

HP-UX Concepts and Tutorials
Vol. 4: Data Communications



HP-UX
HP-UX
HP-UX
HP-UX
HP-UX
HP-UX
HP-UX

HP-UX Concepts and Tutorials

Vol. 4: Data Communications

for the HP 9000 Series 200/500 Computers

Manual Part No. 97089-90004

© Copyright 1984, Hewlett-Packard Company.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

Use of this manual and flexible disc(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

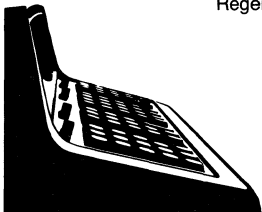
Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

© Copyright 1980, Bell Telephone Laboratories, Inc.

© Copyright 1979, 1980, The Regents of the University of California.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.



Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

July 1984...First Edition

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard Fort Collins Systems Division products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from the date of delivery.* Hewlett-Packard will, at its option, repair or replace equipment which proves to be defective during the warranty period. This warranty includes labor, parts, and surface travel costs, if any. Equipment returned to Hewlett-Packard for repair must be shipped freight prepaid. Repairs necessitated by misuse of the equipment, or by hardware, software, or interfacing not provided by Hewlett-Packard are not covered by this warranty.

HP warrants that its software and firmware designated by HP for use with a CPU will execute its programming instructions when properly installed on that CPU. HP does not warrant that the operation of the CPU, software, or firmware will be uninterrupted or error free.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

* For other countries, contact your local Sales and Support Office to determine warranty terms.

Contents

The articles contained in *HP-UX Concepts and Tutorials* are provided to help you use the commands and utilities provided with HP-UX. The articles have several sources. Some were written at Hewlett-Packard specifically for the HP 9000 family of computers. Others were written at Bell Laboratories and may discuss options or command behavior that does not apply to your system.

HP-UX Concepts and Tutorials has four volumes:

- Volume 1: Text Processing and Formatting
- Volume 2: Program Development and Maintenance
- Volume 3: Shells and Miscellaneous Tools
- Volume 4: Data Communications

This is “Vol. 4: Data Communications” and the articles it includes are:

1. Using the System Console with HP 9000 Series 200 Computers
2. HP-UX and the HP 9000 Model 520 as System Console
3. Mailx: Mail Handler

This volume also contains a tab divider labeled “Asynchronous Communications”. If you have ordered the manual *HP-UX Asynchronous Communications Guide* (part number 97076-90001), then you can insert it behind this tab. The manual **is not** supplied with *HP-UX Concepts and Tutorials* (part number 97089-90004).

Table of Contents

Using the System Console with HP9000 Series 200 Computers

Using the Internal Terminal Emulator	1
Character Entry Group	3
Numeric Pad Group	3
Display Control Group	4
Setting and Clearing Tab Stops	4
Cursor Control	4
Edit Group	8
Function Key Group	9
Defining User Keys Locally	11
Defining User Keys Programmatically	13
Controlling the Function Key Labels Programmatically	13
System Control Group	14
The Display	16
Memory Addressing Scheme	16
Row Addressing	16
Column Addressing	17
Cursor Sensing	17
Absolute Sensing	17
Relative Sensing	17
Cursor Positioning	17
Screen Relative Addressing	18
Absolute Addressing	18
Cursor Relative Addressing	19
Combining Absolute and Relative Addressing	20
Display Enhancements	21
Raster Control	21
Accessing Color (Series 200 Model 236 with Color Video only)	22
Selecting a Pen (Color Pair)	22
Changing Pen Definitions	22
Configuring the ITE	26
Configuration Function Keys	26
Terminal Configuration Menu	26
Description of Fields	27
Changing the Fields	32
ITE Escape Sequences	33
Sequence Types	33
Keyboard Diagrams for Other Languages	37

Using the System Console with HP 9000 Series 200 Computers

Using the Internal Terminal Emulator

The Internal Terminal Emulator consists of “device driver” code contained in the HP-UX kernel and associated with the built-in keyboard and display on the Series 200 Models 226 and 236 with memory management. It also drives the external keyboard and CRT for Series 200 Model 220 computers when the keyboard/HP-IB and composite video interfaces are used for the System Console.

For the remainder of this article, the Series 200 Models 220, 226 and 236 when mentioned have memory management. Memory management means your computer contains a central processing unit designed to run the HP-UX system. To determine if you have memory management, look on the back of your computer for one of the following product numbers:

- 9920U (Model 220)
- 9826U (Model 226)
- 9836U (Model 236)
- 9836CU (Model 236 with color video)

The **system console** is a keyboard and display (or terminal) given a unique status by HP-UX and associated with the special (device) file `/dev/console`. All boot ROM error messages, HP-UX system error messages, and certain system status messages are sent to the system console. Under certain conditions (for example, the single-user state), the system console provides the only mechanism for communicating with HP-UX.

