinguished by the presence of a "select character" at the beginning of the line. The default select character is a period "." but it may be changed to any other character with the .SELECT command (see below). To send a normal data line beginning with the current select character to the remote host, two consecutive select characters should be entered. The minimum abbreviations and defaults for these commands are underscored.

.ATTN={ON|OFF}

This command controls the processing of attention interrupts while the user is communicating with the remote host. The default (ON) permits attention interrupts to be propagated to the remote host. If OFF is specified, attention interrupts are not sent to the remote host; instead, the network subsystem reads an input line from the user's terminal. In both cases, the INPUT is set to *MSOURCE* and the OUTPUT to *MSINK*.

Example: .ATTN=OFF

.CKLENGTH={ON|OFF}

Normally, the NET CLS automatically sets OUTLEN (see Appendix C) equal to the maximum output length allowed by the current .OUTPUT device. If .CKLENGTH is set to OFF, this automatic setting of OUTLEN is not performed.

.CONTROL command

This command may be used to issue a device command to the connection.

Example: .CONTROL JOB

A user can perform the equivalent of a .CONTROL command from within MTS at the local host by issuing the MTS command "\$CONTROL *pdn* command".

.COPY {LCL|RMT} fromfile [TO] [{LCL|RMT}] tofile

This command copies data files between network hosts. It is fully documented in Appendix B.

Example: .COPY LCL FILE1 RMT FILE2

July 1988

.ECHO={ON|OFF}

This command controls the echoing of input lines from SOURCE to SINK. If ECHO is ON, input lines are printed at the user's SINK as well as being sent to the remote host. The default is ON.

Example: .ECHO=ON

.EOF={ON|OFF}

This command controls the processing of end-of-files. An end-of-file may be propagated to the remote host, or it may reset INPUT which sets INPUT to *SOURCE* or sets *SOURCE* to *MSOURCE*. When INPUT is set to *MSOURCE*, all end-of-files are propagated to the remote host, thus EOF is ignored.

With EOF=OFF and INPUT not set to *MSOURCE*, an end-of-file resets INPUT.

With EOF=ON and INPUT not set to *MSOURCE*, an end-of-file is propagated, unless it was immediately preceded by another end-of-file, in which case INPUT is reset. Thus two consecutive end-of-files cause one to be propagated and INPUT to be reset.

Example: .EOF=OFF

.INPUT FDname

This command instructs the NET CLS to read all records bound for the remote host from the file or device specified by "FDname" instead of *SOURCE*. This behavior continues until INPUT is reset by an attention interrupt or an end-of-file (see also the .ATTN and .EOF commands).

Example: .INPUT DATA

.MCMD command

The "command" portion of this network command is passed to MTS. Control is returned to NET upon completion of the MTS command.

Example: .MCMD \$DISPLAY COST

.MTS [command]

This command returns control to the local MTS host and allows an MTS command to be given as a parameter. It does not close the network connection to the remote host but only inactivates it. To later reactivate it, use the MTS command \$NET.

Example: .MTS

July 1988

.NET [{host|*pdn*}]

This command allows users to switch between different active host-to-host connections. The "host" parameter may be any valid two-character Merit host identifier.

Example: .NET WU

.NETPFX={ON|OFF|"string"}

If NETPFX is ON (the default), the net prefix string received from the remote host is prefixed by the value specified. The default string is "). The user may change the prefix to any other value. The net prefix string is associated with the pseudodevice that is active when .NETPFX was assigned.

Example: .NETPFX="wu) "

.OPER text

This command causes "text" to be printed on the operator console of the network's local Communications Computer, where it may or may not be immediately seen.

Example: .OPER I seem to lose data on this connection.

.OUTPUT FDname

This command may be used to route all records received from the remote host to any MTS file or device instead of *SINK* (default). If an end-of-file is received from the network connection, OUTPUT is changed to *SINK* and a message is printed. In general, use of this command has been replaced by the .COPY command (see Appendix B).

Example: .OUTPUT -F

.READ={ON|OFF|n}

This command may be used to specify the number of records to be read from the network. The default (ON) is equivalent to an infinite "n".

Example: .READ=3

.RECEIVE

This command allows a user to "wait" for a reverse attention to be transmitted from the remote host. Certain commands and programs can produce output after the NET CLS has already read from the user's terminal. In this case, the output will not appear at the terminal until further input is entered. The .RECEIVE command allows the read operation to be delayed until the user cancels the

wait by entering an attention at the terminal. Lines from the remote host are printed as they are received. No \$NET commands or data lines may be entered until the wait is cancelled.

Example: .RECEIVE

.REMOTE command

This command may be used to issue a device command to the MTS Merit routines at the remote host. (See Appendix A of the section "Merit/UMnet.")

Example: .REMOTE TABS=ON

A user signed on to a remote MTS host can perform the equivalent of .REMOTE by giving at the remote host the MTS command:

\$CONTROL *MSINK* command

.RESET

This command causes the NET CLS network control switches EOF, ATTN, and READ to be reset to their default values. It also resets INPUT and OUTPUT to *SOURCE* and *SINK* and resets the select character to ".". .RESET also resets NETPFX to ON, but it preserves any user-assigned change to the prefix character.

Example: .RESET

.RETURN

This command is a synonym for the .MTS command except that no MTS command parameters are allowed.

Example: .RETURN

.SELECT={x|ON|OFF}

With SELECT=ON, lines that begin with a single select character are not sent to the remote host, but are processed as \$NET commands. With SELECT=OFF, lines that begin with a single select character are sent to the remote host unchanged (the first select character is not removed).

SELECT is set to ON whenever INPUT is reset since with SELECT=OFF, the command .SELECT=ON is not effective. The select character may be changed to any other character.

Example: .SELECT=/

July 1988

.STOP

This command returns control to the local MTS host but does not close the network connection. With this command, in contrast to the .MTS command, *pdn* is no longer the active device; thus to reactivate the connection the MTS command "\$NET *pdn*" or "\$NET host" is required.

Example: .STOP

.\$command

A NET command line beginning with ".\$" is passed to MTS for execution. There must be no intervening blanks between the "\$" and the MTS command that follows it. After the MTS command has been executed, control returns to NET.

Example: .\$CREATE UMFIL

..text

This sends the line ".text" to the remote host.

July 1988

APPENDIX B: .COPY--THE MERIT INTERACTIVE FILE TRANSFER FACILITYIntroduction

The .COPY command invokes the Merit Network file-copying service, which allows users to copy files in either direction between the UM, UB, and WU hosts. The .COPY facility permits the transfer of lines of up to 32,767 characters in length. It also provides the ability to interrupt and resume the copy process without loss of data.

The .COPY process is completely transparent, even allowing object modules to be transferred back and forth without special handling.

Although the .COPY service is intended to be used over an interactive host-to-host connection, it can also be invoked from a batch job that has opened a host-to-host connection. Users who do the latter should be warned that any errors in a .COPY process invoked through a batch job will cause the entire job to abort, or worse. The usual procedure for batch file-copying does not use the .COPY command; see "Batch File Copying" in the section "Merit/UMnet" for details. There are times when using a batch job to copy files between hosts has definite advantages, particularly if the file to be copied is large, in which case the batch method is likely to be considerably faster and cheaper. Merit staff members will be glad to help users determine the best method for a particular application. See Merit User's Memo 9 for complete documentation of .COPY.

.COPY and the NET CLS

.COPY is one of the commands belonging to the NET Command Language Subsystem (the NET CLS) and therefore can be used only when the NET CLS has already been invoked.

Users need only access their local host in whatever way they choose, sign on, and then issue the command

```
$NET xx
```

where "xx" is the two-letter Merit identifier of the other host to be involved in the file transfer (UM, UB, or WU). Then they must sign or log on to the other host. This will establish a host-to-host connection.

By definition, the end of a host-to-host connection at which the \$NET command was issued is the local end of the connection. The end that is reached with the \$NET command is the remote end. One can always tell which end of the connection is being addressed because the network, by default, prefixes each line of input to or output from a remote host with the net prefix. By default, this is a right parenthesis ")". To

July 1988

perform an interhost file-copy, the user must be addressing the remote host while it is in system command mode. That is, the user must see the following at the terminal before the .COPY command is issued:

```
|      )#
```

Invoking the .COPY Process

The file-copying is performed by issuing a single command. The command must be preceded by a period, which must be the first character on an input line; the period is what indicates to \$NET that what follows is a command to the NET CLS. The command is of the form:

```
.COPY {LCL|RMT} fromfile [TO] [{LCL|RMT}] tofile
```

where:

"fromfile" is the name of the single file or device (such as tape or card reader) from which information is to be copied. It may include a line range as well as modifiers.

"tofile" is the name of the single file or device (such as tape or *PRINT*) to which the "fromfile" information is to be copied. It may include a line range as well as modifiers.

"LCL and "RMT" are location designators denoting "local" and "remote", respectively. They tell .COPY at which end of the connection to look for the files or devices whose names they precede.

If the LCL or RMT designator is omitted from the "tofile", it will be assumed as the opposite of the "fromfile" designator.

For example:

```
|      .COPY LCL UMFIL E WSUFIL E=AF
```

This command will copy file UMFIL E, which exists at the local end of the connection--the one where the \$NET command was issued--into a file called WSUFIL E, which is a file that exists at the remote end of the connection.

```
.COPY RMT DATA1 LCL *PRINT*
```

This command will copy the file DATA1, which exists at the remote end of the connection, to a printer at whichever MTS host happens to be the local end of the connection. Notice that the word TO is omitted; .COPY always assumes that the first file named is the "fromfile" and that the second file is the "tofile".

July 1988

.COPY RMT PHYLE PHYLE

If the LCL or RMT designator for the "tofile" is omitted, .COPY will assume that the "tofile" is at the opposite end of the connection from the "fromfile". This command will then copy the contents of file PHYLE, which is at the remote end of the connection, into a file that is also called PHYLE which is at the local end of the connection.

.COPY in Operation

Once the copy process has been invoked, the user will see nothing at the terminal until .COPY prints out either the number of records transmitted or an error message. All messages from .COPY are prefixed with either an "L:" or an "R:" to indicate whether the local or remote end of the connection, respectively, originated it. The message

```
L:  n record(s) processed.
```

is the indication of a successful file-copy and always comes from the local end of the connection.

Error messages from .COPY will be generated if the "fromfile" or "tofile" does not exist; in this case a replacement filename must be given. Warning messages will also be generated if any lines are truncated during the copy process or if any incompatible characters are encountered.

Line Truncation

Truncation will occur when lines from the "fromfile" are longer than can be stored in the "tofile" (for example, if the user tries to copy 100-character lines to MTS *PUNCH*, which has an 80-character line limit). The user will be notified when the first line is truncated while copying. At the end of the transfer, the number of truncated lines is printed, if any. For example:

```
)OK-.copy rmt whatzit *punch*
)L:  File line truncation occurring.
)L:  27 record(s) processed.
)L:  5 truncated.
```

Interrupting and Resuming the Copy

The user may interrupt the copy process for a status check or termination at any time. When the copy is interrupted, the number of records that have been processed up to that point is printed at the terminal. This is the number of records received by a local host or the number of records transmitted to a remote host. After this intermediate statistic is printed, the user will be prompted for either the continuation or the termination of the transfer. If

July 1988

the user chooses to continue, the process will start up from where it left off without loss or duplication of data. For example:

```

#.copy longfile lcl -temp
!      [pressed an attention-interrupt key]
)L: .COPY attn!
)L: 63 record(s) processed.
)L: OK to continue (Y or N)? y
)L: 128 record(s) processed.

```

Notes and Tips

File Concatenation: The file-copy process does not allow the MTS operations of either explicit or implicit concatenation. If the user tries to use explicit concatenation, an error comment will be generated along with a prompt for a replacement filename:

```

) #.copy lcl progfile+datafile to rmt runfile
)L: Refer to "PROGFILE+DATAFILE"
)L: File concatenation not allowed.
)L: Enter replacement name or "CANCEL"?

```

If the user tries to use implicit concatenation--for example, specifying as "fromfile" a file that contains the line

```
$CONTINUE WITH NEXTFROMFILE
```

the line will be copied to "tofile" like any other data line and otherwise ignored, because .COPY always copies @-IC and this switch cannot be overridden. If the user tries to specify "@IC" on the fromfile, .COPY will simply ignore it. Most other MTS FDname modifiers (such as @I or @-TRIM) as well as line ranges may be specified, however, and will work.

Economical File-Copying: Because of the way the network handles interactive transmission, it turns out always to be more economical (both faster and cheaper) to copy a file from the remote host to the local host. That is, the .COPY command should look like

```
.copy rmt fromfile to lcl tofile
```

This means that if there is a choice, the user should prefer to invoke the NET CLS at the host to which he wants to copy.

|

July 1988

SCREEN-SUPPORT ROUTINESINTRODUCTION

The Screen-Support Routines are a set of system subroutines which allow program control of the user's terminal screen for applications such as \$EDIT visual mode, forms entry, or menu-driven software. The subroutines allow a programmer to send an entire screen's worth of information out to a video terminal at once. While the screen is displayed, the user can read the screen contents and respond by moving the cursor around to make entries or alterations in any desired location. No transmission to MTS occurs until the user presses the attention key or a program-function key (PF-key). The screen changes made interactively by the user are then returned to MTS and made available through these subroutines.

The formats of the screens defined using these routines can be extremely varied. Routines are provided to create, delete, and specify screens as well as fields within those screens. Fields may begin at any location and have any length as long as they do not extend off the lower right-hand corner of the screen. Fields may have different attributes; for example, some fields of text may be defined so that they can be modified by the user while others are "protected" to serve as data or prompts which may not be overwritten. The routines are quite low level, with all of the details of field size and location, initial cursor location, and field attributes set by individual calls to the Screen-Support Routines. If a screen is modified by typing at the keyboard, the changes must be retrieved explicitly by a series of appropriate routine calls in order for the calling program to store it or use it to update the next screen.

After the program defines and initializes the fields in a screen, a pair of calls to the screen write and read subroutines writes a screen out to the user's terminal. The program then waits for the user to signal for the return of changes made to the screen. This occurs when the user presses any attention-generating key such as a program-function key. At that point, any screen modifications which are stored internally by the read operation, can also be identified by name of field and placed in a buffer where the program will have access to them. The calling program can also obtain other information such as the cursor position on return or the total number of fields modified. The identity of the particular PF-key used to terminate the read is returned, and can be used to take action appropriately (for example, it could determine which of several screens will be written next). Since visual mode of the MTS File Editor is one of the most generally used applications for these routines, a description explaining how its screen routines operate appears in Appendix B.

Output Devices

These routines make no assumptions about the type of terminal, or microcomputer performing terminal emulation, that is being controlled. The application must take into account screen size and other relevant device features. A subroutine is provided which allows the user's program to obtain this information.

A program using the Screen-Support Routines should be designed with each particular output device that may be used in mind, since many aspects of the program will vary with the size of the screen, the use of particular PF-keys to transmit screen modifications to the user's program, or interrupt handling on different devices. The routines are intended for use with any video terminal, but currently (due to front-end hardware limitations) only the Ontel terminal, IBM 3278 Display Station, and some microcomputer terminal-emulation programs such as WINDOW for the IBM-PC are supported. Many of the device-dependent features mentioned apply to either the Ontel or the IBM 3278. An explanation of such features appears in the particular context in which they are applicable. The SSINFO subroutine provides device-specific information. Programs which are sufficiently flexible can be written to operate on a variety of devices.

BASIC USE OF THE SCREEN-SUPPORT ROUTINES

The following section gives the details of what the programmer must do to transmit and receive information via fields on a terminal screen. It describes various stages of the process with examples. Appendix A describes each of the subroutine calls individually, giving its calling sequence, parameters, and return codes. Appendix C contains a small sample program in Pascal/VS, and Appendix D contains a different VS Fortran example.

Running a Program

The Screen-Support Routines consist of a set of subroutines which the programmer may access from Pascal, Plus, Fortran, Assembler, or any other programming language which provides for standard S-type calling sequences (see MTS Volume 3, System Subroutine Descriptions, for more information on calling sequences). They are resident in the system, so there is no need to load anything in order to use them. Calling them from a language which handles character data well makes text-string manipulation easier. The examples provided illustrate calls from Pascal, and the appendices contain sample programs in Pascal and VS Fortran. For information on the procedure to follow in a particular case, see the section on external routines in the appropriate MTS volume for that language.

July 1988

Initialization

A call to the routine

```
CALL SSINIT(sscb,clstv)
```

must be issued first to initialize the screen-support routines. "sscb" is the screen-support routines control block, which must appear as a parameter in all subroutine calls. For example, in Pascal the initialization might look like this:

```
{ Initialize the SS routines. }
RC := SSINIT(SSCB, 0);
```

The variable RC is defined throughout these examples in order to access the Screen-Support Routines' return codes, which are otherwise not directly available in Pascal/VS. Once initialized, "sscb" must not be changed for the duration of the program, since it serves as the pointer to all of the screen storage. In FORTRAN, if subroutines are used that call these routines, then SSCB should be placed in a COMMON block. "clstv" stands for "command language subsystem transfer vector". If set to 0, it defaults to the address of the standard MTS transfer vector so that the screen-support routines will call the normal versions of all system subroutines, such as GETSPACE. The "clstv" parameter is available so that systems programs using the screen-support routines can optionally perform some system subroutines themselves. Ordinarily users will always supply 0 as the value for "clstv".

Initializing, Defining, and Ending a Screen

The process of defining any individual screen consists of naming the screen and defining the fields to appear within it. This section describes that process in more detail. Note that these routines simply determine the contents and characteristics of a screen; both the SSWRIT and SSREAD subroutines must be used to actually display the screen on an output device and return user modifications to it.

The SSBGNS subroutine initializes and names a screen, which then becomes the active screen. The call

```
CALL SSBGNS(sscb,screen)
```

either creates a new screen or specifies which existing screen to redefine. For example, here is a fragment of the Pascal program in Appendix C:

```
{ Define a screen with the name 'SCREEN'. }
RC := SSBGNS(SSCB, 'SCREEN');
```

July 1988

In the case of a previously defined screen, a call to SSBGNS will empty existing screen definitions; the screen continuation routine

```
SSCNTS(sscb,screen)
```

should be used to reactivate and amend a previously defined screen without deleting all existing contents. "screen" ceases to be the active screen if another call to SSBGNS is made, or if SSENDS terminates the screen definition process. Multiple 'begins' and 'ends' for a series of screens may be nested logically within the program, but only one screen is active at any point. If no screen is currently active when a field definition routine is called, an error will be signalled.