The HP-UX operating system assigns the system console function according to a prioritized search sequence when the HP-UX kernel gains control during the boot-up process. When a given interface board or the Internal Terminal Emulator (ITE) is assigned the system console function, the terminal associated with that interface board (or the keyboard and display associated with the ITE) becomes the physical system console. HP-UX's search for a system console terminates as soon as one of the following conditions is met:

1. An HP 98626A Serial Interface board or HP 98628A Datacomm Interface board that has its “remote bit” set¹ is installed in the computer. If this condition is met and an ITE is present, the ITE is assigned the special (device) file `/dev/tty00` and is considered to be the first non-system console terminal connected to HP-UX. (If no ITE is present, `/dev/tty00` is available for other assignment.)
2. The appropriate hardware (associated with an ITE) is present. This is the general case with the Series 200 Models 226 and 236.

In the case of multiple occurrences, the HP 98626A Serial Interface board or HP 98628A Datacomm Interface board with the lowest select code is chosen to be the system console.

If none of the above conditions are met, no system console exists. While HP-UX tolerates this, you cannot functionally use HP-UX without a system console.

¹ On the HP 98626A Serial Interface board the remote bit is set by cutting a jumper as described in the installation manual supplied with your computer. On the HP 98628A Datacomm Interface board the remote bit is set by setting a switch on the board as described in that board's installation manual.

Note

To install the HP-UX system it is necessary to communicate with the boot ROM. Because the boot ROM will not use a terminal connected to an HP 98628A Datacomm Interface Board as its display, HP-UX must be installed using either the computer's ITE hardware or an HP 98626A Serial Interface board.

For a complete list of Hewlett-Packard terminals supported by HP-UX, see the "Supported Peripherals" appendix located in your *HP-UX System Administrator Manual*.

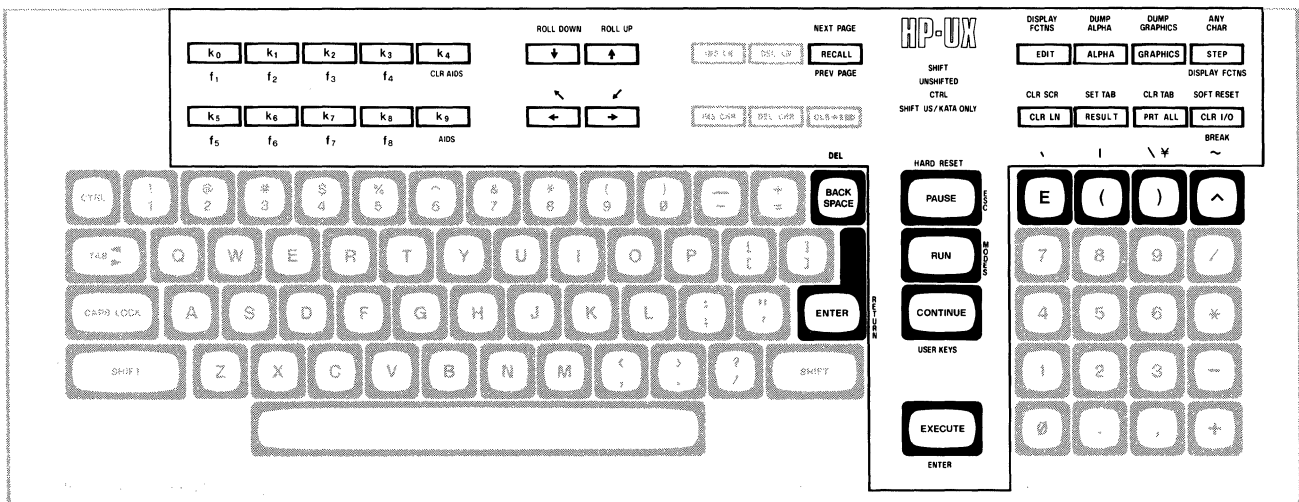
The keyboard overlay supplied with your Series 200 computer allows you to convert it for use as an HP-UX system console keyboard. The key labels on the overlay are color-coded as follows:

- Shifted keys are light blue with the exception of four US/KATA keys. The US/KATA keys and their functions are covered in the "Function Key Group" section. On the overlay, the keys are labeled from left to right: ` , | \ , and ~;
- Unshifted keys are dark brown;
- Control keys are orange;
- Shifted US/KATA keys are dark blue.

The Series 200 computers keyboard is divided into six functional groups:

- Character Entry Group,
- Numeric Pad Group,
- Display Control Group,
- Editing Group,
- Function Key Group,
- Systems Control Group.

These key groups are discussed next.



US ASCII Keyboard and Overlay

Character Entry Group

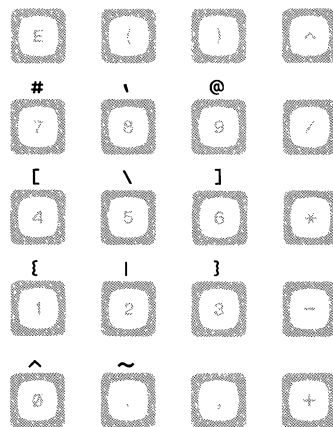
The character entry keys are arranged like a typewriter, but have some added features.

SHIFT	gives you uppercase letters when you are typing in lowercase (caps lock off). When you are typing in upper case (caps lock on) , this key will give you lower case letters.
CAPS LOCK	sets the unshifted keyboard to either upper case (the power-on default) or lower case (normal typewriter operation) letters.
ENTER RETURN	sends the ReturnDef sequence to the computer (the default is to send C_R). When a program is running, this key is used to input information requested by the computer.
TAB	sends a tab character (CTRL-I) to the computer.
CTRL	provides access to the standard ASCII control characters.
BACK SPACE	sends the back space character (CTRL-H) to the computer in the remote mode, and in the local mode it moves the cursor one space to the left.

Numeric Pad Group

The numeric group of keys is located to the right of the character keys. The layout of the numeric key pad is similar to that of a standard office calculator. These keys are convenient for high-speed entry of numeric data.

The numeric key pad on Series 200 computer also provides the non-US ASCII keyboard user with a few of the character keys not found on their keyboard. These characters are accessed by holding the shift key down and pressing the appropriate numeric key, as shown in the diagram below.



Additional Numeric Key Pad Characters

Display Control Group

The display control group consists of the keys that control the location of the cursor on the display. Each display control key and its function is described in the sections that follow.

Setting and Clearing Tab Stops

You can define and delete a series of tab stops by using the following tab functions.

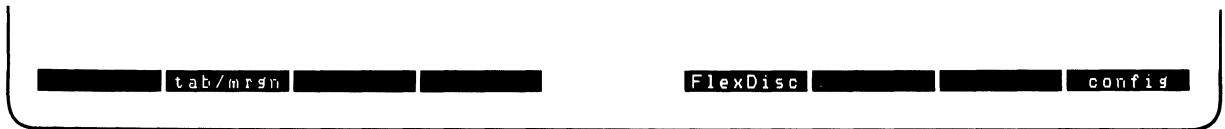
SET TAB sets tab stops. To set a tab stop move the cursor to the desired location, hold the **SHIFT** key down, and press **SET TAB**.

CLR TAB deletes tab stops. To delete tab stops, move the cursor to the tab position which you want to remove, hold the **SHIFT** key down, and press **CLR TAB**.

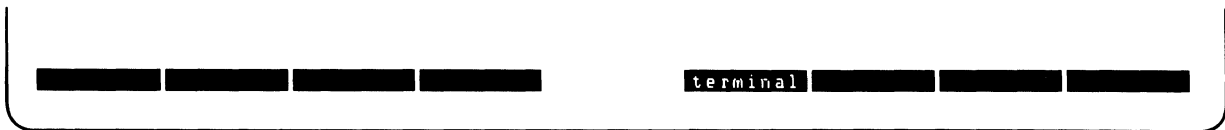
Cursor Control

To use the knob (cursor control wheel) and arrow keys, either have the **transmit functions mode** disabled or be in the **vi** or **ex** editor. The **transmit functions mode** specifies whether escape code functions are executed locally at the terminal (ITE) or transmitted to the host computer. When the system is shipped the default for this mode is: NO. In the **vi** or **ex** editor, the arrow keys and knob can be used as described in this section. To disable the **transmit functions mode**:

press: **k9** (labeled AIDS on the overlay)



press: **k8** (config)



press: **k5** (terminal)

When a screen display similar to the one shown below appears, press the **TAB** key until the cursor is positioned just after the **XmitFunctn(A)**.

```

                                TERMINAL CONFIGURATION
Language      USASCII
ReturnDef     NO
LocalEcho     OFF      CapsLock  OFF      Ascii 8 Bit NO
XmitFunctn(A) NO      InHcolWrP(C) NO

SAVE CFG  NEXT  PREVIOUS  DEFAULT  _____  _____  DSPY FN  config

```

If the word following the label is NO, then the transmit functions mode is not active, so

press: **k8** (config)

to return to the HP-UX environment. If the word following the **XmitFunctn(A)** label is a YES, then

press: **k2** (NEXT)

This turns off the transmit function mode.

press: **k1** (SAVE CFG)





This returns you to the ITE environment.

When the transmit functions mode is off, there are four keys used to change the location of the cursor in the shell environment:

- ↑** moves the cursor up in the output area of the display screen.
- ↓** moves the cursor down in the output area of the display screen.
- ←** moves the cursor to the left in the output area of the display screen.
- moves the cursor to the right in the output area of the display screen.