When the programmer has finished initializing and defining all the fields for a given screen, the program should call:

```
SSENDS(sscb,screen)
```

to end the process of defining that screen. "screen" will then become inactive, but can be written out whenever desired.

Defining Fields

For a screen to be useful, it must be defined as a series of fields which can contain, for example, a string of text or an area where an entry is to be typed. This can be accomplished in several ways. One is by issuing a call to SSDEFF to define multiple aspects of a field at once. The program may also make separate calls to SSCREF, SSLOCN, and SSTEXT. SSCREF creates a field without determining its location and text. Since SSCREF creates a field without specifying further characteristics, separate calls to other subroutines such as SSLOCN assign the other values if it is used. SSLOCN sets or alters an existing field's location, and SSTEXT sets or replaces its contents.

Both the SSCREF and SSDEFF subroutines require the programmer to assign a field name specifying the name of the field to create. Field names must be a string exactly eight characters long; shorter names must be padded with blanks. Field names must be unique within a screen but the same field names may be used in different screen definitions.

Text to be written out in the field can be included as a string or variable in SSDEFF, but if a call to SSDEFF creates a field but no text should appear, a null string and text length of 0 should be provided as parameters. The SSTEXT subroutine then can be used separately to specify the desired text and length.

The programmer can also set controls for other attributes of a field, such as character intensity or write-protection, by calling the SSATTR subroutine. By default, fields have all attributes set off (i.e., equal to 0). All these commands apply to the screen currently active when they are issued.

July 1988

In the following example, a call to SSDEFF creates a field, named A, that is 4 characters long with the string "TEXT" as its contents. It will appear in the uppermost left-hand corner of the screen, at coordinates (0,0), when the screen is written out. The text length is also 4.

```
{ Define a field with the name 'A'. }
RC := SSDEFF(SSCB, 'A', 0, 0, 4, 'TEXT', 4);
```

An error will result if the field already exists when SSDEFF or SSCREF is called.

Specifying Field Locations

Field locations may be assigned on the initial call to SSDEFF or assigned or redefined by means of a separate call to the subroutine

```
SSLOCN(sscb,field,row,column,length)
```

Thus, SSLOCN can be used to move fields without defining a new screen each time. For example, this could be used when "windowing" through a continuous set of data which will not all fit on one screen at the same time. It can only be called for an existing field.

The location of a field is designated by a combination of its starting location and its length. A pair of values for row and column identify the starting location. Possible values for the row and column parameters each start at 0. Thus, the upper-left corner of the screen is position (0,0). They range to the maximum permissible on the output device being used.

The programmer must assign a length to each field. If the length of a field is greater than whatever number of columns a given device supports, the field will "wrap around" into subsequent rows, as long as the field does not extend off the lower right-hand margin of the screen. If that occurs, an error will be generated.

Writing the Text of a Field

If the programmer wishes to include a text string in a field, the "text" and "length" parameters must be specified. This can be done when a call to SSDEFF first defines the field, or

```
SSTEXT(sscb,field,text,length)
```

may be called for an existing field. If a user modifies a field, and the same screen is rewritten, SSTEXT must be used to replace the

July 1988

original text with the modified field contents. Otherwise, the changes will disappear; on rewriting the screen, the routines will produce the original field contents over again. This might of course be what the programmer desires for that particular application, but in many cases the fields which are modified will need to be 'updated' by SSTEXT.

The text string may be any length, but the actual number of characters must correspond to the number specified as the value for "length", or untoward results may occur. The length of the text string should also be less than or equal to the length allocated for the field. If the field length is shorter the text will be truncated and an error will be returned; however, the field will still be defined. If the field length exceeds the text string length, the text will be left-justified within the field. Although the subroutine will not pad it with any fill character, that section of the screen will appear empty.

The following example is a call to SSTEXT which occurs in the example program in Appendix C. There, (after calls to SSWRIT and SSREAD) it serves to replace the text of the field with the modified text entered by the user and obtained previously by the program, using other calls to the Screen-Support Routines. When the screen is rewritten, the user's changes will appear rather than the original entry. In this call, RC can be used to check to make sure that the modified field did not cause any truncation errors, and that the field exists and the screen is active.

```
RC := SSTEXT(SSCB, 'A', V3.TEXT_BUFFER, TEXT_LENGTH);
```

Since field locations consist of a starting row or column and a length, an unintentional overlap could occur. This will only generate an error condition if the CHECK facility is set on. While initially setting up a screen it is helpful to use the SSCTRL subroutine, with the CHECK parameter set = 1, to check for overlapping fields. Since the checking process expends considerable CPU time, once the programmer has set locations and lengths for all the fields for a screen the CHECK facility should be disabled.

Existing screens and fields may be modified or deleted at any time that the screen is active, or the SSCNTS subroutine may be used to reactivate and modify a screen without emptying its definitions.

Setting Field Attributes

The subroutine

```
SSATTR(sscb,field,attr,value)
```

allows the programmer to specify certain field attributes. Many of them are device dependent; a call which sets an attribute that the device in use does not support will be ignored. No error will be returned.

July 1988

The legal attributes are:

<u>Attribute</u>	<u>Value</u>	<u>Device</u>
'AUTO' - Autoskip	0/1 (off/on)	IBM 3278 only
'BLAN' - Blank	0/1 (off/on)	--any--
'HIGH' - Highlight	0/1 (off/on)	--any-- (same as REVE on Ontel)
'NUME' - Numeric	0/1 (off/on)	IBM 3278 only
'PROT' - Protected	0/1 (off/on)	--any--
'REVE' - Reverse	0/1 (off/on)	not IBM 3278

"Autoskip" will automatically move the cursor to the next unprotected field when the user comes to the end of the current one. "Blank" disables character echoing on the screen so nothing appears in the field as an entry is typed. The MTS blanking for passwords provides an example of this effect. "Highlight" brightens the video in a field. The visual editor on the IBM 3278 uses this to intensify the line number to indicate a field which contains nonprinting characters. "Numeric" will set a field to accept only number input. Both autoskip and numeric emulate features common on a keypunch and facilitate data entry and error checking. "Protected" prevents modification of a field if the user types over it, but protected fields are not displayed in any special manner. "Reverse" provides reverse video, which visual editor mode uses for line numbers on the Ontel to indicate the presence of nonprinting characters. Note that a call to SSATTR sets only one attribute. To assign multiple attributes to a field, separate calls to SSATTR must be issued.

Cursor Position

The position of the cursor within the currently active screen may be set with a call to

```
SSCURS(sscb,row,column)
```

which places the cursor at the location specified by "row" and "column". The cursor will appear at that position whenever the screen is written by SSWRIT, until the cursor location is explicitly reset by another call to SSCURS. A position which is off the screen is illegal.

Deleting a Field or Screen

A call to

```
SSDELF(sscb,field)
```

deletes "field" from the currently active screen definition. Similarly, the routine

```
SSDELS(sscb,screen)
```

deletes an entire screen definition.

Ending the Screen Definition Process

Once the programmer has finished work on the currently active screen, the subroutine:

```
SSENDS(sscb,screen)
```

retires it from active status. If a new screen is begun by calling SSBGNS or continued by calling SSCNTS without calling SSENDS for the active screen, the new screen becomes the active screen and the previously activated screen is placed on a stack. Successive calls to SSENDS terminate the currently active screen and remove each previously activated screen from the stack in turn, making them active again.

Writing and Reading Screens

Once the programmer has initialized a screen and defined its fields, the screen can be displayed on the terminal by calling SSWRIT and SSREAD. Any modifications made to the fields of the screen by the user are placed in the screen-support routines control block "sscb" by the SSREAD subroutine. Then, to actually access or locate the modified fields the programmer must call SSINFO, using the options MFTEXT, MFTXTLEN, or MFLENLOC. MFTXTLEN provides length information which allows MFTEXT to place the text of the modified fields in an appropriately sized buffer.

Writing a screen to the output terminal requires only the specification of the "sscb", screen name, and TERM for the output device, on a call to the subroutine:

```
SSWRIT(sscb,screen,device)
```

"screen" then appears on the user's terminal. If "screen" does not exist, an error will be returned. At present, "device" must be TERM. The following is an example of a call to SSWRIT from a Pascal program:

```
{ Display the screen. }
RC := SSWRIT(SSCB, 'SCREEN', 'TERM');
```

July 1988

The subroutine which reads back user changes to the displayed screen is:

```
SSREAD(sscb,device,aid,row,column,modf,field,offset).
```

An error will occur if there is no displayed screen or if "device" is not TERM as described above for SSWRIT. The user supplies the above parameters, and SSREAD returns a variety of data about the condition of the screen upon termination of the read operation. An identification (or "aid") number indicates which key terminated the read, as follows:

<u>Value</u>	<u>Key on 3278</u>	<u>Key on Ontel</u>
-4	CLEAR	No Equivalent
-3	SYS REQ	No Equivalent
-2	PA2	EOF
-1	PA1	ATTN
0	ENTER	PFA0

For PF-keys, the number of the key (e.g., 10 for PF10) corresponds to the "aid" number, for however many PF-keys the device supports. This information may be obtained by calling SSINFO to return a value for MFAIDCUR.

"row" and "column" give the cursor location when the read operation was terminated. "modf" gives the number of fields modified. "field" provides the name of the field holding the cursor at that point, and "offset" indicates the offset of the cursor within that field. For example, this allows the programmer to determine where the cursor was left when a read operation was terminated and replace it in that location when a fresh screen is written, so modifications to that area may continue conveniently. An error will result if the cursor is not in any defined field. For example:

```
{ Read back changes, etc. }
RC := SSREAD(SSCB, 'TERM', AID, ROW, COLUMN, COUNT,
             FIELD_ID, OFFSET);
Writeln(' AID: ', AID:2, ' Row: ', ROW:2, ' Column: ',
        COLUMN:2, ' Count: ', COUNT:2);
```

Much of the useful information provided by the screen subroutines is obtained by using the subroutine SSINFO. For example, a list of the names of the fields which were modified can be obtained by calling SSINFO with the parameter MFNAMES. SSINFO must also be called to obtain information that can be used to create a buffer so that the programmer can gain access to the modified text which the SSREAD subroutine stores in the "sscb". Three options provide information on particular fields of the designated screen. MFTXTLEN returns the length of the modified text for a field. This allows the programmer to create a buffer of the appropriate length, and MFTEXT then places the modified text there. MFLENLOC returns both the length and location of the modified text. Note the two-step sequence of calls described above, as it appears in the following Pascal program fragment from Appendix C. Also, V4 is

July 1988

required because Pascal will not accept a variable number of parameters, even though it is not used in this case.

```

if COUNT > 0 then begin
  V1.TEXT_SCREEN := 'SCREEN';
  V2.TEXT_FIELD := 'A';
  { Get modified text length for field 'A'. }
  RC := SSINFO(SSCB, 'MFTXTLEN', V1, V2, V3, V4);
  TEXT_LENGTH := V3.TXTLEN_LENGTH;
  RC := SSINFO(SSCB, 'MFTEXT', V1, V2, V3, V4);

```

Information, Error Checking, and Attention Handling

The SSCTRL and SSINFO subroutines provide a variety of facilities useful to the programmer. SSCTRL can be used to set up the optional error-detection facility which indicates if any two fields overlap, as described earlier. SSCTRL provides options that completely erase the screen before it is next rewritten. In addition for IBM 3278-type devices, SSCTRL sets up an attention trap; this last feature has no effect on other devices such as the Ontel. The field-overlap check feature causes the field definition routines to compare each pair of fields to determine if any overlap exists. It need only be invoked while the program initially is being debugged, by setting the CHECK parameter to 1 on a call to SSCTRL. This option is very expensive to include in normal program execution.

SSINFO, when called with different parameters, obtains the characteristics of the display terminal in use or returns information about any fields which the screen user modified. The SCREEN parameter returns the number of rows and columns for the terminal being used, the number of program-function keys, and the size of the FAC (Field Attribute Character). The FAC is a nonprinting character which appears at the start of each field and contains bits set to encode information such as the field's protection status. On the IBM 3278, it takes up one character position per field; on the Ontel, the FAC does not take up any space on the screen. Thus, to make a program work on both types of terminals, it must take the different FAC sizes into account in assigning a field length and text length for every field on the screen. As a result, the program may result in Ontel fields having one additional space, so that the same text will fit if an IBM 3278 is used. The VS Fortran program example in Appendix D illustrates one possible way to do this.

The following Pascal example of a call to SSINFO shows a way to obtain the FAC size along with other information the program can use to respond correctly on different devices:

```

{ Get some useful info about the device. }
RC := SSINFO(SSCB, 'SCREEN', V1, V2, V3, V4);
{ FAC size is V3.FACSIZE. }

```

July 1988

Another device-dependent option `NONPRINT` scans a given text string and sets a flag if any characters which are nonprinting on that device appear in the string.

Three additional `SSINFO` options are `MFAIDCUR`, `MFCOUNT`, and `MFNAMES`, which all return general information about a screen write/read operation. `MFAIDCUR` indicates the identification number for the attention ID (or "aid") number terminating the read (also device-dependent; see the description of `SSREAD` in Appendix A for details) and the cursor row and column. `MFCOUNT` gives the number of fields modified by the user, and `MFNAMES` returns the names of all modified fields. To update the above information when the next call to `SSREAD` occurs for that screen, the entire process of calling `SSINFO` with those parameters must be repeated.

Finally, three `SSINFO` options describe aspects of particular fields on the designated screen. The use of these is described more fully above. `MFTXTLEN` returns the length of the modified text for a field. This allows the programmer to use `MFTEXT` with a buffer of the appropriate length. `MFTEXT` then places the modified text there. `MFLNLOC` returns both the length and location of the modified text.

Termination

A call to

```
SSTERM(sscb)
```

terminates the use of the screen-support routines, and releases the storage space allocated for them.

July 1988

APPENDIX A: SCREEN-SUPPORT ROUTINE DESCRIPTIONS

The following pages contains the descriptions of each of the calling sequences for the Screen-Support Routines. All parameters are fullword (4 bytes) except where otherwise noted. Optional parameters are indicated with brackets. Error codes are returned in general registers R0 and R15.

July 1988

SSATTR

Subroutine Description

Purpose: To specify the attributes of a field.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSATTR(sscb,field,attr,value)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
field is an eight-character string specifying the name of the field for which attributes are to be set.
attr is a four-character string specifying what attribute to set.
value is the value to use when setting the attribute.

Returned Values:

R0 return value followed by R15 value in parenthesis.
 0 (0) Successful return.
 1 (4) Illegal value.
 2 (8) Illegal attribute.
 3 (12) Field does not exist.
 4 (16) Screen does not exist.

Description: The legal attributes and their values are:

<u>Attribute</u>	<u>Value</u>	<u>Device</u>
'AUTO' - Autoskip	0/1 (off/on)	IBM 3278 only
'BLAN' - Blank	0/1 (off/on)	--any--
'HIGH' - Highlight	0/1 (off/on)	--any-- (same as REVE on Ontel)
'NUME' - Numeric	0/1 (off/on)	IBM 3278 only
'PROT' - Protected	0/1 (off/on)	--any--
'REVE' - Reverse	0/1 (off/on)	not IBM 3278

The default values for all attributes are 0. Note that the setting of these attributes is device-dependent. If a

July 1988

particular device does not support an attribute it will be ignored. Due to the way this routine is implemented only one attribute can be specified per call. Multiple attributes must be specified with multiple calls.

July 1988

SSBGNS

Subroutine Description

Purpose: To begin the definition of or to redefine a screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSBGNS(sscb,screen)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

screen is an eight-character string specifying the name of the screen to begin.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.

Description: A screen is created if it does not already exist; otherwise, its definition is emptied.

SSCREF

Subroutine Description

Purpose: To create a field within a screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSCREF(sscb,field)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

field is an eight-character string specifying the name of the field to be created.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (4) Field already exists.
- 2 (8) No screen is currently active.

July 1988

SSCTNS

Subroutine Description

Purpose: To continue the definition of a screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSCTNS(sscb,screen)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

screen is an eight-character string specifying the name of the screen to continue.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.

1 (4) Screen does not exist.

SSCTRL

Subroutine Description

Purpose: To set information for a device or the
Location: Resident System device-independent portion of the screen-support routines.

Calling Sequence:

FORTRAN: CALL SSCTRL(sscb,what[,pars])

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
what is a character string specifying what information is to be set. A description of legal values for "what" is given below.
pars are the parameters necessary for what. A description of the "pars" for each "what" is given below.

Returned Values:

R0 return value followed by R15 value in parenthesis.
0 (0) Successful return.
1 (4) Illegal "what" value.
2 (8) Illegal "pars" value.

Description: The information for "what" is sent to the appropriate device or the device-independent portion of the screen routines along with its accompanying "pars". The legal values for "what" and their associated "pars" are:

'ERASE'

Specifying ERASE will cause the screen to be erased before it is written the next time. This will only occur once; any subsequent screen writes will not erase the screen unless the particular device requires it (e.g., the IBM 3278).

July 1988

`'CHECK',value`

"value" is either 0 or 1 and is used to control whether fields are checked for overlap when they are defined. If "value" is 1, fields are checked. If 0, they are not. Overlap checking is useful in the initial stages of screen construction. Each field is checked when it is defined to make sure that it does not overlap an existing field. Once it is determined that no fields overlap and the screen layout will not change, checking can be disabled. Overlap checking is fairly time consuming as each field must be checked against "all" others when it is defined. By default, overlap checking is disabled.

`'SETATTN',value`

"value" is either 0 or 1 and is used to control whether an attention trap is set on all subsequent read operations. If "value" is 1, attention traps are set up. If 0, they are not. Attention traps are used only by the IBM 3278 Device-Dependent Routine; this call will have no effect on the Ontel. By default, attention traps are always set up for IBM 3278-type devices. The trap is actually set up when the screen is written (SSWRIT). A wait for an attention is performed before the changes are read (SSREAD).

SSCURS

Subroutine Description

Purpose: To set the cursor position for the screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSCURS(sscb,row,column)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

row is the row in which the cursor is to be placed. Row numbers start at 0.

column is the column in which the cursor is to be placed. Column numbers start at 0.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.

1 (4) No screen currently active.

2 (8) Illegal cursor position.

Description: The cursor position is set for the currently active screen.