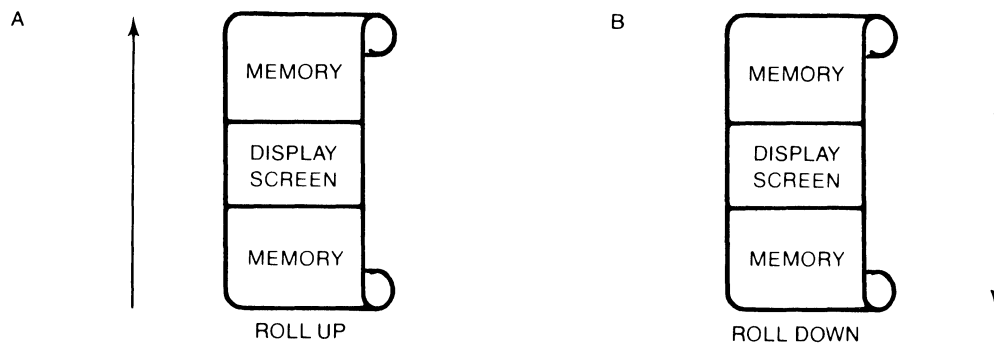
6 Series 200 Console

These same keys when used with the **SHIFT** key enable you to scroll up and down through the screen display, and to move the cursor to its upper or lower home position. These keys are listed as follows:

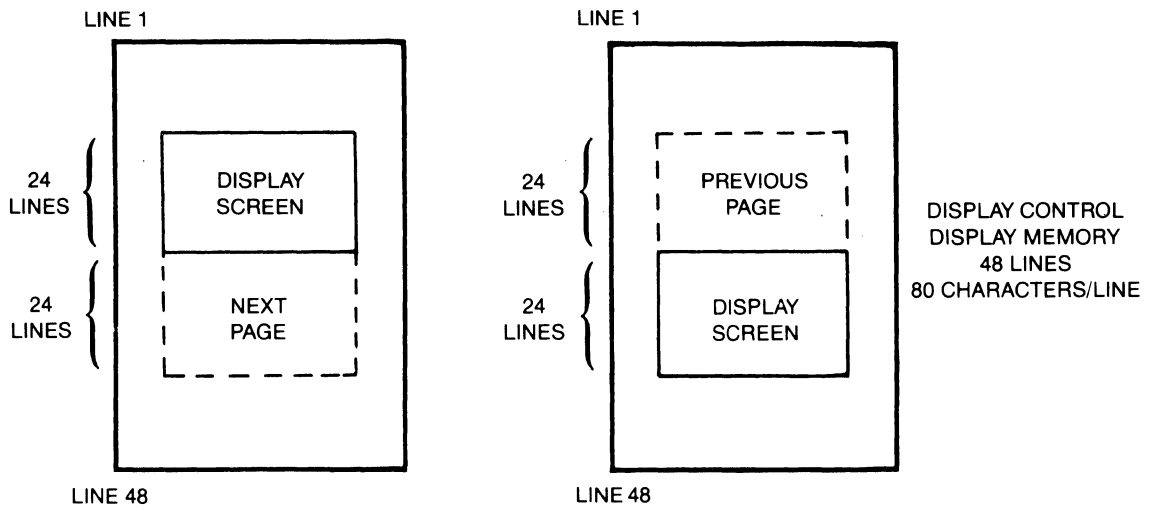
- ROLL DOWN
shift-  scrolls the screen display downward.
- ROLL UP
shift-  scrolls the screen display upward.
- shift-  moves the cursor to its upper home position.
- shift-  moves the cursor to its lower home position.

Another method used to move the cursor is the **knob**. Rotating the knob either clockwise or counter-clockwise moves the cursor horizontally. Rotating the knob while pushing the **SHIFT** key moves the cursor vertically until the top or bottom of the display screen is reached, at that time the screen display rolls up or down in display to the next page or previous page.

What is the next page or previous page? Data in display memory can be accessed (displayed on the screen) in blocks that are known as “pages”. A page consists of 24 (23 for a Model 226) lines of data. The current page is that sequence of lines which appears on the screen at any given time. The **previous page** is the preceding 24 lines in display memory. The **next page** is the succeeding 24 lines in display memory. This concept, along with the concept of rolling data through the display screen and memory, are shown in the following illustrations.



The “Roll Up and Roll Down” Functions



Previous Page and Next Page Concepts

Edit Group

The edit group consists of the keys that allow you to modify the data presented on the screen. However, the edited data cannot be read back by the system. Typically these features are used to modify text within a program or file, to view data which has scrolled out of the screen window and to clear the screen.

To use these features in the vi editor, you should have an `.exrc` file in your **\$HOME** directory which maps these keys to their function using escape code sequences. Reference to this file is made in the *HP-UX System Administrator Manual* and in "The Vi Editor" selected article.

The **RECALL** key has been redefined as follows:

RECALL moves you back a page within your text.
PREV PAGE

RECALL moves you forward a page within your text.
NEXT PAGE
shift - **RECALL**

All line edit keys and character edit keys function as stated below:

CLR END clears the characters in a line from the present cursor position to the end of the line.

INS LN causes the line containing the cursor and all text lines below it to move down one line, and a blank line to be inserted in the row containing the cursor. The cursor will move to the left margin of the blank line.