July 1988

SSDEFF

Subroutine Description

Purpose: To create a field and set its location and text.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSDEFF(sscb,field,row,column,flength,text,
tlength)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
field is an eight-character string specifying the name of the field to create.
row is the row the field is to start in.
column is the column the field is to start in.
flength is the length of the field.
text is the text for the field.
tlength is the length of the text.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.
 1 (4) Field already exists.
 2 (8) No screen is currently active.
 3 (12) Field will overlap another field (only returned if CHECK = 1).
 4 (16) Field will go off screen.
 5 (20) Field length <= 0.
 6 (24) New field length will cause text to be truncated.

Description: The field name given by "field" in the currently active screen is created with the given location and text information. This routine is functionally the same as separate calls to SSCREF, SSLOCN, and SSTEXT. flength does not include the FAC length.

SSDELFL

Subroutine Description

Purpose: To delete a field from a screen definition.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSDELFL(sscb,field)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

field is an eight-character string specifying the name of the field to delete.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (4) Field does not exist.
- 2 (8) No screen is currently active.

July 1988

SSDELS

Subroutine Description

Purpose: To delete the definition of a screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSDELS(sscb,screen)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

screen is an eight-character string specifying the name of the screen to end.

Returned Values:

0 (0) Successful return.
1 (4) Screen does not exist.

SSENDS

Subroutine Description

Purpose: To end the definition of a screen.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSENDS(sscb,screen)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

screen is an eight-character string specifying the name of the screen to end.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (4) Screen is not currently active.
- 2 (8) Screen does not exist.

July 1988

SSINFO

Subroutine Description

Purpose: To get information about the device in use or about characteristics of an already defined or modified screen.

Location: Resident System

Calling Sequence:

FORTTRAN: CALL SSINFO(sscb,what[,pars,...])

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
what is a character string specifying what information is desired. A description of legal values for "what" is given below.
pars are the parameters necessary for what. A description of the "pars" for each "what" is given below.

Returned Values:

R0 return value followed by R15 value in parenthesis.
 0 (0) Successful return.
 1 (4) Illegal "what" value.
 2 (8) Illegal "pars" value.

Description: If SCREEN appears as the "what" value in SSINFO, the program will return information about the device in use. NONPRINT sets a flag that indicates whether any characters will not print on the device. The following three values for "what" return information about a modified screen when the programmer specifies the name of the screen as a variable name or eight character string. The last three "what"s return information about the modified text for a particular field when the programmer passes it the parameters "screen" and "field". The information returned for that "what" is placed in the appropriate accompanying "pars". The legal values for "what" and their associated "pars" are:

July 1988

'SCREEN', rows, columns, facsize, pfkeys

The number of rows (lines) on the device's screen is placed in "rows", the number of columns in a row in "columns", the size of the FAC (Field Attribute Character) in characters in "facsize", and the number of program-function keys in "pfkeys". The size of the FAC is a hardware-dependent feature. It is a nonprinting character which appears at the start of each field and encodes information about the field attributes.

'NONPRINT', text, length, flag

The text string "text" of length "length" is checked for nonprinting characters. If any are present, "flag" is set TRUE; if all characters are printable "flag" is set FALSE. This information is device dependent.

'MFAIDCUR', screen, aid, row, column

For "screen", the "aid" ("Attention ID" - an integer number representing the character terminating the read; see the description for SSREAD for the AID values indicating which PFkey was used), and the cursor location ("row" and "column") are returned.

'MFCOUNT', screen, count

The count of modified fields for "screen" is placed in "count".

'MFNAMES', screen, count, vector

The names of all the modified fields of screen "screen" are placed contiguously (8 bytes each) in "vector". The size of vector must be at least the number of modified fields times 8 bytes. The information about modifications to fields of a screen is held until the next time that screen is read from.

'MFTXTLEN', screen, field, len

The length of the modified text for the field in "screen" whose name is "field" (returned by MFNAMES) is placed in "len".

'MFTEXT', screen, field, buffer

The modified text for the field in "screen" whose name is "field" (returned by MFNAMES) is placed in "buffer". Because it is assumed "buffer" is long enough, it is first necessary to ask for the modified text length (via MFTXTLEN).

July 1988

'MFLENLOC', screen, field, len, loc

The length of the modified text for the field in "screen" whose name is "field" (returned by MFNAMES) is placed in "len". The location of the modified text is placed in "loc". This indicates what storage location is being used to hold it within the "sscb"; It is only useful if you can access that space, for example, with a pointer variable.

SSINIT

Subroutine Description

Purpose: To initialize the screen-support routines.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSINIT(sscb,clstv)

Parameters:

clstv is the address of the CLS transfer vector to use or 0 if the standard system transfer vector (STDTV) is to be used.

Returned Values:

sscb is the screen-support routines control block. This is needed to call any other screen-support routine.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (4) Bad device.
- 2 (8) Initialization failure.

July 1988

SSLOCN

Subroutine Description

Purpose: To specify the location of a field.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSLOCN(sscb,field,row,column,flength)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
field is an eight-character string specifying the name of the field to have its location set.
row is the row in which the field is to start.
column is the column in which the field is to start.
flength is the length of the field.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0	(0)	Successful return.
1	(4)	Field will overlap another field if CHECK = 1.
2	(8)	Field will go off screen at bottom right.
3	(12)	Field length <= 0.
4	(16)	New field length will cause text to be truncated.
5	(20)	Field does not exist.
6	(24)	No screen is currently active.

Description: See the description of SSDEFF.

SSREAD

Subroutine Description

Purpose: To read back user changes to the most recently written screen.

Location: Resident System

Calling Sequence:

FORTTRAN: CALL SSREAD(sscb,device,aid,row,column,modf,
field,offset)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.
device is a four-character string specifying the name of the device to write the screen to. Currently, this must be 'TERM'.

Returned Values:

aid is an identifier of what key was used to terminate the read operation. The number of keys is device-dependent, so check the individual device. The total number of keys a device has may be determined dynamically by calling SSINFO. For PF-keys the identification number corresponds to the number of the key (i.e., 10 for PF10). For the IBM 3278:

ENTER	returns 0
PA1	returns -1
PA2	returns -2
SYS REQ	returns -3
CLEAR	returns -4

For the Ontel:

PFA1	returns 0
ATTN	returns -1
EOF	returns -2

The other keys have no Ontel equivalent.
row is the row the cursor was in when the read operation was terminated.

July 1988

column is the column the cursor was in when the read operation was terminated.

modf is the number of fields which were modified. A list of the names of the fields which were modified may be obtained using SSINFO.

field is the name of the field the cursor was in when the read was terminated. This value and the next are undefined if the cursor is not in any defined field when the read is terminated. The length of the string depends is eight characters.

offset is the offset of the cursor in that field.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (4) Device does not exist.
- 2 (8) No screen written.
- 3 (12) Cursor is not in any field.
- 4 (16) Field cannot be located.

SSTERM

Subroutine Description

Purpose: To terminate the screen-support routines.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSTERM(sscb)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.

July 1988

SSTEXT

Subroutine Description

Purpose: To specify the text of a field.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSTEXT(sscb,field,text,tlength)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

field is an eight-character string specifying the name of the field to have its text set.

text is the text for the field.

tlength is the length of the text.

Returned Values:

R0 return value followed by R15 value in parenthesis.

0 (0) Successful return.

1 (4) Field length will cause text to be truncated.

2 (8) Field does not exist.

3 (12) No screen is currently active.

Description: See the description of SSDEFF.

SSWRIT

Subroutine Description

Purpose: To write a screen definition to a device.

Location: Resident System

Calling Sequence:

FORTRAN: CALL SSWRIT(sscb,screen,device)

Parameters:

sscb is the screen-support routines control block returned from call to SSINIT.

screen is an eight-character string specifying the name of the screen to write.

device is a four-character string specifying the name of the device to write the screen to. Currently, this must be 'TERM'.

Returned Values:

R0 return value followed by R15 value in parenthesis.

- 0 (0) Successful return.
- 1 (1) Device does not exist.
- 2 (8) Screen does not exist.

July 1988

APPENDIX B: VISUAL MODE IN THE FILE EDITOR

One of the major applications of the screen-support routines is in visual mode for the MTS File Editor, which can be used with the Ontel terminals available in the public work stations on campus. Because many users are already familiar with the behavior of these routines, an explanation of how the screen-support routines work in this particular context may be helpful and informative for people who wish to develop other applications. For this reason a brief description is included here.

When visual mode is invoked, the screen-support routines are initialized and a single screen is defined. The field definition process in the editor is designed to fill every available location on the screen with fields, so no undefined areas exist. If the file lines do not fill all the remaining rows, blank fields are defined for them. As they appear to an interactive user, the fields in any visual-mode screen thus consist of

- (1) A series of "protected" fields on the left that display the MTS line numbers or the "." indicating a yet-unnumbered line. (A field for which the attribute "protected" has been set cannot be modified).
- (2) Fields displaying the text found at the corresponding line numbers in the user's file; these are at least 68 characters in length and are extended to 148, etc., as needed. They may be modified.
- (3) A cost field at the lower left, displaying the cost of the session; a line number you wish to "move" to may also be entered here.
- (4) A work field from which commands may be executed, text inserted into the file lines, etc.
- (5) The "ruler" at the bottom, another protected field.
- (6) Several fields (not normally visible) that print error messages at the bottom of the screen, highlighted if the device supports that field attribute.

Each field has a name, which is 8 characters long; the names of the line number fields are of the form "LINE" followed by the internal form of the line number. The names of the data fields are of the form "DATA" followed by the internal form of the line number. The autoskip attribute is used to move the cursor automatically to the next unprotected field when the end of an earlier one is reached. Under most circumstances, a new screen is defined every time the old one is returned to MTS, and the old one is thrown away, no matter how minor the modifications. The routines allow modifications of existing screens and fields, but the visual editor happens to use a 'clean sweep' approach.

The visual editor displays a screen by making a pair of calls to the screen-support routines SSWRIT and SSREAD. SSWRIT produces the output; the SSREAD also is done immediately, and the routines sit and wait until the user gives some indication that something should be done with the

July 1988

changes: that the screen should move to a new line-number range, that a field needs to be extended, or whatever. Those instructions are signalled by attention-generating keys, i.e., the different PF-keys. The SSINFO subroutine with the parameter MFAIDCUR lets the calling program know which PF-key signalled the return of the screen so the program can take the appropriate action. For example, on an IBM 3278 terminal PA1 terminates the visual mode, PF19 moves the screen up one line (actually by writing a new screen that displays a different line range), PF23 extends the field where the cursor is by 80 more characters, and so on. The number and operation of all the attention-generating keys varies from device to device, but in general they include all the keys described earlier. The character, cursor-movement, insert, or delete keys do not generate a signal to the screen routine calling program. As long as only these are used, the same screen remains on display while the user works, and nothing is sent to MTS.

Once a signal to return the screen is given, a single call to SSINFO gets the names of all the changed fields, and the editor loops through the list of names to record all the changes by rewriting those lines in the user's file. Once that has been completed, the visual editor checks what PF-key was returned, using that to determine what to do next.

Many factors have to be taken into account in rewriting each screen. The size of the FAC (Field Attribute Character) must be counted in determining where the cursor can be positioned, for example. A change in the size or position of any one field affects all the rest. In the IBM 3278, part of the screen reading process is performed by the hardware, while in the Ontel software performs that function. In spite of the complexity of the process, familiarity with the visual editor may help users in understanding how the Screen-Support Routines can be applied in other uses.

July 1988

APPENDIX C: EXAMPLE PASCAL/VS PROGRAM

This sample program writes one field, "TEXT", in the upper-left corner of the screen. It tests and displays the screen-support subroutine return codes, in the process. While most programmers ignore these, incorporating them into a program allows the programmer to check for errors much more easily if something is not working correctly. The program continues to write and read screens whenever a PF-key is hit unless the user returns an ATTN (PA1) or an error causes the program to halt. Finally, it writes out some diagnostic information, and the contents of the modified field, at the terminal.

This program is in the file *SS.PAS.EX. All the types and function definitions are also in the file *SS.PASCALLIB which can be used in an application.

```

Program SS_Routines_Example;
{ *** These define some types and declare
  the screen-support routines calling
  sequences. ***}
type ID_TYPE = Packed array [1..8] of Char;
   DEVICE_TYPE = Packed array [1..4] of Char;
{ *** 3188 is a PASCAL/VS implementation restriction
  on the length of literal strings! *** }
   TEXT_TYPE = Packed array [1..3188] of Char;
Function SSINIT (var SSCB : Integer;
                const Init_CLS_Tv : Integer) : Integer;
                FORTRAN;
type NAMES_LIST = Packed array [1..1000] of Packed array [1..8]
                of Char;
SSINFO_TYPE = (SS_Count, SS_Names, SS_Text, SS_Txtlen,
              SS_Lenloc, SS_Screen, SS_Aidcur, SS_Nonpc);
INFO_VALUE_1 = record
  case SSINFO_TYPE of
    SS_Count: (COUNT_SCREEN: ID_TYPE);
    SS_Names: (NAMES_SCREEN: ID_TYPE);
    SS_Text: (TEXT_SCREEN: ID_TYPE);
    SS_Txtlen: (TXTLEN_SCREEN: ID_TYPE);
    SS_Lenloc: (LENLOC_SCREEN: ID_TYPE);
    SS_Screen: (ROWS: Integer);
    SS_Aidcur: (AIDCUR_SCREEN: ID_TYPE);
    SS_Nonpc: (NONPC_BUFFER: TEXT_TYPE);
  end;
INFO_VALUE_2 = record
  case SSINFO_TYPE of
    SS_Count: (COUNT_COUNT: Integer);
    SS_Names: (NAMES_COUNT: Integer);
    SS_Text: (TEXT_FIELD: ID_TYPE);
    SS_Txtlen: (TXTLEN_FIELD: ID_TYPE);
    SS_Lenloc: (LENLOC_FIELD: ID_TYPE);
    SS_Screen: (COLUMNS: Integer);
    SS_Aidcur: (AID: Integer);

```

July 1988

```

                SS_Nonpc: (NONPC_LENGTH: Integer);
            end;
INFO_VALUE_3 = record
    case SSINFO_TYPE of
        SS_Count: ();
        SS_Names: (NAMES: NAMES_LIST);
        SS_Text: (TEXT_BUFFER: TEXT_TYPE);
        SS_Txtlen: (TXTLEN_LENGTH: Integer);
        SS_Lenloc: (LENLOC_LENGTH: Integer);
        SS_Screen: (FACSIZE: Integer);
        SS_Aidcur: (ROW: Integer);
        SS_Nonpc: (FLAG: Integer);
    end;
INFO_VALUE_4 = record
    case SSINFO_TYPE of
        SS_Count: ();
        SS_Names: ();
        SS_Text: ();
        SS_Txtlen: ();
        SS_Lenloc: (LOCATION: TEXT_TYPE);
        SS_Screen: (PFKEYS: Integer);
        SS_Aidcur: (COLUMN: Integer);
        SS_Nonpc: ();
    end;
Function SSINFO (const SSCB : Integer;
                const Info_What : ID_TYPE;
                var Value_1 : INFO_VALUE_1;
                var Value_2 : INFO_VALUE_2;
                var Value_3 : INFO_VALUE_3;
                var Value_4 : INFO_VALUE_4) : Integer;
FORTRAN;
Function SSBGNS (const SSCB : Integer;
                const Id : ID_TYPE) : Integer;
FORTRAN;
Function SSDEFF (const SSCB : Integer;
                const Id : ID_TYPE;
                const Row : Integer;
                const Column : Integer;
                const Length : Integer;
                const Text : TEXT_TYPE;
                const Text_Len : Integer) : Integer;
FORTRAN;
Function SSWRIT (const SSCB : Integer;
                const Id : ID_TYPE;
                const Device : DEVICE_TYPE) : Integer;
FORTRAN;
type SSREAD_Field_Id = Packed array [1..8] of Char;
Function SSREAD (const SSCB : Integer;
                const Device : DEVICE_TYPE;
                var Aid : Integer;
                var Row : Integer;
                var Column : Integer;
                var Modified_Field_Count : Integer;

```

July 1988

```

        var   Field_Id : SSREAD_Field_Id;
        var   Field_Offset : Integer) : Integer;
FORTRAN;
Function SSTEXT (const SSCB : Integer;
                const Id : ID_TYPE;
                const Text : TEXT_TYPE;
                const Text_Len : Integer) : Integer;
FORTRAN;
Function SSTERM (const SSCB : Integer) : Integer;
FORTRAN;
Var
SSCB : Integer; { This must never be modified! }
RC   : Integer; { All SS routines return this. }
AID  : Integer; { Code for PF-key that terminates an SSREAD. }
ROW  : Integer;
COLUMN : Integer;
COUNT : Integer;
V1 : INFO_VALUE_1; { First parameter to SSINFO. }
V2 : INFO_VALUE_2; { Second parameter to SSINFO. }
V3 : INFO_VALUE_3; { Third parameter to SSINFO. }
V4 : INFO_VALUE_4; { Fourth parameter to SSINFO. }
FIELD_ID : SSREAD_Field_Id; { Must match SSREAD definition. }
OFFSET : Integer;
TEXT : Packed array [1..255] of CHAR;
TEXT_LENGTH : Integer;
Begin
    { Initialize the SS routines. }
    RC := SSINIT(SSCB, 0);
    case RC of
        0: ;
        1,2: begin
            { 1: Bad device
              2: Initialization failure }
            Writeln(' SSINIT failure. RC =', RC:2);
            Halt
            end;
    end;
    { Get some useful info about the device. }
    RC := SSINFO(SSCB, 'SCREEN', V1, V2, V3, V4);
    case RC of
        0: Writeln(' Rows: ', V1.ROWS:2, ' Columns: ', V2.COLUMNS:2,
                  ' Facsize:', V3.FACSIZE:2, ' PFKeys: ', V4.PFKEYS:2);
        1,2: begin
            { 1: Bad Value (depends on INFO desired)
              2: Invalid INFO request }
            Writeln(' SSINFO failure. RC =', RC:2);
            Halt
            end;
    end;
    { Define a screen with the name 'SCREEN'. }
    RC := SSBGNS(SSCB, 'SCREEN');
    { Define a field with the name 'A'. }
    RC := SSDEFF(SSCB, 'A', 0, 0, 4, 'TEXT', 4);

```

```

case RC of
  0: ;
  1,2,3,4,5,6: begin
    { 1: Field already exists
      2: No screen active
      3: Field will overlap (if checking enabled
      4: Field will go off screen
      5: Bad length (<=0)
      6: Field will be truncated }
    Writeln(' SSDEFF failure. RC =', RC:2);
    Halt
  end;
end;
repeat
  { Display the screen. }
  RC := SSWRIT(SSCB, 'SCREEN', 'TERM');
  case RC of
    0: ;
    1,2: begin
      { 1: Bad Device
        2: No screen }
      Writeln(' SSWRIT failure. RC =', RC:2);
      Halt
    end;
  end;
  { Read back changes, etc. }
  RC := SSREAD(SSCB, 'TERM', AID, ROW, COLUMN, COUNT,
    FIELD_ID, OFFSET);
  Writeln(' AID: ', AID:2, ' Row: ', ROW:2, ' Column: ',
    COLUMN:2,
    ' Count: ', COUNT:2);
  case RC of
    0: Writeln(' ', FIELD_ID, ' Offset: ', OFFSET:2);
    1,2,3,4: begin
      { 1: Bad device
        2: No screen written
        3: Cursor not in any field
        4: Error in returned data }
      Writeln(' SSREAD failure. RC =', RC:2);
    end;
  end;
  if COUNT > 0 then begin
    V1.TEXT_SCREEN := 'SCREEN';
    V2.TEXT_FIELD := 'A';
    { Get modified text length for field 'A'. }
    RC := SSINFO(SSCB, 'MFTXTLEN', V1, V2, V3, V4);
    case RC of
      0: begin
        TEXT_LENGTH := V3.TXTLEN_LENGTH;
        Writeln(' Field A Length:', TEXT_LENGTH:3);
      end;
      1,2: begin
        { 1: Bad Value (depends on INFO desired)

```

July 1988

```

                2: Invalid INFO request }
                Writeln(' SSINFO failure. RC =', RC:2);
                Halt
            end;
        end;
    end;
    RC := SSINFO(SSCB, 'MFTEXT', V1, V2, V3, V4);
    case RC of
        0: Writeln(' Field A: ', V3.TEXT_BUFFER:TEXT_LENGTH);
        1,2: begin
            { 1: Bad Value (depends on INFO desired)
              2: Invalid INFO request }
            Writeln(' SSINFO failure. RC =', RC:2);
            Halt
        end;
    end;
    RC := SSTEXT(SSCB, 'A', V3.TEXT_BUFFER, TEXT_LENGTH);
    case RC of
        0: ;
        1,2,3: begin
            { 1: Text will be truncated
              2: No field
              3: No screen active }
            Writeln(' SSTEXT failure. RC =', RC:2);
            Halt
        end;
    end;
end;
end;
until AID = -1;
{ Release SS routines storage. }
RC := SSTERM(SSCB); End.

```

APPENDIX D: EXAMPLE VS FORTRAN PROGRAM

This example shows the beginning of a hypothetical full screen data entry program. A screen is defined which contains the main menu near the bottom of the screen. To keep the example simple, no return codes from the Screen-Support Routines are checked. This program is in the file *SS.FTN.EX.

Four fields are defined:

SELECT: A protected field containing the message

'Select 1, 2 or 3'

This field is left-justified in the first line of the menu.

USER: A highlighted field for the user's response, extending from the end of the SELECT field to the right-hand edge of the screen.

CHOICES: A protected field containing the message:

'1. define data entry form 2. enter data 3. stop'.

It occupies the second line of the menu. Acceptable user input is 1, 2, 3, or \$<mts_command>.

MESSAGE: A protected, highlighted field for displaying messages from the program. It occupies the third line of the menu.

```

IMPLICIT INTEGER (A-Z)
CHARACTER*80 USRTXT
DATA ON/1/, OFF/0/, BEGCOL/0/
*   *** Initialize Screen-Support.
CALL SSINIT (S, 0)
CALL SSINFO (S, 'SCREEN', SCRROW, SCRCOL, FACSIZ, NUMKEY)
*   *** Initialize Main Menu.
BEGROW = SCRROW - 4
CALL SSBGNS (S, 'MAINMENU')
*   *** Create Select field.
SELLEN = 16
CALL SSDEFF (S, 'SELECT ', BEGROW, BEGCOL, SELLEN + FACSIZ,
+           'Select 1, 2 or 3', SELLEN)
CALL SSATTR (S, 'SELECT ', 'PROT', ON)
*   *** Create User Response field.
CALL SSCREF (S, 'USER ')
CALL SSLOCN (S, 'USER ', BEGROW, BEGCOL + SELLEN
+           FACSIZ, SCRCOL - (SELLEN + FACSIZ) - BEGCOL)
CALL SSATTR (S, 'USER ', 'HIGH', ON)
CALL SSCURS (S, BEGROW, BEGCOL + SELLEN + FACSIZ + 1)
*   *** Create Choice field.

```

July 1988

```

CALL SSDEFF (S, 'CHOICES ', BEGROW + 1, BEGCOL, SCRCOL,
+ '1. define data entry form 2. enter data 3. stop', 53)
CALL SSATTR (S, 'CHOICES ', 'PROT', ON)
*   *** Create Message field.
CALL SSCREF (S, 'MESSAGE ')
CALL SSLOCN (S, 'MESSAGE ', BEGROW + 2, BEGCOL, SCRCOL)
CALL SSATTR (S, 'MESSAGE ', 'PROT', ON)
CALL SSATTR (S, 'MESSAGE ', 'HIGH', ON)
*   *** Write the screen.
10 CALL SSWRIT (S, 'MAINMENU', 'TERM')
*   *** Read the screen and pick up user's response.
CALL SSREAD (S, 'TERM', PFKEY, CURROW, CURCOL, NUMMOD,
+ FIELD, OFFSET)
IF (NUMMOD .EQ. 0) GOTO 40
CALL SSINFO (S, 'MFTXTLEN', 'MAINMENU', 'USER ', TXTLEN)
CALL SSINFO (S, 'MFTEXT', 'MAINMENU', 'USER ', USRTXT)
*   *** Skip leading blanks.
I = 1
20 IF (USRTXT(I:I) .EQ. ' ') THEN
    I = I + 1
    IF (I .GT. TXTLEN) GOTO 40
    GO TO 20
ENDIF
*   *** MTS command.
IF (USRTXT(I:I) .EQ. '$') THEN
    CALL CMDNOE (USRTXT(I:TXTLEN), TXTLEN - I + 1)
    CALL SSTEXT (S, 'MESSAGE ', ' ', 1)
    GO TO 10
ENDIF
*   *** Skip trailing blanks, if any.
30 IF (USRTXT(TXTLEN:TXTLEN) .EQ. ' ') THEN
    TXTLEN = TXTLEN - 1
    GOTO 30
ENDIF
IF (I .NE. TXTLEN) GOTO 40
*   *** 1. Define data entry form. *** Subroutine FORM would
*   define a new screen that prompts the user to specify a
*   data entry screen for his application. It would end by
*   rewriting the main menu screen.
IF (USRTXT(I:I) .EQ. '1') THEN
    CALL FORM
    CALL SSTEXT (S, 'MESSAGE ', 'Task 1 finished', 15)
*   *** 2. Enter data. *** Subroutine ENTER would iteratively
*   write the user-defined data entry screen, read in data
*   values from the user, and place them in a file. It
*   would end by rewriting the main menu screen.
ELSEIF (USRTXT(I:I) .EQ. '2') THEN
    CALL ENTER
    CALL SSTEXT (S, 'MESSAGE ', 'Task 2 finished', 15)
*   *** 3. Stop. *** Terminate the program.
ELSEIF (USRTXT(I:I) .EQ. '3') THEN
    CALL SSTERM (S)
    STOP

```

July 1988

```
*      *** Not 1, 2 or 3.
      ELSE
        GOTO 40
      ENDIF
*      *** Tasks 1 or 2 finished.  Return to Main Menu.
      GOTO 10
*      *** Invalid response.
40 CALL SSTEXT (S, 'MESSAGE ', 'Invalid response. Try again.',
+           28)
      GOTO 10
      END
```


July 1988

INDEX

| *TERMTYPES, 137-138

| ? device command, 135

| answerback, 56

| answerback format, Merit/UMnet,
| 144

| ASCII code, 70

| ASCII codes, 139

| attention interrupt,
| Merit/UMnet, 62

| Ontel, 15

| ATTN key, Ontel, 15

| ATTN network command, 155

| BACKSPACE key, Merit/UMnet, 61

| BACKSPACE key, Ontel, 13

| BACKSPECIALECHO device command, 80

| BACKTAB key, Ontel, 14

| BAUD device command, 35

| BELL device command, 35

| BIN modifier,
| Merit/UMnet, 72

| BINARY device command, 81

| binary input,
| Merit/UMnet, 75, 109

| BINARY mode, 75, 109

| BLANK device command, 36, 63, 82

| BREAK key, 75

| BROADCAST device command, 83

| buffer overflow, 71

| CAPS LOCK key, Ontel, 15

| carriage control,
| Merit/UMnet, 84, 143

| case conversion,
| Merit/UMnet, 64, 106, 131, 132

| Ontel, 43, 48

| CBEGIN device command, 36

| CC device command, 84

| CC modifier,
| Merit/UMnet, 72

| CEND device command, 36

| CKLENGTH network command, 155

CLOCK device command, 85

CONNECTIONS device command, 86

CONTROL network command, 155

conversation buffer,
| Ontel, 16, 22, 40, 40, 49

COPY network command, 155, 160

CRDELAY device command, 87

CTRL key, Ontel, 13

CTRL-C, 61

CTRL-D, 74

CTRL-E, 62

CTRL-H, 61

CTRL-P, 61

CTRL-Q, 74

CTRL-S, 74

cursor-positioning keys, Ontel, 13

data lost message, 71

DCC device command, 37, 63, 88

DEF? device command, 30, 39

DEFINE device command, 29, 38

DEL key, Ontel, 14

DELETE key, Ontel, 14

delete-line character,
| Merit/UMnet, 61

delete-previous character,
| Merit/UMnet, 61

DEQUEUE device command, 89

device commands,
| Merit/UMnet, 62

| Ontel, 22, 33-49

device-command character,
| Merit/UMnet, 62, 88

| Ontel, 22, 37

DONT device command, 90

DUMP device command, 39

DUPLEX device command, 91

ECHO device command, 92

ECHO network command, 156

EDIT device command, 93

EMPTY device command, 40

end-of-file character,
| Merit/UMnet, 61

	Ontel, 15	literal-next character,	
	end-of-line character,	Merit/UMnet, 61	
	Merit/UMnet, 61	lowercase conversion,	
	ENTER device command, 40	Merit/UMnet, 64, 106, 131, 132	
	EOF key, Ontel, 15	Ontel, 43, 48	
	EOF network command, 156		
	ERASE TO EOL key, Ontel, 15	magnetic-tape cassette input, 74,	
	Error messages, Merit/UMnet, 146	109	
		MCMD network command, 156	
	FAST device command, 41	Merit/UMnet Computer Network,	
	FAST output mode, Ontel, 41	55-148	
	FDname modifiers, 145	Merit/UMnet device commands,	
	FDX mode, 73, 109	76-136	
	FIX key, Ontel, 15	?, 135	
	FLIP device command, 98	BACKSPECIALECHO, 80	
	FORMFEED device command, 99	BINARY, 81	
	FREEZE device command, 41, 48	BLANK, 82	
	full-duplex, 73, 109	BROADCAST, 83	
	FUNCTION device command, 26, 42	CC, 84	
		CLOCK, 85	
	GRAB device command, 100	CONNECTIONS, 86	
		CRDELAY, 87	
	half-duplex, 73, 109	DCC, 88	
	hard-wired connection, 56	DEQUEUE, 89	
	HDX mode, 73, 109	DONT, 90	
	HELLO device command, 101	DUPLEX, 91	
	HEX device command, 102	ECHO, 92	
	hexadecimal editing,	EDIT, 93	
	Merit/UMnet, 102	FLIP, 98	
	HIST device command, 43	FORMFEED, 99	
	HOME key, Ontel, 14	GRAB, 100	
		HELLO, 101	
	IBM 3278, 166	HEX, 102	
	INLEN device command, 103	INLEN, 103	
	input area, Ontel, 16	INTERLOCK, 104	
	INPUT network command, 156	JOB, 105	
	input-record length, 103	LCI, 106	
	INSERT key, Ontel, 14	LINES, 107	
	insertion mode,	MIX, 108	
	Ontel, 14	MODE, 109	
	INTERLOCK device command, 104	n, 136	
		NPC, 111	
	JOB device command, 105	OPERATOR, 112	
		OUTLEN, 113	
	keyboard editing, Merit/UMnet, 93	PAGWAIT, 114	
		PARITY, 116	
	LC device command, 43	PFX, 117	
	LCI device command, 106	QUIT, 118	
	LINE device command, 43	READER, 119	
	LINE output mode, Ontel, 43	RESET, 120	
	line-editing functions, 61	SCROLL, 121	
	line-reentry, Ontel, 20	SENSE, 122	
	LINES device command, 107	SNS, 123	

July 1988

STATUS, 124	device-command character, 22,
TABI, 125	37
TABSPECIALECHO, 126	keyboard, 11
TBDELAY, 127	numeric pad, 28, 44
TERMINAL, 57, 128	program-function keys, 25, 45
TEST, 129	window, 49
UCB, 130	Ontel device commands,
UCI, 131	BAUD, 35
UCO, 132	BELL, 35
WAIT, 133	BLANK, 36
WIDTH, 134	CBEGIN, 36
Microcomputers, 166	CEND, 36
MIX device command, 108	DCC, 37
MODE device command, 73, 109	DEF?, 30, 39
MTS FDname modifiers, 72	DEFINE, 29, 38
MTS network command, 156	DUMP, 39
	EMPTY, 40
n device command, 136	ENTER, 40
NET command, 64, 149	FAST, 41
NET network command, 157	FREEZE, 41, 48
NET network commands, 155-159	FUNCTION, 26, 42
ATTN, 155	HIST, 43
CKLENGTH, 155	LC, 43
CONTROL, 155	LINE, 43
COPY, 155, 160	PAD, 28, 44
ECHO, 156	PAGE, 44
EOF, 156	PF?, 26, 46
INPUT, 156	PFA, 25, 45
MCMD, 156	PFB, 25, 45
MTS, 156	RESET, 46
NET, 157	ROLL, 47
NETPFX, 157	TABI, 47
OPER, 157	THAW, 48
OUTLEN, 157	UC, 48
READ, 157	WB, 23, 49
RECEIVE, 157	WF, 23, 49
REMOTE, 158	WL, 23, 49
RESET, 158	WR, 23, 49
RETURN, 158	OPER network command, 157
SELECT, 158	OPERATOR device command, 112
STOP, 159	OUTLEN device command, 63, 113
NETPFX network command, 157	OUTLEN network command, 157
Network batch access, 64	output modes, Ontel, 24
Batch file copying, 67	FAST, 24, 41
Execution jobs, 65	LINE, 24, 43
Print jobs, 66	PAGE, 24, 44
NPC device command, 111	ROLL, 24, 47
numeric pad, Ontel, 13, 28, 44	output-record length, 113
	Merit/UMnet, 63
Ontel, 11-54, 166	overrun error, 71
case conversion, 43, 48	
conversation buffer, 40, 40	PAD device command, 28, 44
device commands, 22, 33-49	PAGE device command, 44

| PAGE output mode, Ontel, 44
 | PAGEWAIT device command, 114
 | paper-tape input, 74, 109
 | parity,
 | Merit/UMnet, 70
 | PARITY device command, 116
 | parity error, 70
 | password, 57
 | PAUSE key, Ontel, 15, 20
 | PCP, (see primary communications
 | processor)
 | PF? device command, 26, 46
 | PFA device command, 25, 45
 | PFB device command, 25, 45
 | PFX device command, 117
 | port number, Merit/UMnet, 57
 | primary communications processor,
 | 55
 | program-function keys,
 | Ontel, 25, 45
 |
 | QUIT device command, 118
 |
 | rates,
 | deferred-priority, 58
 | low-priority, 58
 | minimum-priority, 58
 | READ network command, 157
 | READER device command, 119
 | RECEIVE network command, 157
 | REMOTE network command, 158
 | RESET device command, 46, 63, 120
 | RESET network command, 158
 | return codes,
 | Merit/UMnet, 141
 | RETURN key, Ontel, 13
 | RETURN network command, 158
 | ROLL device command, 47
 | ROLL output mode, Ontel, 47
 |
 | SCP, (see secondary communications
 | processor)
 | Screen-Support Routines, 165-208
 | SCROLL device command, 121
 | secondary communications proces-
 | sor, 55
 | SELECT network command, 158
 | SENSE device command, 122
 | SHIFT key, Ontel, 13
 | SIGNON command, 57
 | SNS device command, 123
 | SP modifier,
 | Merit/UMnet, 72, 74
 |
 | SSATTR subroutine, 170, 177
 | SSBGNS subroutine, 167, 179
 | SSCNTS subroutine, 168
 | SSCREF subroutine, 168, 180
 | SSCTNS subroutine, 181
 | SSCTRL subroutine, 182
 | SSCURS subroutine, 171, 184
 | SSDEFF subroutine, 168, 185
 | SSDELF subroutine, 171, 186
 | SSDELS subroutine, 172, 187
 | SSENDS subroutine, 168, 172, 188
 | SSINFO subroutine, 174, 189
 | SSINIT subroutine, 167, 192
 | SSLOCN subroutine, 169, 193
 | SSREAD subroutine, 173, 194
 | SSTERM subroutine, 175, 196
 | SSTEEXT subroutine, 169, 197
 | SSWRIT subroutine, 172, 198
 | STATUS device command, 124
 | STOP network command, 159
 | suspending output,
 | Ontel, 20
 |
 | TAB device command, 63
 | TAB key, Ontel, 14
 | TABI device command, 47, 125
 | TABSPECIALECHO device command, 126
 | tabulation,
 | Merit/UMnet, 63, 125
 | Ontel, 47
 | TAPE mode, 74, 109
 | TBDELAY device command, 127
 | telephone connection, 56, 90
 | TERMINAL device command, 57, 128
 | terminal identification, 57
 | terminal type,
 | Merit/UMnet, 57, 128, 137-138
 | TEST device command, 129
 | THAW device command, 48
 | time-out feature, 133
 |
 | UC device command, 48
 | UCB device command, 130
 | UCI device command, 64, 131
 | UCO device command, 64, 132
 | uppercase conversion,
 | Merit/UMnet, 64, 131, 132
 | Ontel, 43, 48
 |
 | WAIT device command, 133
 | WB device command, 23, 49
 | WF device command, 23, 49
 | WIDTH device command, 64, 134

July 1988

- | window,
 - | Ontel, 16, 22, 49
 - | WL device command, 23, 49
 - | WR device command, 23, 49

- | ZIP key, Ontel, 14

|

| Reader's Comment Form

| Terminals and Networks in MTS
| Volume 4
| July 1988
|

| Errors noted in publication:

| Suggestions for improvement:

| Your comments will be much appreciated. Send the completed form to the
| Computing Center by Campus Mail or U.S. Mail, or drop it in the
| Suggestion Box at the Computing Center, NUBS, or UNYN.

| Date _____

| Name _____

| Address _____

| _____

| _____

| Publications
| Computing Center
| University of Michigan
| Ann Arbor, Michigan 48109

Update Request Form

Terminals and Networks in MTS
Volume 4
July 1988

Updates to this manual will be issued periodically as errors are noted or as changes are made to MTS. If you would like to have these updates mailed to you, please submit this form.