DEL LN deletes the line containing the cursor from display memory. All text lines below this line will roll up one row, and the cursor will move to the left margin.

CLR LN performs the same function as CLR END.

INS CHR sets the insert mode, allowing you to insert characters to the left of the cursor.

DEL CHR deletes the character at the cursor.

DEL sends the DEL character to the computer.
shift - **BACK SPACE**

Function Key Group

In the upper left-hand corner of your keyboard next to the knob are a set of ten function keys (HP-UX recognizes only eight function keys: **f1** through **f8**). The remaining keys are used as "AIDS" keys which will be discussed later on in this section. The functions performed by these keys change dynamically as you use the computer. At any given time the applicable function labels for these keys appear across the bottom of the display screen.

The function key group also includes four US/KATA keys which are accessed using the numeric key pad. These keys are very useful when using HP-UX and they have been given the following names:

shift- E	accent grave.
shift- (vertical bar.
shift-)	backslash.
shift- ^	tilde.

The remainder of this section covers the relationship of the eight function keys, **k1** through **k8**, to the overlay functions: MODES, AIDS and USER KEYS.

RUN	M O D E S	allows access to one of the modes described in the following sections. An example of the function key display is given:
------------	--	---



You may use these function keys to enable and disable various terminal operating modes. Each defined mode selection key alternately enables, and disables a particular mode. When the mode is enabled, an asterisk (*) appears in the associated key label on the screen, as seen in the REMOTE key label above.

When the **remote mode** is enabled and a key is pressed, the terminal transmits the associated ASCII code to HP-UX. In **local mode** (remote mode is disabled), when a character key is pressed, the associated character is displayed at the current cursor position on the screen (nothing is transmitted to HP-UX).

When the **auto line feed mode** is enabled, an ASCII line feed control code is automatically appended to each ASCII carriage-return control code generated through the keyboard. ASCII carriage-return control codes can be generated through the keyboard in any of the following ways:

- By pressing **RETURN**.
- By holding down CTRL and pressing **M**.
- By pressing any of the user keys **f1** through **f8**, provided that a carriage-return code is included in the particular key definition.

When the **display functions mode** is enabled, the terminal (ITE) displays ASCII control codes, and escape sequences but does not execute them.

k9
AIDS

provides another menu in the display, showing three general control keys:



tab/mrgn pressing this softkey provides another menu with: SET TAB, CLR TAB, and CLR TABS. The first two softkey functions were previously defined in the display control group section. The last softkey [CLR TABS] when pressed clears all tab stops currently set.

FlexDisc pressing this softkey provides another menu with these labels: fd.1 and fd.0. Whenever this symbol * appears in the label block the internal disc drive 1 (left drive) or 0 (right drive) is in use.

config pressing this softkey provides another menu with only the softkey [terminal] in it. If you press this softkey, you get the **TERMINAL CONFIGURATION** display on your screen as shown in the display control group section of this article.

CLR AIDS clears the screen display of the function key labels. The user function keys, however, are still enabled.

k4

EXECUTE

ENTER

not implemented.

CONTINUE

USER KEYS

displays eight function keys which can be defined either locally by the user or remotely by a program executed in a host computer. By “defined” it is meant:

- You can assign to each key a string of ASCII alphanumeric characters and/or control codes (such as carriage return or line feed).
- You can specify each key’s operational attribute: whether its key definition is to be executed locally at the terminal, transmitted to the computer, or both.
- You can assign to each key an alphanumeric label (up to 16 characters) which, in **user keys mode**, is displayed across the bottom of the screen.

Defining User Keys Locally

When defining a key from the keyboard, the key content may include explicit escape sequences (entered using display functions mode) that control or modify the ITE's operation.

The definition of each user key may contain up to 80 characters (alphanumeric characters, ASCII control characters, and explicit escape sequence characters).

To define **USER KEYS** locally (from the keyboard), press the **SHIFT** - **CONTINUE** (USER KEYS) keys simultaneously. The following user keys menu will appear:

KEY DEFINITION FIELD	ATTRIBUTE FIELD	LABEL FIELDS
f1	LABEL	f1
ε _{cp}		
f2	LABEL	f2
ε _{cq}		
f3	LABEL	f3
ε _{cr}		
f4	LABEL	f4
ε _{cs}		
f5	LABEL	f5
ε _{ct}		
f6	LABEL	f6
ε _{cu}		
f7	LABEL	f7
ε _{cv}		
f8	LABEL	f8
ε _{cw}		

This menu contains the default values for all of the fields. If your screen does not contain the default values as shown and you want them set and displayed press **f4** (DEFAULT).

The menu contains a set of fields that you access using the **TAB** key.