Updates are also available in the files at some of the Computing Center's larger public stations such as NUBS and UNYN; there you may obtain any updates to this volume that may have been issued before the Computing Center receives your form. Please indicate below if you want to have the Computing Center mail you any previously issued updates.

Name _____

Address _____

Previous updates needed (if applicable): _____

Send the completed form to the Computing Center by Campus Mail or U.S. Mail, or drop it in the Suggestion Box at the Computing Center, NUBS, or UNYN. Local users should give a Campus Mail address.

Publications
Computing Center
The University of Michigan
Ann Arbor, Michigan 48109

Users associated with other MTS installations (except the University of British Columbia) should return this form to their respective installations. Addresses are given on the reverse side.

| Addresses of other MTS installations:

| Publications Clerk
| 352 General Services Bldg.
| Computing Services
| The University of Alberta
| Edmonton, Alberta
| Canada T6G 2H1

| Information Officer, NUMAC
| Computing Laboratory
| The University of Newcastle upon Tyne
| Newcastle upon Tyne
| England NE1 7RU

| Rensselaer Polytechnic Institute
| Documentation Librarian
| 310 Voorhees Computing Center
| Troy, New York 12181

| Simon Fraser University
| Computing Centre
| User Services Information Group
| Burnaby, British Columbia
| Canada V5A 1S6

| Wayne State University
| Computing Services Center
| Academic Services Documentation Librarian
| 5925 Woodward Ave.
| Detroit, Michigan 48202

THE IBM 3278 DISPLAY STATION

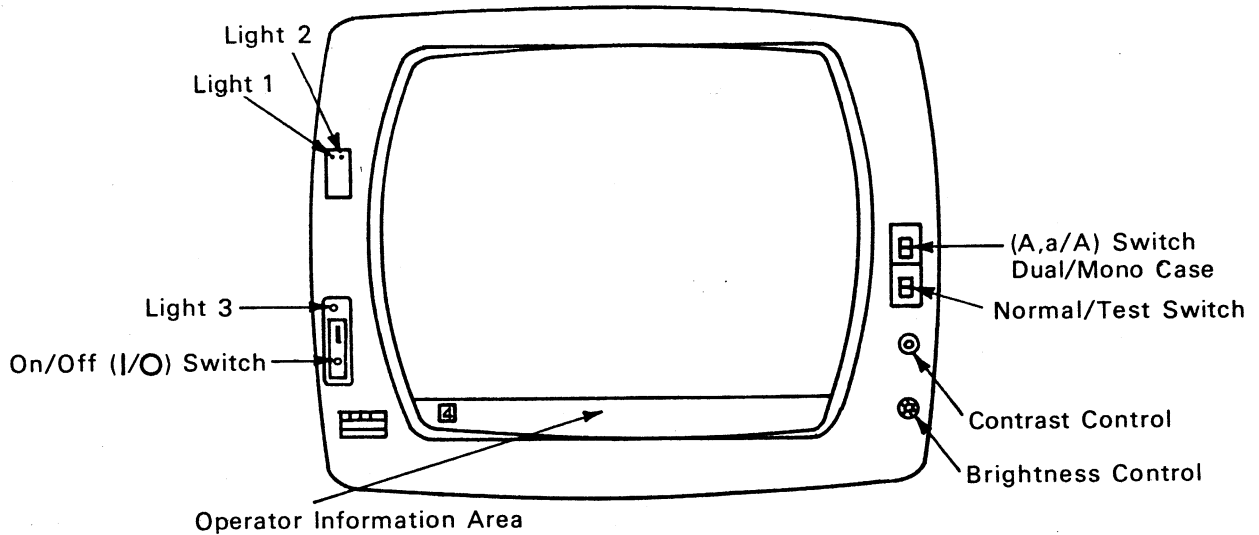
This section describes the use of the IBM 3278 Display Station. An equivalent terminal, the Lee Data terminal, is described in Appendix F to this section. Both of these terminals are available only at the Computing Center Building on North Campus and generally are not available for public use.

The IBM 3278 Display Station consists of a cathode-ray-tube screen (CRT) console and a separate keyboard unit. The screen is organized in a 24-line, 32-line, or 43-line by 80-column format depending on the particular model, e.g., IBM 3278 Model 2, 3, 4, or 5, respectively. Information can be entered onto the screen from either the keyboard or the computer. Characters typed from the keyboard remain in a hardware buffer inside the terminal and are not transmitted to the computer until the ENTER key is pressed. Thus, changes can be made and errors can be corrected directly on the screen before the computer reads the information.

The power switch for the terminal is located to the left side of the screen console (the red switch). It takes about 15 seconds for the CRT to warm up. On the right side of the screen are two blue switches. The upper blue switch controls the alphabetic case of the display screen. When set to "A,a", the screen will display both upper- and lowercase alphabetic characters; when set to "A", only uppercase alphabetic characters are displayed. This switch does not affect the entering of characters into the system from the keyboard; thus, it should always be set to the "A,a" position. The lower switch controls whether the terminal is in normal operating mode or in test mode; the switch should always be set to normal mode. The contrast and brightness control knobs are located in the lower-right corner of the screen console. The contrast control is on top and adjusts the contrast between normal- and high-intensity output and normally should be set to the highest contrast position. The brightness control knob is just below and varies the intensity of the picture. Note that an overly bright picture may damage the CRT and is also uncomfortable to view for extended periods. Figure 1 illustrates the display screen console.

An operator information area is located at the bottom of the screen. This area is used by the system to display information about the status of the system and the terminal.

The cursor on the terminal indicates the position at which the next character typed on the keyboard will be placed. The normal cursor resembles the underscore character "_". There is also an alternate cursor which resembles a box.




4 Ready. Control unit is working.

A Control unit is connected to the system.

■ Terminal is connected to your job.

TEST Terminal is in TEST mode.

X DO NOT ENTER (Input inhibited).

X  Wait. System is performing a function.

X  **?**+ What? Input not understood. Depress RESET, and try again.

X **SYSTEM** Wait. System is performing a function.

X  Input inhibited. Depress RESET, and move cursor.

^ INSERT mode. Depress RESET to exit.

^ Upshift.

^ Keyboard locked in upshift.

Figure 1. The IBM 3278 Screen Console

THE KEYBOARD

The keyboard provides a character set containing 96 graphical characters along with several special editing and program-function keys. Figure 2 illustrates the keyboard.

The SHIFT key (indicated by a wide up-arrow) selects the uppercase character or symbol on the character keys. Pressing the SHIFT LOCK key logically presses the SHIFT key and locks the keyboard in a shifted condition, until the SHIFT key is pressed to return to an unshifted condition. A small up-arrow will appear in the center of the operator information area when the terminal is in the shifted condition. The upper blue switch to the right of the screen affects only the appearance of lowercase characters on the screen. When set to the "A" position, lowercase letters are displayed in uppercase; however, they are still transmitted to the system as lowercase letters.

The ALT key selects the alternate control functions available on many keys. These functions are indicated on the lower-front side of the key. The ALT key must be pressed in conjunction with the desired key to select the alternate function.

Ordinary character keys enter the indicated characters or symbols onto the screen at the current position of the cursor. Entry of a character advances the cursor to the next character position. Note that it is possible to replace a character on the screen by simply overstriking it with another character.

Most keys are typematic; that is, they repeat their function at the rate of about ten characters per second if held down for at least one second.

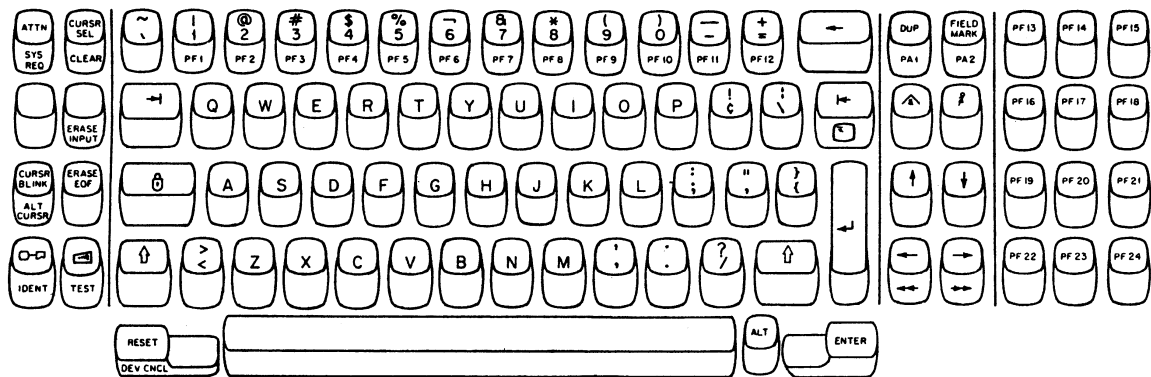
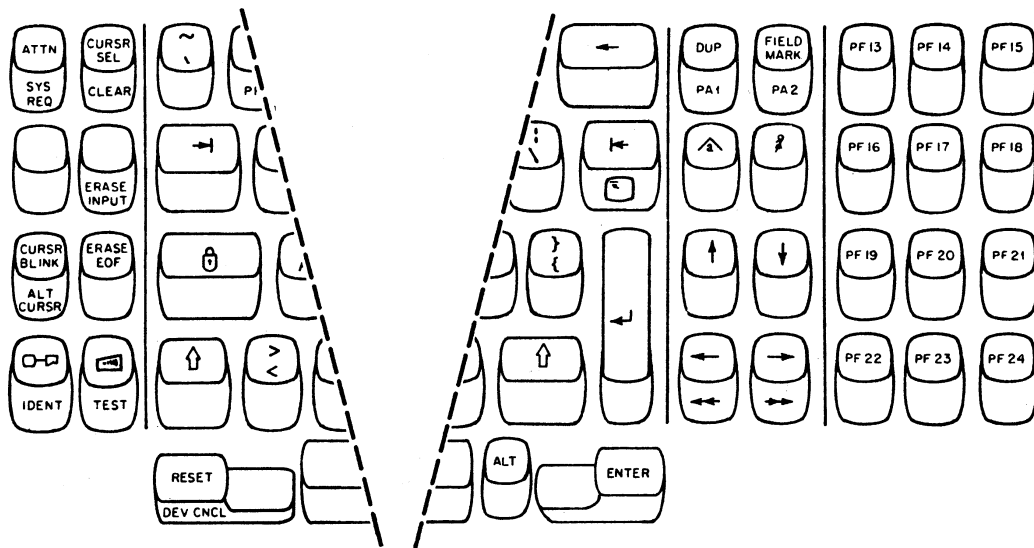


Figure 2. The IBM 3278 Keyboard



indicator will appear in the right side of the operator information area and the keyboard will be locked. Normally, the indicator is reset by the system when the input is accepted.

Cursor-Positioning Keys

The SPACE BAR advances the cursor one character position and replaces the character over which it passes with a blank. This is different from the RIGHT (right-arrow) key which moves the cursor to the right without replacing the character with a blank.

The remaining "cursor-positioning keys" fall into two groups: character-based keys and line-based keys. All of these keys position the cursor without changing any of the information already on the screen.

The character-based keys include the UP, DOWN, LEFT, RIGHT, and BACKSPACE keys, which move the cursor one character position at a time in the indicated direction. These keys are labeled with up, down, left, and right arrows (see Figure 2). The BACKSPACE and LEFT keys perform the same function. The UP, DOWN, LEFT, and RIGHT keys are typematic. In addition, the LEFT and RIGHT keys will move the cursor twice as fast if pressed in conjunction with the ALT key.

The line-based keys include the TAB, BACKTAB, and RETURN keys. The RETURN and TAB keys position the cursor to the first character position of the next displayed line. Note that the RETURN key does not transmit a line to the system as is the case for other terminals. The BACKTAB key moves the cursor to the first character position of the current line (or the previous line, if the cursor is already at the first position of the current line). All of these keys repeat their function at a rate of about 10 operations per second if held down for at least one second.

Special-Editing Keys

The character at the cursor position can be deleted by pressing the DELETE key. All characters on the current line to the right of the deleted character are shifted left one position. The character after the deleted character is then at the cursor position. The DELETE key is not typematic.

Characters can be inserted by pressing the INSERT key to enter insertion mode; a small up-arrow will appear in the center of the operator information area. Each character typed while in insertion mode is inserted before the character in the cursor position. The cursor and the remainder of the line to its right are displaced one position to the right as each character is inserted. If the line is completely filled,

the insertion is not allowed. While the terminal is in insertion mode, the cursor can be moved and characters inserted at various places on the screen. Insertion mode is exited by pressing the RESET, SYS REQ, CLEAR, ENTER, PA1 or PA2 keys, or any of the PF (program-function) keys. A deletion/insertion example follows; the underscore indicates the current cursor position:

CHANGE TH <u>IS</u> THING	Initial display
CHANGE TH <u>S</u> THING	DELETE key pressed
CHANGE TH <u>_</u> THING	DELETE key pressed
CHANGE TH <u>A</u> _THING	INSERT key pressed (to enter insertion mode); 'A' typed
CHANGE TH <u>A</u> S_THING	'S' typed
CHANGE TH <u>A</u> <u>S</u> _THING	BACKSPACE pressed
CHANGE TH <u>A</u> T <u>_</u> THING	RESET pressed (to exit insertion mode) 'T' typed

Additional Control Keys

The CLEAR key (alternate function of CURSOR SEL) rewrites the screen with a clear user input area representing the contents of the screen after the last time the ENTER key was pressed. This may be useful if portions of the screen have been accidentally altered. Also, if CLEAR is used during session termination, it terminates the display of signoff statistics, thus enabling the user to sign on again as quickly as possible.

The ERASE EOF key deletes all characters in the line from the cursor position to the end of the line. To retype an entire line, first position the cursor to the beginning of the line if necessary using the BACKTAB key, and then press the ERASE EOF key to erase what was previously typed. The cursor will remain in position for retyping the line.

The SYS REQ key (alternate function of ATTN) causes the system to "pause" in displaying output. Its function is described in the section "Pause/Continue."

The PA1 key (alternate function of DUP) generates an attention interrupt. The "X" indicator will appear in the operator information area and the keyboard will be locked until the interrupt is processed.

The PA2 key (alternate function of FIELD MARK) generates an end-of-file condition.

Note that the CLEAR, SYS REQ, PA1, and PA2 keys occasionally may be ignored by the system if it is writing output on the screen. Pressing the key again should produce the desired result.

The ERASE INPUT key deletes all of the text on the screen and positions the cursor to the first character position of the first line. Its usefulness is limited, since under normal conditions this is not a desirable result.

The CURSOR BLINK key can be used to select either a blinking or an unblinking cursor. The ALT CURSOR key can be used to select either an underscore cursor or a box-shaped cursor. Either of the above options is only a matter of personal choice.

The NOISE key (alternate function of TEST) controls the amount of noise produced by the keyboard. The two possible settings are loud and soft. When the keyboard is locked (an "X" displayed in the operator information area), the effect of the NOISE key is reversed; that is, if during normal entry a soft clicking is heard, when the keyboard is locked, a loud clicking will be heard if an attempt is made to enter data.

The power switch can be used to force the system to pause while producing output in line or roll mode. The power switch should be turned off briefly and turn back on again.

The following keys have no useful function in MTS: ATTN, CURSR SEL, DEV CNCL, DUP, FIELD MARK, IDENT, and TEST.

CONVERSATIONAL OPERATION

Initiating the Session

Initially, a block MTS is displayed as shown in Panel 1 if the terminal is already powered on. If the power is off, the user should turn the power on, wait a few seconds, and then press the CLEAR key to obtain the block MTS display. The information on the first line gives the device name (DS04) and the task number (12) associated with the terminal. Any communication with the system operators with respect to terminal or system malfunctions should refer to these identification items. These items also may be obtained by the %JOB device command.

The cursor initially is positioned at the beginning of the input area (Panel 2). Entry of the first input line alerts the system to the user's presence and initiates the session. When the SIGNON command is entered, the screen is cleared of the block MTS.

After the SIGNON command has been entered, MTS will prompt the user for the password (Panel 3). A special input area is provided for the password so that it is not displayed. In this way, the password is totally secure as it is typed.

Conversation Buffer

Occasionally, a user may need to refer to a previous portion of the current terminal session which is no longer displayed on the screen. A large portion of the session is saved automatically by the system in an internal area called the "conversation buffer." The conversation buffer normally holds up to 512 lines of text from the session. When it becomes full, the oldest lines are discarded as new ones are added. The first 77 characters of each of the most recent conversation buffer lines usually are displayed on the screen and are termed the "window." This window can be moved to the right or left to view the undisplayed parts of long lines. The window also can be "scrolled" forward and backward to view all parts of the conversation buffer. This is described in the section "Scrolling."

Input/Output Procedures

All input and output lines have a prefix character that identifies the system component (MTS command language interpreter, file editor, etc.) which issued the output line or which expects an input line. Prefix characters are discussed in more detail in the section "Prefix Characters."

When the terminal receives an output line from the system, it places the prefix and the line into the conversation buffer. The prefix and the first 77 characters of the output line then are displayed on the next screen line. (The prefix is placed in the prefix area which is three character positions long, and the first 77 characters of the output line are placed in the remainder of the screen line.) The remaining portion of the line can be displayed using the procedures described in the section "Scrolling" or the section "Program-Function Keys." Up to 41 lines (IBM 3278 Model 4 terminals) can be displayed on the screen at one time. When this number is exceeded, the screen is erased, the most recent lines in the conversation buffer (normally 14) are displayed at the top of the screen, and the input/output process continues at the next screen line.