For each user key the menu contains three unprotected fields:

ATTRIBUTE FIELD — This one character field always contains an uppercase L, T, or N signifying whether the content of the particular user key is to be:

- L Executed locally only.
- T Transmitted to the host computer only.
- N Treated in the same manner as the alphanumeric keys. If the ITE is in local mode, the content of the key is executed locally. If the ITE is in the remote mode and LocalEcho is disabled (OFF), the content of the key is transmitted to the host computer. If the ITE is in remote mode and local echo is enabled (ON), the content of the key is both transmitted to the host computer and executed locally.

The alphanumeric keys are disabled when the cursor is positioned in this field. You change the content of this field by pressing (NEXT) or (PREVIOUS).

LABEL FIELD — This field eight character field to the right of the word "LABEL" allows you to supply the user key's label. When the ITE is in user keys mode, the key labels are displayed from left to right in ascending order across the bottom of the screen.

KEY DEFINITION FIELD — The entire line (80 characters) immediately below the attribute and label field is available for specifying the character string that is to be displayed, executed, and/or transmitted whenever the particular key is physically pressed.

When entering characters into the key definition field you may use the display functions mode. Note that this implementation of display functions mode is separate from that which is enabled/disabled via the mode selection keys. When entering the label and key definition you may access display functions mode by way of function key (DSPY FN) for the Model 236 and for the Model 226 use function key .

The (RETURN) key can be used to include carriage return (C_R) codes (with display functions mode enabled) in key definitions. If auto line feed mode is also enabled, the (RETURN) key will generate a $C_R L_F$, otherwise it is considered a cursor movement key.

When the user keys menu is displayed on the screen you may use the , and keys for editing the contents of the label and key definition fields.

When you are finished defining all the desired keys, press the (AIDS), (MODES) or (USER KEYS) key (in all three cases the user keys menu disappears from the screen). When you press (USER KEYS), the defined user key labels are displayed across the bottom of the screen and the through user keys, as defined by you, are enabled.

Defining User Keys Programmatically

From a program executing in a host computer, you can define one or more keys using the following escape sequence format:

```
^c&.f <attribute><label length><string length><label><string>
```

where:

```
<attribute> =
0 a : normal (0 is the default)
1 a : local only
2 a : transmit only

<key> = 1-BK : f1-f8, (1 is the default)
        respectively

<label length> = 0-16d (0 is the default)
<string length> = 0-80L (1 is the default)
                (-1 causes field to be erased)
```

The <attribute>, <key>, <label length>, and <string length> parameters may appear in any sequence but must precede the label and key definition strings. You must use an uppercase identifier (A, K, D, or L) for the final parameter and a lowercase identifier (a, k, d, or l) for all preceding parameters. Following the parameters, the first 0 through 16 characters, as designated by <label length>, constitute the key's label; however, only the first 8 characters are recognized. The next 0 through 80 characters, as designated by <string length>, constitute the key's definition string. The total number of displayable characters (alphanumeric data, ASCII control codes such as ^CR and ^LF, and explicit escape sequence characters) in the label string must not exceed 16, and in the definition string must not exceed 80.

Example: Assign login as the label and TOM as the definition for the user key. The key is to have attribute "N".

```
^c&.f5k5d4LloginTOM^c^jB
```

After issuing the foregoing escape sequence from your program to the terminal, the portion of the user keys menu is as follows:

```
f5 N LABEL login
TOM^c^R
```

If the transmit only attribute (2) is designated, the particular user key will have no effect unless the terminal is in remote. The ^Ec&^jB sequence turns on the user labels.

Controlling the Function Key Labels Programmatically

From a program executing in a host computer, you can control the function key labels display as follows by using escape sequences:

- You can remove the key labels from the screen entirely (this is the equivalent of pressing (CLR AIDS)).
- You can enable the mode selection keys (this is the equivalent of pressing the (MODES) key).
- You can enable the user keys (this is the equivalent of pressing the (USER KEYS) key).

The escape sequences are as follows:

- ESC & J @ Enables the user keys and remove all key labels from the screen.
- ESC & J A Enables the modes key.
- ESC & J B Enables the user keys.

System Control Group

These keys are located in the upper-right corner of your keyboard. They control system functions related to the display, printer and editing operations.

- PAUSE** ESC generates special ASCII control character number 27 (escape character).
- EDIT** not implemented.
- DISPLAY FCTNS
shift - **EDIT** enables **display functions mode** as defined in the function key group. Pressing the **DISPLAY FCTNS** key again cancels the **display functions mode**.
- ALPHA**
and
GRAPHICS These commands work together to control the Alpha and Graphic displays. The following table shows how these commands work.

If	And you	Result
Alpha ON Graphics OFF	Press Alpha	No change in display
Alpha ON Graphics OFF	Press Graphics	Both displays on screen
Alpha ON Graphics ON	Press Alpha	Graphic display turned off
Alpha OFF Graphics ON	Press Graphics	No change in display
Alpha OFF Graphics ON	Press Alpha	Both displays on screen
Alpha ON Graphics ON	Press Graphics	Alpha display turned off

- DUMP GRAPHICS
shift-**GRAPHICS** not implemented.
- DUMP ALPHA
shift-**ALPHA** not implemented.
- ANY CHAR
shift-**STEP** causes the next three characters typed (must be integers) to be interpreted as the decimal specifier of an HP extended ASCII character.
- STEP** not implemented.
- CLR LN** was defined in the Edit Group.
- CLR SCR
shift-**CLR LN** clears the entire alpha portion of the screen display.

RESULT	not implemented.
PRT ALL	not implemented.
SET TAB shift- RESULT	sets a tab at the current cursor position. Tabs are in effect in the alpha display until cleared by CLR TAB .
CLR TAB shift- PRT ALL	clears a tab previously set at the current cursor position.
CLR I/O	not implemented.
SOFT RESET shift- CLR I/O	<p>does the following:</p> <ul style="list-style-type: none"> • Sounds the computer's beeper. • Disables display functions mode (if enabled). • Halts any datacomm transfers currently in progress, clears the datacomm buffers, and reinitializes the datacom port according to the appropriate power-on datacomm configuration parameters. <p>The data on the screen, all terminal operating modes (except display functions mode), and all active configuration parameters are unchanged.</p>
HARD RESET shift- PAUSE	<p>has the same effect as turning the computer's power off and then back on. A hard reset does the following:</p> <ul style="list-style-type: none"> • Sounds the computer's beeper. • Clears all of alphanumeric memory. • Resets the terminal configuration menu parameters to their power on values. • Resets certain operating modes and parameters as follows: <ul style="list-style-type: none"> - Disables display functions mode, and caps mode. - Turns off the insert character edit function. - Resets the user keys to default values. - Turns on the alphanumeric display. - Resets color pairs to their default values.
BREAK	creates an interrupt signal (SIGINT) which is sent to all processes within your terminal (ITE). For information on this signal read the sections SIGNAL(2) and TTY(4) in your <i>HP-UX Reference</i> . To learn how to use this signal in a shell script read "Shell Programming" in your <i>HP-UX Selected Articles</i> .

The Display

The Internal Terminal Emulator's (ITE's) display has many features of its own, video highlights (such as inverse video and blinking), raster control, cursor sensing and addressing, and color highlight control (for Model 236 computers equipped with a color display). These functions are accessed only through escape sequences and are discussed in the sections that follow. As you read this section, note the last letter in an extended escape sequence is always capitalized. An extended escape sequence consist of the escape-code character followed by at least two subsequent characters.

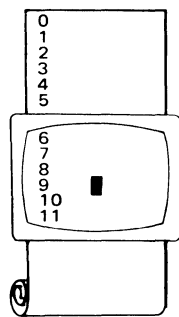
Memory Addressing Scheme

Display memory positions can be addressed using absolute or relative coordinate values. On the Model 220 and 236, display memory is made up of 80 columns (0 thru 79) and two 24 line pages (0 thru 47) with 80 characters per line. A Model 26 upgraded for HP-UX use has a display memory made up of 50 columns (0 thru 49) and two 23 line pages (0 thru 45) with 50 characters per line. The types of addressing available are absolute (memory relative), screen relative, and cursor relative.

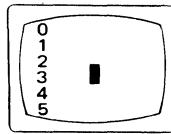
Row Addressing

The figure below illustates the way that the three types of addressing affect row or line numbers. The cursor is shown positioned in the fourth row on the screen. Screen row 0 is currently at row 6 of display memory. In order to reposition the cursor to the first line of the screen the following three destination rows could be used:

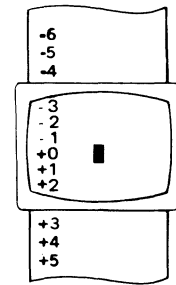
- Absolute: row 6
- Screen Relative: row 0
- Cursor Relative: row -3



a.) Absolute: row 6



b.) Screen Relative: row 0



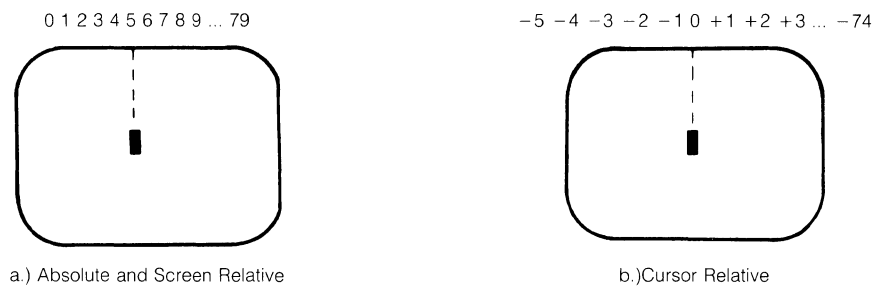
c.) Cursor Relative: row -3

Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen relative and absolute addressing. The figure below illustrates the difference between absolute and cursor relative addressing. The cursor is shown in column 5.