When an input line is requested by a component of the system (e.g., the file editor), its prefix is displayed and the cursor is positioned at the beginning of an input area; the input line can then be entered. Alternatively, lines can be entered even though the system has not yet requested input. When the percent sign (%) prefix is displayed, the user can type input and enter it via the ENTER key. Such lines are queued by the terminal and are delivered to the appropriate system component in response to subsequent input requests.

```

University of Michigan Computing Center - Device: DS08 Task: 88
% _

MM      MM  TTTTTTTTTT  SSSSSSSSS
MMM     MMM  TTTTTTTTTT  SSSSSSSSSSS
MMMM    MMMM   TT        SS      SS
MM MM  MM MM   TT        SS
MM  MMM  MM    TT        SSS
MM  MM  MM    TT        SSSSSSSSS
MM      MM    TT        SSSSSSSSS
MM      MM    TT          SSS
MM      MM    TT         SS      SS
MM      MM    TT        SSSSSSSSSSS
MM      MM    TT        SSSSSSSSS

```

1

Panel 1. Initial Display

```

University of Michigan Computing Center - Device: DS08 Task: 88
% signon wxyz_

MM      MM  TTTTTTTTTT  SSSSSSSSS
MMM     MMM  TTTTTTTTTT  SSSSSSSSSSS
MMMM    MMMM   TT        SS      SS
MM MM  MM MM   TT        SS
MM  MMM  MM    TT        SSS
MM  MM  MM    TT        SSSSSSSSS
MM      MM    TT        SSSSSSSSS
MM      MM    TT          SSS
MM      MM    TT         SS      SS
MM      MM    TT        SSSSSSSSSSS
MM      MM    TT        SSSSSSSSS

```

1

Panel 2. Entering the SIGNON Command

```
University of Michigan Computing Center - Device: DS08 Task: 88
# SIGNON WXYZ
# ENTER USER PASSWORD.
?
-
```

1

Panel 3. Entering the Password

```
University of Michigan Computing Center - Device: DS08 Task: 88
# SIGNON WXYZ
# ENTER USER PASSWORD.
?
# **LAST SIGNON WAS: 16:15:05 THU NOV 06/80
# USER "WXYZ" SIGNED ON AT 18:20:45 ON FRI NOV 07/80
# LIST FYLE
> 1 ABC DEF GHI
> 2 ABC
# END OF FILE
# SIG $
# WXYZ 18:23:21-18:24:48 FRI NOV 07/80
# $.15
# $133.12
```

1

Panel 4. Complete Sample Session

As each input line is delivered to the system, it is also placed into the conversation buffer and the first 77 characters of the input line are then displayed on the next available screen line. Note that any queued input is discarded if the PA1 (attention interrupt) key is pressed.

The terminal will preserve any input being typed if an output operation interrupts the input process unless the %DUPLEX=OFF device command has been issued. The keyboard can be locked momentarily but this normally will not affect the contents of the input area or the cursor position. On occasion, especially if the output rate is higher than the normal rate of 5, a character may be lost. If duplex mode has been disabled, the entire line will be lost.

In duplex input mode, a request for input (for which there are no queued lines) which occurs while the user is typing in the input area will cause the "%" prefix to be changed to the prefix associated with the input request and the system will then await completion of the line. If duplex mode is disabled, the input line will be lost.

The screen-editing facilities of the terminal can be used to correct an input line before the line is entered. Although the cursor can be moved into the three-character prefix area, any attempt to enter characters in this area is ignored. The cursor must be moved out of the prefix area before typing can proceed.

When input is requested, any line already displayed on the screen can be entered by moving the cursor to any position in the desired line and pressing the ENTER key. The selected line can be edited before entering. However, if an attempt is made to insert characters in a line that is longer than 77 characters (without deleting any characters), the insertion is suppressed. The line can be moved to the normal input area by using any of several program-function keys (see the section "Program-Function Keys"). If the line to be reentered is not entirely displayed on the screen, the undisplayed parts are appended automatically before it is delivered to the system. These features allow an erroneous line that was rejected by the system to be corrected and reentered.

In duplex mode, if an output operation occurs before the line is entered, the entire line will be moved to the input area, along with the cursor and any changes that have been made to the line. If this occurs and duplex mode has been disabled, the cursor will be moved to an empty input area.

If the system sends an unprintable character to the terminal, it is displayed on the screen as a question mark "?".

If the system is awaiting input and a line is not entered within 30 minutes, it assumes that the terminal has been abandoned and automatically signs the user off. The 30-minute time limit is enforced during binary as well as nonbinary I/O which means, for example, that I/O from the MTS editor visual mode is subject to the time limit.

Pause/Continue

The SYS REQ key causes the system to either "pause" or "continue." After the key is pressed, all further output is held until the key is pressed again. The input area prefix is displayed at a higher-than-normal intensity while the terminal is in pause state, and %PAUSE is displayed in the lower-right corner of the screen. If the user presses the SYS REQ key and then leaves the terminal in a pause state for more than 30 minutes, the system assumes that the terminal has been abandoned and automatically signs the user off. At certain times when operating in roll or line modes, it may be difficult to get the terminal to respond to the SYS REQ key. If this happens, the power should be switched off and then immediately on again to force a "pause."

End-of-Files

An end-of-file condition may be placed in the input queue by pressing the PA2 key or entering the %EOF device command.

Attention Interrupts

An attention interrupt can be generated at any time by pressing the PA1 key or issuing the %ATTN device command. When an attention interrupt is generated, any queued input lines are discarded. Note that the ATTN key does not generate an attention interrupt.

Device Commands

Device commands can be used to control the device-dependent functions such as scrolling, case translation, hexadecimal input editing, tabulation editing, etc. Appendix A gives the syntax and effects of all the device commands.

Most device commands can be issued whenever an input line is expected. However, they are processed by the terminal routines rather than by MTS. Device commands must begin with the current device-command character, initially a "%", in order to indicate that the input line is a device command. After processing the command, the user is prompted again for another input line. Most device commands can also be issued asynchronously like normal input lines, but are acted upon immediately and are not queued.

Device commands are not recorded in the conversation buffer and they disappear from the screen as soon as they are processed.

The acknowledgment of device commands depends on the type of command entered and on whether or not it is entered correctly. If the command is in error, the message

```
***Invalid device command: xxxxx
```

where "xxxxx" is the device command in error, is printed and the user can correct the command and reenter it. If the command calls for some overt action such as scrolling, its execution constitutes the acknowledgment. If the command merely sets a parameter and no error message appears, the user may assume that the command was successful.

Some device commands can be issued only when the system is awaiting input. If an attempt is made to enter one of these device commands at another time, the message

```
***Device command restricted: xxxxx
```

will be printed.

To enter a data line that begins with a "%", the user must either use the literal-next character, change the device-command character via the %DCC device command, or double the device-command character "%%".

Scrolling

Seven device commands allow the display window to be scrolled to any part of the conversation buffer; they are:

```
%WINDOW=n - display window "n"  
%WF=n     - "window" forward "n" lines  
%WB=n     - "window" backward "n" lines  
%WR=n     - "window" right "n" columns  
%WL=n     - "window" left "n" columns  
%WF=*     - "window" forward 1 screen unit  
%WB=*     - "window" backward 1 screen unit
```

The window is normally 77 characters wide by 11 lines deep. Only the current contents of the conversation buffer can be displayed.

At signon time, a window number is initialized and displayed in the lower right-hand corner of the screen. This window number can be used to refer to particular screen segments when scrolling forward or backward through the conversation buffer. The %WINDOW=n command displays the specified window.

The %WR=n command moves the window to the right by "n" columns. This is convenient for viewing long lines. The %WL=n command moves the window to the left. The prefix area is not affected by these commands and remains displayed. The horizontal displacement (the column number of the first column displayed) of the the window is shown in the lower right-hand corner of the screen after the window number (if it is not 1). The %WB=n command moves the window backward in the conversation buffer by "n" lines and is useful for viewing previous parts of the session. %WF=n moves the window forward. There is no horizontal displacement involved in the window backward and window forward operations.

If no parameter is given with a move window command, the maximum movement is assumed. For example, %WL moves the window to the extreme left of the conversation buffer, and %WB causes the first window in the conversation buffer to be displayed.

Each time the conversation buffer is updated, i.e., after the completion of each system input or output operation, the window is forced to the most recent line, overriding any backward displacement. The horizontal displacement is not affected. The program-function keys also can be used to conveniently issue these device commands (see the section "Program-Function Keys").

Output Modes

Four output modes are provided: line, roll, page, and fast; these are controlled by corresponding device commands:

```
%LINE   - enter line mode
%ROLL   - enter roll mode
%PAGE   - enter page mode
%FAST   - enter fast mode
```

In line mode, output lines appear in the next available screen line as they are produced by the system. When the 41st screen line (IBM 3278 Model 4 terminals) has been used for output, the screen is erased, the most recent lines in the conversation buffer (normally 14) are displayed at the top of the screen, and the input/output process continues at the next available screen line. Line mode is the default.

In roll mode, each new input or output line added to the end of the screen causes the top line to be erased. Thus, the output appears to be rolling up the screen.

In page mode, output is displayed only when a whole new screenful is available from the system or when an input line is requested. A "pause" state is automatically entered after each screenful is displayed, and the the SYS REQ key must be pressed to continue to the next screenful (see the section "Pause/Continue").

In fast mode, output lines are recorded in the conversation buffer but are not written to the screen until an input line is requested by MTS. If more than a screenful of output is generated between input requests, only the last screenful is displayed and as much of the preceding screenfuls that can be maintained in the conversation buffer are saved. If large amounts of output are produced between input requests, it is possible for the data to be lost as the conversation buffer "wraps-around."

Lowercase

Lowercase alphabetic characters can be translated to uppercase as they are entered from the keyboard and placed into the conversation buffer. This translation can be controlled by the device commands:

```
%LC - disable uppercase translation
%UC - enable uppercase translation
```

The default is no uppercase translation.

Hard-Copy Output

A "hard-copy" of the conversation buffer can be obtained by issuing the command

```
$COPY *MSOURCE*@SP *PRINT*
```

when in MTS command mode. The conversation buffer is treated as a read-only MTS line file which is addressed as *MSOURCE*@SP. The COPY command causes the entire buffer to be written on *PRINT*. By default, this buffer can contain the last 512 lines of the session; the %BUFFER device command can be used to increase the size of the conversation buffer. The PF2 or PF14 program-function keys can also be used to obtain a copy of the conversation buffer while in MTS command mode (see below).

Line-number ranges can be used to obtain a partial copy of the buffer. Line 1 refers to the header line (which contains a "1" in column 1 for carriage control), line 2 contains the first line in the buffer, and *L (or LAST) contains the last line in the buffer. Combinations of these can be used to obtain any desired portion of the buffer, either forward or backward. Line numbers outside this range and nonintegral line numbers will yield unpredictable results. The %CBEGIN and %CEND device commands also can be used to obtain a partial copy of the buffer.

Program-Function Keys

There are two sets of program-function keys. The first set, labeled PF1 through PF12, is located across the top of the keyboard as alternate functions of the numeric keys. The second set, labeled PF13 through PF24, is located on the right side of the keyboard. Pressing one of these keys causes a predefined character string such as a device command or an MTS command to be entered as though it had been typed on the keyboard.

All program-function keys have initially defined values which can be changed by the user. Figure 4 shows the predefined values of the twelve keys in the PF13 through PF24 program-function key matrix. Program-function keys PF1 through PF12 have the same definitions as PF13 through PF24, respectively. The device command %? causes the terminal to display the current status of the program-function keys and device-command settable options. Program-function keys defined as device commands must use the "%" device-command character regardless of the current device-command character setting.

Normally, when a program-function key is used, the cursor will be pointing to a null input area. However, if the cursor is pointing to another line, that line will be written into the input area. This line can then be edited and sent to the system via the ENTER key. In this manner, a previous line that is longer than 77 characters can be edited before reentering it.

Terminating the Session

The terminal session can be terminated by entering a SIGNOFF command while in MTS command mode (# prefix). The system produces the statistical summary, including charges, for the session and displays it on the screen (Panel 4). The summary will remain for several seconds and then the screen is erased and the block MTS is displayed again in preparation for the next session. The SYS REQ key can be used after signoff to allow extended viewing of the signoff statistics for up to 5 minutes.

Since the terminals are hard-wired to the system, it is not possible to "hang-up" the phone to terminate a session. Turning off the power does not terminate the session or cause the current user to be signed off. The %QUIT device command can be used to force a signoff.

The CLEAR key can be used to terminate the display of the signoff statistics.

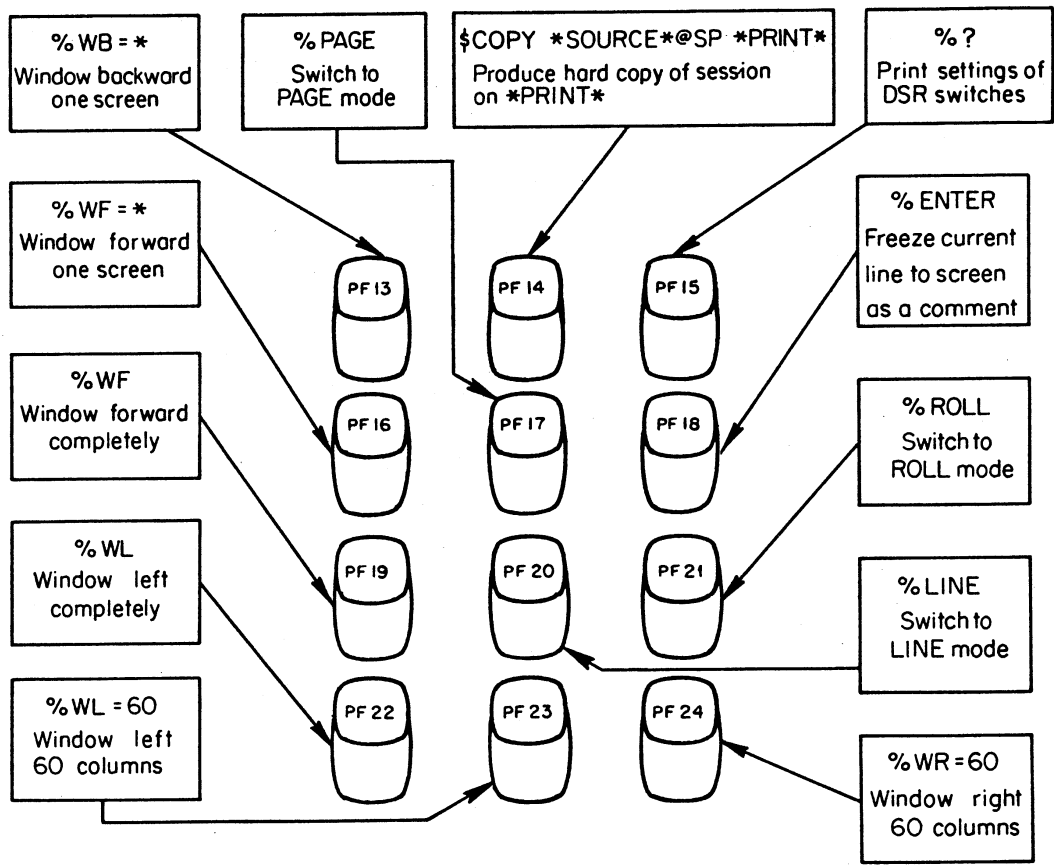


Figure 4. Initial Definitions of Program-Function Keys

APPENDIX A: IBM 3278 DEVICE COMMANDS

Device commands must begin with the current "device-command character", initially a percent sign "%", immediately followed by the command name or a permissible abbreviation. The minimum abbreviation for each command is underlined in the command descriptions. The command can be followed by zero or more parameters, depending on the particular command. Delimiters such as equal signs "=", commas and semicolons are ignored and can be replaced with blanks.

To enter a data line that begins with a "%", the user must either use the literal-next character, change the device-command character via the %DCC device command, or double the device-command character "%%".

This appendix describes the device commands presently available. The following notation is used to describe commands:

<u>UPPERCASE</u>	- must appear as shown
<u>lowercase</u>	- replace appropriately
	- separates alternative forms
{ }	- encloses alternative forms
[]	- encloses optional parameters
_	- indicates command abbreviations
...	- indicates repeatable fields

For most commands which have an optional {ON|OFF} parameter, the following rule applies: if ON or OFF is not specified, the status will be changed to ON if it was previously OFF and to OFF if it was previously ON. For example,

HEX [= {ON|OFF}] ["x"]

indicates that the command can have the form %HEX=ON or %HEX=OFF or %HEX, and any of these forms can optionally be followed by a character in quotes. In addition, the semicolon and the equal sign can be replaced with blanks. For example, HEX=ON "?" and HEX ON "?" are equivalent forms of the same command. H is a legitimate abbreviation of %HEX=.

Summary of Device Commands

<u>ACTIVITY</u> [= {ON OFF}]	task status display
<u>ATTN</u>	generate an attention interrupt.
<u>BLANK</u>	nondisplayed input area
<u>BRIGHT</u>	brighten next input or output window
<u>BUFFER</u> [=n]	change size of conversation buffer
<u>CBEGIN</u>	specify first line of *MSOURCE*@SP
<u>CEND</u>	specify last line of *MSOURCE*@SP
<u>COMMENT</u> string	displays any character string
<u>CONTRACT</u> [=n]	decrease input area size
<u>DCC</u> ="x"	change device-command character

<u>DC?</u>	display mode, switches, tabs, etc.
{ <u>DEQUEUE</u> <u>DQ</u> } [=n]	remove last "n" lines from input queue
<u>DUPLEX</u> [= {ON OFF}]	duplex input
<u>EMPTY</u>	empty (clear) the conversation buffer
<u>ENTER</u>	enter input area to conversation buffer
<u>EOF</u>	send end-of-file signal
{ <u>EQ</u> <u>MTQ</u> }	empty the input queue
<u>EXPAND</u> [=n]	increase input area size
<u>FAST</u>	FAST output mode
<u>FLIP</u>	switch terminal to other job
<u>FOOTER</u> [=] text	display text at bottom of screen
<u>FREEZE</u> =n	freeze top "n" lines of screen
<u>GRAB</u>	start a new MTS job
<u>HEX</u> [= {ON OFF}] ["x"]	hexadecimal input editing
<u>HIST</u> [=n]	conversation history lines
<u>INSERT</u> =string	insert a string into input area
{ <u>JF</u> <u>JB</u> } [=n]	jump forward or backward
<u>JOBID</u>	displays device and task number
{ <u>JSET</u> <u>JRESET</u> <u>JCLEAR</u> }	set or reset jump windows
<u>KEYBOARD</u> ={UC LC}	input case translation
<u>LCI</u> [= {ON OFF}]	lowercase input translation
<u>LINE</u>	LINE output mode
<u>LNC</u> ={"x" OFF}	literal-next character
<u>ORL</u> [=n]	change GDINFO output record length
<u>PAGE</u>	PAGE output mode
<u>PAUSE</u>	wait on the next read or write
<u>PF</u> n=definition	program-function key definition
{ <u>PFSAVE</u> <u>PFRESTORE</u> <u>PFDELETE</u> }=name	save or restore PF key definitions
<u>PFRESTORE</u> =DEFAULT	restore the PF definitions to default
<u>PF?</u>	display program-function keys
<u>QUEUE</u>	display the input queue
<u>QUIT</u>	emergency signoff
<u>RATE</u> [=n]	output speed (n=1-9)
<u>REFRESH</u>	rewrite a clear input area
<u>RESET</u>	restore device command parameters to SIGNON state
<u>ROLL</u>	ROLL output mode
<u>SNS</u>	return information about the session
{ <u>TABI</u> <u>TABS</u> <u>TI</u> } [= {ON OFF}] ["x"] [{A F P t,...}]	enable or disable input tab stops
{ <u>TABO</u> <u>TO</u> } [= {ON OFF}] ["x"] [{A F P t,...}]	enable or disable output tab stops
<u>THAW</u>	undo effect of FREEZE
<u>TOP</u> [= {ON OFF}]	change meaning of carriage control "1"
<u>UCI</u> [= {ON OFF}]	uppercase input translation
<u>WAIT</u>	inhibit 30-minute time-out
{ <u>WB</u> <u>WF</u> <u>WR</u> <u>WL</u> } [= {n *}]	move window
<u>WINDOW</u> =n	display window number "n"
<u>YOUTAKEIT</u>	redo the last interrupted user action ¹
<u>ZIP</u>	move cursor to end of input area
<u>?</u>	combination of DC? and PF?

ACTIVITY

Command: ACTIVITY [= {ON|OFF}]

Purpose: To produce a full-screen display of the task's status and resource utilization while a program or command is executing. The activity display is produced one second after entering the ON state and is updated every three seconds. The display is automatically terminated whenever it is necessary to rewrite the screen normally. The activity display contains an input area at the bottom of the screen that can be used for normal input; the PF keys and other keys can be used with their normal effect. The format of the display is as follows:

```
DT  EXQ PRT PCH BP AB BPH LL AL   VP  RP  DPA PA  DA  CA  % PI  Q
nn.nn nnn nnn nnn nn nn nnn nnn nnn nnnnn nnnn nnnn nnn nnn nnn nn.nn nn
```

```
Id: <ccid>                    Proj: <proj>                    Task: <job#>
Virtual Pages: <vp>            Real Pages: <rp>                Real Pages Needed: <nnnn>
Page-ins: <nnnn>                Disk I/O: <nnnn>                Time: <date>
CPU Time: <time>                (nn% Supervisor)                Elapsed Time: <time>
Status:                        <xxxxxxxxxxxx>; PSW=<addr>
Devices:                        DSnn <dev2> <dev3> ... <devn>
Pages Printed: <pages>        Cards Punched: <cards>        Real Pages Needed: <rpn>
CPU Time: <time> SSSPPP
Memory: <vp> RRRDDD
Page-ins: <nn> PPPPPP
Disk I/O: <nn> DDDDDD
```

where

```
nn = same as printed by $SYSTEMSTATUS LOAD command (2nd line)
time = minutes and seconds (CPU Time and Elapsed Time fields)
xx = same as printed by $SYSTEMSTATUS TASKS command (Status field)
S = 0.05 seconds of supervisor state time (CPU Time field)
P = 0.05 seconds of problem state time (CPU Time field)
R = 8 real memory pages (Memory field)
D = 8 d pages on auxiliary paging device (Memory field)
P = each character represents one page from the auxiliary paging
```

device (Page-ins field)
D = each character represents one page from the auxiliary paging
device (Disk I/O field)

Example: %A

The output will be as follows:

DT	EXQ	PRT	PCH	BP	AB	BPH	LL	AL	VP	RP	DPA	PA	DA	CA	%	PI	Q
20.00	4	42	0	2	0	83	0	79	5658	1318	3046	14	19	76	79.41	4	

Id: WABC Proj: WABC Task: 3594
Virtual Pages: 30 Real Pages: 21 Real Pages Needed: 1412
Page-ins: 32 Disk I/O: 345 Time: 23:10:15 Mon Mar 13/84
CPU Time: :16.59 (54% Supervisor) Elapsed Time: 161:41
Status: DORMANT; PSW=00000000
Devices: DS05
Pages Printed: 0 Cards Punched: 0 Real Pages Needed: 22
CPU Time: 0.00
 Memory: 30 RRRD
Page-ins: 0
Disk I/O: 0

ATTN

Command: ATTN

Purpose: To generate an attention interrupt. This is the same as the fixed definition of the PA1 key.

Example: %AT

BLANK

Command: BLANK

Purpose: To indicate that the next input line should not be displayed on the screen. This is useful for maintaining password security.

A second BLANK command can be used to cancel the effect of the previous BLANK command, for example to enable printing of the password at SIGNON time.

Examples: %B
%BLANK

BRIGHT

Command: BRIGHT

Purpose: To cause the next input or output line to be displayed at high intensity. A line so brightened will retain this attribute until it leaves the buffer. Another %BRIGHT command, entered before the input or output line in question, will reverse the effect of the preceding %BRIGHT command.

Example: %BR

BUFFER

Command: BUFFER [=n]

Purpose: To change the size of the conversation buffer to "n" pages (4096 characters per page). Although "n" can be any number between 0 and 32 (inclusive), it is rounded up to the nearest power of 2. If BUFFER=0 is specified, a half-page buffer is allocated. The buffer allocated will hold 1/32nd as many lines as it holds characters. The %BUFFER command cannot be entered asynchronously, i.e., it is valid only through the CONTROL subroutine or when the system is awaiting input. The default buffer size is 4 pages.

Examples: %BU=32
%BUFFER=4

CBEGIN, CEND

Commands: CBEGIN
 CEND

Purpose: To specify the first and last lines to be read from *MSOURCE*@SP. After the %CBEGIN command is executed, the first line after the last line currently displayed on the screen becomes line one of the pseudofile read with the @SP special modifier. The %CEND command makes the last line on the screen the last line in the pseudofile. When the %CBEGIN command is in effect, the normal header line is not read. Whenever an end-of-file is read with the @SP modifier, the CBEGIN and CEND lines are reset.

These commands can be used to facilitate a nonoverlapping hard copy of an entire terminal session, as follows (assume that the current window is at the end of the buffer (%WF)):

- (1) Issue the %CEND command.
- (2) Issue \$COPY *MSOURCE*@SP ... (or use PF2).
- (3) Issue the %CBEGIN command.
- (4) Repeat (1) through (3) often enough to prevent the buffer from wrapping around.
- (5) End the session with (1) and (2).

This procedure will produce a nonoverlapping copy that omits the individual \$COPY command lines themselves.

Examples: %CB
 %CE

COMMENT

Command: COMMENT string

Purpose: To put any character string on the screen. For example, this command can be used with the \$CONTROL command to produce comments on the terminal screen from a sigfile.

Examples: %C TABS AT 10,16; HEX ON WITH "
 %COMMENT HAVE A NICE DAY

DCC

Command: `DCC="x"`

where "x" is any character.

Purpose: To redefine the "device-command character." This character must be prefixed to all device commands. It is initially set to a percent sign "%".

Device commands entered from a PF key must use the "%" device-command character regardless of the setting of the current device-command character.

Examples: `%DC "*"
*DC "%
%DCC="|"`

DC?

Command: `DC?`

Purpose: To display the current values of output mode, rate, duplex switch, history, DCC, buffer, ORL, hex switch and character, tab settings; and, if not set to their default values, top switch, LNC, and case.

Examples: `%DC?`

DEQUEUE

Command: {DEQUEUE|DQ} [=n]

where "n" is a positive integer.

Purpose: To remove the last "n" lines of input which have been entered asynchronously (typed ahead) and which have not yet been read by the system. If there are fewer than "n" such lines, all lines are dequeued. If "n" is omitted, one line is dequeued.

Examples: %DEQUEUE=3
 %DQ

DUPLEX

Command: DUPLEX [= {ON|OFF}]

If the ON or OFF parameter is not given, the status will be changed to ON if it was previously OFF and to OFF if it was previously ON.

Purpose: To specify whether input which the user is typing when an output operation interrupts the input process should be preserved. If %DUPLEX=ON then the output operation can momentarily lock the keyboard but the contents of the input area and the position of the cursor within it will not be affected. If, however, %DUPLEX=OFF then the line which is being typed will be lost. The default at signon time is ON.

Examples: %D
 %DUPLEX=OFF

EMPTY

Command: EMPTY

Purpose: To empty (clear) the conversation buffer of all lines.

Example: %EM

ENTER

Command: ENTER

Purpose: To add the current contents of the input area to the conversation buffer without returning the line to the system. This is useful for saving a long input line so that it can be reentered later, perhaps after the user realizes that another input line must be entered first. This command is only useful when defined by a PF key, since the input area is cleared following the use of the ENTER key.

Example: %EN

EOF

Command: EOF

Purpose: To send an end-of-file signal. This is the same as the fixed definition of the PA2 key.

Examples: %EOF
%E

EQ

Command: {EQ|MTQ}

Purpose: To empty the queue of input lines which have been entered asynchronously (typed ahead) and which have not yet been read by the system.

Examples: %EQ
 %M

EXPAND, CONTRACT

Commands: EXPAND [=n]
 CONTRACT [=n]

Purpose: To change the minimum size of the input area. %EXPAND increments the size by "n" lines; %CONTRACT decrements the size by "n" lines. The input area cannot be made smaller than the normal two lines at the bottom of the screen, nor larger than 39 lines. %EXPAND is useful when it is necessary to enter a very long input line (longer than about 145 characters--the normal size with no footers). The default value for "n" is 1. The maximum length for an input line is 255 characters.

Examples: %EXPAND=2
 %CONTRACT

FAST

Command: FAST

Purpose: To redefine the output mode of the terminal. In FAST mode, output lines are recorded in the conversation buffer but are not written to the screen until an input line is requested by MTS. If more than a screenful of output is generated between input requests, not all of it is displayed on the screen and some of it may be lost out the top of the conversation buffer. The initial mode is LINE. For more information, see the section "Output Modes."

Examples: %F
%FAST

FOOTER

Command: FOOTER=text

where "text" is any arbitrary character string.

Purpose: To specify footer text which will be displayed at the bottom of the screen, immediately preceding the window number. This might be used via the CONTROL subroutine by a program which wanted to inform its users that it was active.

Examples: %FO=I HAVE TAKEN OVER
%FOOTER=QQSV

FREEZE

Command: FREEZE=n

Purpose: To cause the top "n" lines currently on the screen to remain on the display in spite of any vertical scrolling operations, further input, output, etc. The prefix of the last line frozen is brightened if it exists and is nonblank. "n" must be greater than zero and less than the number of lines currently displayed, or currently frozen if a %FREEZE command is currently active. The frozen lines can be edited and returned as input, windowed horizontally, etc., in the normal manner. If the frozen lines leave the conversation buffer due to wraparound, the freeze is deactivated. The effect of a FREEZE command can be deactivated explicitly by using the THAW command.

Example: %FR=10

GRAB, FLIP

Commands: GRAB
FLIP

Purpose: To allow the running of two MTS jobs simultaneously from the same terminal. The GRAB command starts a second job, and displays the signon window for the new job. The user can then sign on and proceed normally. The two jobs will be completely independent. The %GRAB command may fail if the system load is too high.

The FLIP command can be used to switch the terminal back and forth between jobs. Only one job can do input or output to the terminal at a time. The job which is detached from the terminal may continue executing, but as soon as it tries to do input or output to the terminal, it will wait until a FLIP command is issued.

When the detached job is awaiting input, a brightened asterisk (*) will appear in the lower right-hand corner of the screen to the left of the window number.

HEX

Command: HEX [= {ON|OFF}] ["x"]

where "x" is any character not already assigned to a special editing function. If no parameters are given, the status will be changed to ON if it was previously OFF and to OFF if it was previously ON. For example, when hexadecimal editing is off, %HEX and %HEX=ON have the same effect.

Purpose: To enable or disable the hexadecimal input editing facility and to define the delimiter to be used. The default at SIGNON time is OFF with a delimiter of prime (').

Examples: %H ", "
%H=OFF
%HEX=ON ";"

HIST

Command: HIST [=n]

where "n" is any number from 1 to 41.

Purpose: To specify the number of lines of conversation history which are redisplayed at the top of the screen when the 41st line has been used in line mode. The default value is 14.

Examples: %HI 5
%HIST=17

INSERT

Command: INSERT=string

Purpose: To insert a string at the current cursor position in the input area. The cursor will be positioned following the inserted string. This command is only useful when defined by a PF key.

Examples: %I=SCARDS=
%INSERT=*PRINT*

JF, JB

Commands: JF [=n]
JB [=n]

Purpose: To jump forward or backward to the "n"th jump point relative to the current window. The default for "n" is 1. These commands are similar to the %WF and %WB commands except that they jump to preset places set by the %JSET device command.

Examples: %JF
%JB=2

JOBID

Command: JOBID

Purpose: To display the device name, task number, signon ID, and current time and date.

Examples: %J
%JOBID

JSET, JRESET, JCLEAR

Commands: JSET
JRESET
JCLEAR

Purpose: The %JSET command defines the current window as a jump point (a type of vertical tab setting) for use with the %JF and %JB device command. A maximum of 10 jump points can be in effect at any one time. If more are defined, the oldest jump points (in conversation buffer order) are deleted. The first and last lines in the buffer are always defined as jump points, but do not contribute to the maximum of 10.

The %JRESET command removes the current window from the table of jump points. This command is invalid if the current window is not in the table.

The %JCLEAR command clears all currently defined jump points.

Examples: %JS
%JR
%JC

KEYBOARD

Command: KKEYBOARD={UC|LC}
UCI=[{ON|OFF}]
LCI=[{ON|OFF}]

Purpose: To disable or enable automatic uppercase translation on input lines. The initial state is KEYBOARD=UC, that is, all lowercase input letters will be translated to uppercase. By setting KEYBOARD=LC, the user can enter true lowercase alphabetic data.

Unlike many other device commands, the %UCI and %LCI device commands do not toggle the setting of the UC/LC switch if ON or OFF is not specified. ON is always assumed.

Examples: %K LC
%UC
%K=UC

LINE

Command: LINE

Purpose: To redefine the output mode. The initial mode is LINE. For more information, see the section "Output Modes."

Examples: %L
%LINE

LNC

Command: `LNC={"x"}|OFF}`

where "x" is any character that is not already assigned to a special editing function.

Purpose: To respecify the "literal-next character" or set it back to its original undefined state. This character is useful when you want to enter a special editing character without its usual significance. The default is OFF.

Examples: `%LN "!"`
`%LNC="φ"`

ORL

Command: `ORL [=n]`

Purpose: To change the maximum output record length returned by the GDINFO subroutine to "n"; "n" must be in the range of $0 < n \leq 255$. This command does not affect the behavior of the terminal (which always has a maximum output record length of 255), but many commands and programs use the GDINFO length to break up their output streams. The normal ORL setting is 77, the number of characters displayed on the screen with a one-character prefix. If no parameter is specified, the default ORL setting 77 is assumed.

Example: `%ORL=60`

PAGE

Command: PAGE

Purpose: To redefine the output mode. In PAGE mode, the terminal automatically enters a PAUSE state when the screen has been filled and will continue only when the SYS REQ key is pressed. The default at SIGNON time is LINE mode. For more information, see the section "Output Modes."

Examples: %P
%PAGE

PAUSE

Command: PAUSE

Purpose: To force the executing program or command to wait the next time a read or write operation is directed to the terminal screen. This function is normally performed by the SYS REQ key. For more information, see the section "Pause/Continue."

Examples: %PAUSE
%PAU

PF

Command: PFn=definition

where "n" is an integer from 1 to 24.

Purpose: To redefine a program-function key to a given string. If the definition is a device command (e.g., PFn=%L), then the device-command character used in the definition must be a "%" and not the current DCC. To define a PF key that starts with a percent sign that is not to be interpreted as a device-command character, two percent signs must be used.

Unlike other device commands, the "definition" portion of the PFn=definition device command is never converted to uppercase regardless of the setting of the UCI switch (LCI or K={UC|LC}). When a PF key is actually used, the data line or device command defined will be converted according to the setting of the UCI switch in effect at that time.

Examples: %PF6 %H
%PF2 C *MSOURCE*@SP TO *PRINT*
%PF10=CANCEL

PFSAVE, PFRESTORE, PFDELETE

Commands: PFSAVE=name
PFRESTORE=name
PFDELETE=name

Purpose: The %PFSAVE command saves the current PF key definitions under "name" so that they can be restored later by the %PFRESTORE command. "name" can be any string from 1 to 8 characters. This command cannot be entered asynchronously, i.e., it is valid only through the CONTROL subroutine or when the system is awaiting input. If "name" already exists, the new definition replaces the previous definition.

The %PFRESTORE command restores the PF key definitions saved under "name". The special name DEFAULT can be used with the %PFRESTORE command to set the PF definitions to their original state at signon unless a PF cluster with the name DEFAULT was explicitly created via the %PFSAVE command.

The %PFDELETE command deletes the PF key definition saved under "name". This command cannot be entered asynchronously.

These commands may, for example, be used to expand the scope of PF keys by defining several clusters of PF key definitions, each of which has keys that restore other clusters. Thus, some PF key operations may require two PF key strokes instead of one stroke.

Examples: %PFS STANDARD
%PFSAVE ALTERNATE
%PFR STANDARD
%PFD ALTERNATE

PF?

Command: PF?

Purpose: To display the current status of the program-function keys and the names of any PF cluster definitions.

Example: %PF?

QUEUE

Command: QUEUE

Purpose: To display the queue of input lines which have been entered asynchronously and which have not yet been read by the system.

Examples: %QUEUE
%Q

QUIT

Command: QUIT

Purpose: To immediately sign the user off. This is equivalent to hanging up the phone for a dial-up terminal. It is useful, for example, if a program gets into a loop trapping attention interrupts.

Example: %QU

RATE

Command RATE [=n]

where "n" is any integer between 1 and 9. The default value is 5.

Purpose: To respecify the output speed. The rate can vary from 1 to 9 with 9 being the fastest. If "n" is omitted, the rate is reset to 5.

Examples: %RA 8
%RATE

REFRESH

Command: REFRESH

Purpose: To rewrite the screen with a clear user input area. This is the same as the fixed definition of the CLEAR key.

Example: %REF

RESET

Command: RESET

Purpose: To restore to their original SIGNON state all parameters and functions that can be changed by a device command, except the conversation buffer size and PF cluster definitions.

Examples: %RE
%RESET

ROLL

Command: ROLL

Purpose: To redefine the output mode. In ROLL mode each new input or output line added to the conversation buffer causes the screen to be erased and the latest screenful from the buffer to be displayed. The output appears to be rolling up the screen. The initial mode is LINE. For more information, see the section "Output Modes."

Examples: %R
%ROLL

SCROLLING

Command: {WB|WF|WR|WL} [{=n|*}]

where "n" is any integer between 1 and 255 for WR and WL, or any integer between 1 and the maximum number of lines in the conversation buffer.

Purpose: To scroll the display window to any part of the conversation buffer. An attempt to move the window beyond the boundaries of the conversation buffer will not be honored. The WR=n command causes the window to be displaced to the right by "n" characters and the WL=n command moves the window left. The WF=n command moves the window forward by n lines and the WB=n command moves the window backwards. The WF=* command moves the window forward by the number of lines currently on the screen less the number of frozen lines and the number of lines in the input area. The WB=* command moves the window backward by the number of lines currently on the screen less the number of frozen lines and the number of lines in the input area. If no parameter is specified the extreme movement is assumed.

Examples: %WR
%WL=60
%WB
%WF=20

SNS

Command: SNS

Purpose: To return information about the current terminal session. This command can be issued only by calling the CONTROL subroutine (see MTS Volume 3, System Subroutine Descriptions). The calling program must provide an information area (the first argument on the call to CONTROL) which has "SNS" as the first three characters. The length (second argument) must give the length (in bytes) of the information including the three leading characters "SNS". The format of the information returned to the calling program is as follows:

<u>Byte</u>		<u>Function</u>
0-2	SNS	(supplied by calling program)
3	Unused	(reserved for future use)
4-7	DSnn	(MTS device name)
8-11	3270	(MTS device type)
12-15	DSn	(control unit name)
16-19	xxxx ¹	(control unit type)
20-23	DSnn	(port type and number)
24-47	DSxx3270	(answerback code)
48-55	xxxxxxxx ²	(terminal type)

¹Possible values are:

3272
3274
3036
3066

²Possible values are:

3277MOD1	3278-2L
3277MOD2	3278-3L
3278MOD1	3278-4L
3278MOD2	3278-5L
3278MOD3	3066
3278MOD4	3066

The public file *SENSEDESECT contains a subroutine dsect description of the information returned by the CONTROL subroutine.

TABS

Command: {TABI|TABS|TI} [= {ON|OFF}] ["x"] [{A|F|P|t,...}]
 {TABO|TO} [= {ON|OFF}] ["x"] [{A|F|P|t,...}]

where "x" is any character that is not already assigned to a special editing function. "t" is an integer value between 1 and 255. Up to 9 "t" values can be specified, in ascending order. If no parameters are given, the status will be changed to ON if it was previously OFF and to OFF if it was previously ON.

Purpose: To enable or disable input (TABI/TABS/TI) or output (TABO/TO) tabulation editing and to specify the logical tab character and tab stops. Both input and output tab editing are initially off with no tab stops defined and with no tab character defined. Three special sets of tab stops are predefined. They correspond to the accepted statement field positions of 360/370 Assembler Language, FORTRAN, and PL/I and are denoted by A, F, and P, respectively, in the tab stop portion of the device command. Their implied values are:

A=10,16,35
F=7,73
P=5,10,15,20,25,30,35,40,45

Examples: %T ON 10 20 30
 %T 10 20 30
 %TABO OFF "/"
 %T "*" A
 %T
 %TABI=ON "-" F

THAW

Command: THAW

Purpose: To deactivate any frozen lines placed on the screen by using the FREEZE command.

Examples: %THAW
%TH

TOP

Command: TOP [= {ON|OFF}]

If the ON or OFF parameter is not given, the status will be changed to ON if it was previously OFF and to OFF if it was previously ON.

Purpose: To specify the interpretation of the carriage control '1'. If TOP mode is OFF, which is the default, 2 blank lines are inserted preceding a carriage control '1'. If ON, the most recent line containing a carriage control '1' will be forced to be at the top of the screen, whenever it is displayed. A line loses this property whenever it moves off the screen, whenever the CLEAR key is pressed, or whenever the next such line appears in the output stream. In PAGE mode the terminal will PAUSE upon receipt of a carriage control '1', and will display the current page, just as if the screen were full.

Examples: %TOP=ON
%TOP

WAIT

Command: WAIT

Purpose: To inhibit the 30-minute time-out used while waiting for input or waiting in pause mode. Normally, the user will be signed off after 30 minutes. Use of the WAIT command will prevent this for one operation only. The next time the terminal goes into input wait or pause mode, the time-out will again be in effect.

WB, WF, WL, WR

Commands: WB [= {n|*}]
WF [= {n|*}]
WL [= {n|*}]
WR [= {n|*}]

where "n" is any integer between 1 and 255.

Purpose: To allow the display window to be "scrolled" to any part of the conversation buffer. An attempt to move the window beyond the boundaries of the conversation buffer is not valid. The WR command causes the window to be displaced to the right by "n" characters and the WL command moves the window left. The WF command moves the window forward by "n" lines and the WB command moves the window backwards.

%WB=* and %WF=* mean scroll backward or forward one screen, respectively. %WB and %WF mean scroll all the way backward or forward, respectively. %WL=* and %WL move the window all the way to the left. %WR=* and %WR move the window all the way to the right.

Examples: %WR=60
%WF 20
%WL *
%WB

WINDOW

Command: WINDOW=n

where "n" is an integer between 1 and 999.

Purpose: To display the window designated by "n", where "n" refers to the window number which is displayed in the lower right-hand corner of the screen. If a WR or WL command has been executed, the window number will be followed by a space and the horizontal offset specified in the WR or WL command.

Examples: %W 5
%WINDOW=147

YOUTAKEIT

Command: YOUTAKEIT

Purpose: To reinterpret the last user action that caused an attention interrupt. This command is only useful to programs that do binary input/output to the terminal (see Appendix C). The attention control bits are first reset to their default value and to the last hardware attention interrupt to be reinterpreted. In general, the caller should not reset the attention ID (AID) configuration before calling the CONTROL subroutine with the %YOUTAKEIT command. This command can be used only by calling the CONTROL subroutine (see MTS Volume 3, System Subroutine Descriptions).

ZIP

Command: ZIP

Purpose: To move the cursor to the end of the current nonnull contents of the input area. This command is only useful when defined by a PF key.

Example: %Z

?

Command: ?

Purpose: To accomplish %DC? and %PF? in one device command.

Example: %?

APPENDIX B: IBM 3278 CHARACTER CODES

<u>Graphic</u>	<u>Hex Code</u>	<u>Graphic</u> ⁶	<u>Hex Code</u>	<u>Graphic</u>	<u>Hex Code</u>
DUP	1C ⁸	a	81	A	C1
FIELD MARK	1E ⁸	b	82	B	C2
Space	40	c	83	C	C3
φ	4A	d	84	D	C4
.	4B	e	85	E	C5
<	4C	f	86	F	C6
(4D	g	87	G	C7
+	4E	h	88	H	C8
	4F	i	89	I	C9
&	50	{	8B ^{3 7}	J	D1
!	5A	j	91	K	D2
\$	5B	k	92	L	D3
*	5C	l	93	M	D4
)	5D	m	94	N	D5
;	5E	n	95	O	D6
⌋	5F	o	96	P	D7
-	60	p	97	Q	D8
/	61	q	98	Q	D8
	6A ^{4 7}	r	99	R	D9
,	6B	`	9A ^{2 7}	\	E0 ^{1 7}
%	6C	}	9B ^{3 7}	S	E2
	6D	~	A1 ^{5 7}	T	E3
>	6E	s	A2	U	E4
?	6F	t	A3	V	E5
'	79 ^{1 7}	u	A4	W	E6
:	7A	v	A5	X	E7
#	7B	w	A6	Y	E8
@	7C	x	A7	Z	E9
'	7D	Y	A8	0	F0
=	7E	z	A9	1	F1
"	7F	\	BA ^{2 7}	2	F2
		{	C0 ^{1 7}	3	F3
		}	D0 ^{1 7}	4	F4
				5	F5
				6	F6
				7	F7
				8	F8
				9	F9

¹Output only. This is the standard IBM EBCDIC code for this graphic (grave "`", reverse slant "\", left brace, right brace). MTS EBCDIC uses a different code.

²This is the standard MTS EBCDIC code for this graphic (grave "`", reverse slant "\"). IBM EBCDIC uses a different code.

³This is the standard MTS EBCDIC and IBM print-train code for this graphic (left brace, right brace). IBM EBCDIC uses a different code.

⁴The name for this graphic is broken vertical line and it is different from the graphic logical OR. ASCII terminals that only have a broken vertical line and no logical OR map this into logical OR. Since the IBM 3278 terminals support both the broken vertical line and logical OR, no such mapping is done.

⁵ASCII terminals which have no logical NOT normally map the tilde "~" into the EBCDIC code for logical NOT. Since the IBM 3278 supports both logical NOT and tilde, no such mapping is done. The EBCDIC code for tilde results in the printing of the degree symbol "°" when printed with the TN character set.

⁶On terminals that are not equipped with the lowercase feature, lowercase letters will be displayed in uppercase.

⁷These graphics are only available on the IBM 3278 terminals. They cannot be entered directly from the IBM 3277, 3036, 3066, or Courier C-270 terminals. On output, they are mapped into the question mark "?".

⁸FIELD MARK and DUP cannot be entered from the IBM 3066 terminal. On output, they are mapped into the question mark "?".

APPENDIX C: BINARY INPUT/OUTPUT ON THE IBM 3278

A facility is provided for input/output to the IBM 3278 which bypasses the buffering and editing procedures in the device support routines. Anyone interested in using this facility should be familiar with the IBM Systems publication, IBM 3270 Information Display System Component Description, form GA27-2749.

This facility allows a program to erase the screen, to write messages starting at the current buffer address (which can be specified by the program), to read input from the terminal without it being edited by the support routines, and to intercept signals from the different function keys according to program specification.

Because these features are entirely dependent on the IBM 3278 terminal and because the Computing Center does not guarantee that IBM 3278 terminals will always be available in the future, the use of these features by user programs is discouraged. Instead, programmers are encouraged to use the MTS Screen Support Routines.

The following describes the procedure for intercepting function key signals:

- (1) To intercept an attention signal from the terminal, the ATTNTRP subroutine (see MTS Volume 3, System Subroutine Descriptions) is called as usual. This will normally only give the program control when the PA1 key is pressed.
- (2) To control which other function keys are intercepted by the ATTNTRP subroutine, the CONTROL subroutine (see MTS Volume 3) is called specifying a 32-bit string where the first ten bits are zero and the last 22 bits correspond to the Attention ID (AID) configurations as listed in the IBM Systems manual. Bit 31 should correspond to the "Operator Identification Card Reader - Enter" configuration.

Example:

```
CALL CONTROL, (BITS, LEN, FDUB, RET)
      .
      .
BITS   DC   X'00', BL3'000010011111110110110010'
```

- (3) A binary read is done after getting control from ATTNTRP to get the AID and data.

The possible kinds of reads and writes and the I/O modifiers used to specify them are as follows:

READ

@BIN read modified
@BIN,@~CC read buffer
@BIN,@MCC erase all unprotected

WRITE

@BIN write
@BIN,@~CC erase/write
@BIN,@MCC erase all unprotected

To write a message on the screen, the program must specify a write control character (WCC - see the IBM Systems manual) followed by the message. This will write the message on the screen starting at the current buffer address. In order to specify a starting address for the message the program must follow the WCC with a set buffer address (SBA - X'11') order and a two-byte code for the buffer address according to the list in the IBM Systems manual.

The cursor address and the buffer addresses which are transferred to the program when a read command is issued are also coded according to specifications given in the IBM Systems manual. The following two subroutines can be used to encode and decode 3278 buffer addresses.

```
*
*      ENCODE 3278 BUFFER ADDRESS
*
*      BAL    RE,ENBA
*            R0 CONTAINS COLUMN NUMBER
*            R1 CONTAINS ROW NUMBER
*            R2 POINTS TO NEXT BYTE OF OUTPUT AREA AND IS UPDATED
*
ENBA      MVI    0(R2),SBA          INSERT THE SET BUFFER ADDRESS ORDER
          MH     R1,=H'80'         CALCULATE RELATIVE SCREEN POSITION
          AR     R0,R1
          SRDL  R0,6              ISOLATE THE LEADING 6 BITS
          STC   R0,1(,R2)         AND PUT THEM IN OUTPUT AREA
          SRL   R1,26             GET LAST 6 BITS
          STC   R1,2(,R2)         AND PUT THEM AWAY
          TR    1(2,R2),SBATABLE  TRANSLATE THEM INTO EBCDIC
          LA    R2,3(,R3)         UPDATE OUTPUT POINTER
          BR    RE                AND RETURN
          SPACE 3
SBATABLE  DS    0X
          DC    C' ABCDEFGHIJ.<(+|'
          DC    C' &&JKLMNOPQR!$*);~'
          DC    C' -/STUVWXYZ ,%_>?'
          DC    C' 0123456789:#@' '='
          SPACE 3
*
*      DECODE 3278 BUFFER ADDRESS
*
```

```

*          BAL   RE,DEBA
*          R0 WILL CONTAIN COLUMN NUMBER
*          R1 WILL CONTAIN ROW NUMBER
*          R2 POINTS TO INPUT SBA OR AID AND IS UPDATED
*
DEBA      NC    1(2,R2),=X'3F3F'   TRANSLATE ADDRESS FROM EBCDIC
          SR    R1,R1
          IC    R1,1(,R2)         GET FIRST 6 BITS
          SLL   R1,6              MOVE THEM OVER TO MAKE ROOM
          EX    R1,DEBAOR         PAY BACK TWO BITS
          IC    R1,2(,R2)         GET LAST 8 BITS
          SR    R0,R0
          D     R0,=F'0'         CALCULATE ROW AND COLUMN
          LA    R2,3(,R2)        UPDATE INPUT POINTER
          BR    RE                AND RETURN
          SPACE 2
DEBAOR   OI    2(R2),0          DULY EXECUTED
          EJECT

```

Four translate (TR) tables and two translate-and-test (TRT) tables are available as part of the resident system to assist programs that perform binary I/O to the IBM 3278 terminal. These are as follows:

- TRIN3278 - A translate table that maps the codes used the IBM 3278 terminal into EBCDIC.
- TROT3278 - A translate table that maps the EBCDIC codes used by MTS into the codes used by the IBM 3278 terminal. MTS EBCDIC codes that cannot be displayed are mapped into the question mark "?".
- TRT3278 - A translate-and-test table that contains the value zero for those characters that can be displayed on the IBM 3278 terminal, the value one for those characters which cannot be displayed, and the value two for those characters which can be displayed but for which the code is not the standard MTS EBCDIC code used for the graphic in question.

The tables TRI3278A, TRO3278A, and TRT3278A perform similar functions for the Lee Data terminal. The tables TRIN3277, TROT3277, and TRT3277 perform similar functions for the IBM 3277, IBM 3036, IBM 3066, and Courier C-270 terminals.

Because there is a one-to-one mapping between the codes produced by the IBM 3278, Lee Data, IBM 3277, IBM 3036, IBM 3066, and the Courier C-270 and the codes used by MTS, the tables TRIN3278, TRI3278A, and TRIN3277 perform no real actual translations and are provided only for symmetry.

APPENDIX D: IBM 3278 ROUTINES RETURN CODES

The following return codes are generated by the IBM 3278 terminal routines.

Input:	0	Successful return
	4	End-of-file (PA2 key or %EOF device command)
	8*	Error during binary input
Output:	0	Successful return
	4	End-of-file (ending line number exceeded)
	8*	Error during binary output
CONTROL:	0	Successful return
	4	Should not occur
	8	Invalid CONTROL command

*Return to the calling program is made only if the I/O call specified the ERRRTN I/O modifier.

APPENDIX E: IBM 3278 CARRIAGE-CONTROL CHARACTERS

The IBM 3278 terminal routines recognize the following logical carriage-control characters. This recognition is under the control of the @CC I/O modifier.

blank	single space (skip 0 lines)
0	double space (skip 1 line)
-	triple space (skip 2 lines)
+	single space
&	no space after printing
9	single space
1	triple space (see also the %TOP device command)
2	triple space
4	triple space
6	triple space
8	triple space

All other logical carriage-control characters are treated as a request to single-space and print the entire line (including the carriage-control character).

Lines written using the @MCC I/O modifier (machine carriage control) are always single-spaced and the machine carriage-control character is ignored.

APPENDIX F: THE LEE DATA TERMINAL

This appendix summarizes the differences between the Lee Data terminal and the IBM 3278 Display Station. The majority of these differences are in the keyboard layout.

- (1) The ON/OFF switch is located at the right rear of the console base for the Lee Data terminal.
- (2) On the Lee Data terminal, the following functions are generated without the use of the ALT function key: PA1, PA2, SYS REQ, and CLEAR.
- (3) An additional caps (CAPS LOCK) function is provided on the Lee Data terminal that generates uppercase alphabetic characters while not affecting the upper/lowercase function of the nonalphabetic keys.
- (4) The following characters are located on different keys or key positions: tilde (~), reverse slant (\), and broken vertical line (|). Two additional characters are provided on the Lee Data terminal: the left and right brackets ([]).
- (5) The ALT-{ key operates as a multiple-delete key to delete several characters at a time.
- (6) The operator information area uses words to display information about the terminal. During normal operation, the words "NO PRINTER" are displayed. The next field to the right often contains a phrase that indicates the current state of the terminal:

INHIBIT-WAITING indicates that MTS is processing the input line.
INHIBIT-OVERRUN indicates that data was entered too fast.
INHIBIT-FIELD FULL indicates that an input line was too long.
INHIBIT-PROTECTED indicates that the user attempted to enter data in a protected field.

Other phrases indicate either some esoteric condition or a system malfunction. To the left of the "NO PRINTER" field, the following symbols are often displayed:

CPS means that caps lock is in effect.
SFT means that shift lock is in effect.
INS means that insert mode is in effect.