Whenever the row or column addresses exceed those available, the largest possible value is substituted. In screen relative addressing, the cursor cannot be moved to a row position that is not currently displayed. For example, in the cursor relative portion of the figure on the previous page (showing row addressing), a relative row address of -10 would cause the cursor to be positioned at the top of the current screen (relative to row -3). Column positions are limited to the available screen positions. For example, in the following illustration, the absolute column addressing example shows limits of 0 and 79, while the relative column addressing example shows limits of -5 and $+74$. The cursor cannot be wrapped around from column 0 to column 79 by specifying large negative values for relative column positions.



Column Addressing

Cursor Sensing

The current position of the screen cursor can be sensed. The position returned can be the absolute position in the display memory or the location relative to the current screen position.

Cursor sensing functions only when the "terminal" is in remote mode.

Absolute Sensing

When a program sends the escape sequence $E_{c a}$ to the terminal, the terminal returns to the program an escape sequence of the form $E_{c \& a x x x c y y y R^c_R}$, where xxx is the absolute column number and yyy is the absolute row number of the current cursor position. You will later see that this escape sequence is identical to the escape sequence for an absolute move of the cursor.

Relative Sensing

When a program sends the escape sequence $E_{c \backslash}$, the terminal returns to the program an escape sequence of the form $E_{c \& a x x x c y y y Y^c_R}$ where xxx is the column number of the cursor and yyy is row position of the cursor relative to screen row 0. This escape sequence is identical to the escape sequence for a relative move of the cursor (discussed later in this article).

Cursor Positioning

The cursor can be positioned directly by giving memory or screen coordinates, or by sending the escape codes for any of the keyboard cursor positioning operations.

Screen Relative Addressing

To move the cursor to any character position on the screen, use any of the following escape sequences:

```

E_c&.a<column number> c <row number>Y
E_c&.a<row number> y <column number>C
E_c&.a<column number>C
E_c&.a<row number>Y

```

where: <column number> is a decimal number specifying the screen column to which you wish to move the cursor. Zero specifies the leftmost column.

<row number> is a decimal number specifying the screen row (0 thru 23) to which you wish to move the cursor. Zero specifies the top row of the screen; 23 specifies the bottom row.

When using the escape sequences for screen relative addressing, the data on the screen is not affected (the cursor may only be moved around in the 24 rows and 80 columns currently displayed, thus data is not scrolled up or down).

If you specify only <column number>, the cursor remains in the current row. Similarly, if you specify only <row number>, the cursor remains in the current column.

Example

The following escape sequence moves the cursor to the 20th column of the 7th row on the screen:

```
E_c&.a6y19C
```

Absolute Addressing

You can specify the location of any character within display memory by supplying absolute row and column coordinates. To move the cursor to another character position using absolute addressing, use any of the following escape sequences:

```

E_c&.a<column number> c <row number>R
E_c&.a<row number> r <column number>C
E_c&.a<column number>C
E_c&.a<row number>R

```

where: <column number> is a decimal number (0 thru 79) specifying the column coordinate (within display memory) of the character at which you want the cursor positioned. Zero specifies the first (leftmost) column in display memory, 79 the rightmost column.

<row number> is a decimal number (0 thru max) specifying the row coordinate (within display memory) of the character at which you want the cursor positioned. Zero specifies the first (top) row in display memory, max specifies the last. The value of max is specified as:

$$[24 (\text{lines/page}) \times \text{num_page} (\text{pages})] - 1$$

where num_page is the number of pages of display memory specified by the system configuration. As shipped to you, the configuration dictates that 2 pages of display memory be allocated. Thus, the last row that can be addressed is 47.

When using the above escape sequences, the data visible on the screen rolls up or down (if necessary) in order to position the cursor at the specified data character. The cursor and data movement occur as follows:

- If a specified character position lies within the boundaries of the screen, the cursor moves to that position; the data on the screen does not move.
- If the absolute row coordinate is less than that of the top line currently visible on the screen, the cursor moves to the specified column in the top row of the screen; the data then rolls down until the specified row appears in the top line of the screen.
- If the absolute row coordinate exceeds that of the bottom line currently visible on the screen, the cursor moves to the specified column in the bottom row of the screen; the data then rolls up until the specified row appears in the bottom line of the screen.

If you specify only a <column number>, the cursor remains in the current row. Similarly, if you specify only a <row number>, the cursor remains in the current column.

Example

To position the cursor (rolling the data if necessary) at the character residing in the 60th column of the 27th row in display memory, the escape sequence is:

```
ESC & a 26 r 59 C
```

Cursor Relative Addressing

You can specify the location of any character within display memory by supplying row and column coordinates that are relative to the current cursor position. To move the cursor to another character position using cursor relative addressing, use any of the following escape sequences:

```
ESC & a ±<column number> c ±<row number> R
ESC & a ±<row number> r ±<column number> C
ESC & a ±<column number> C
ESC & a ±<row number> R
```

where: <column number> is a decimal number specifying the relative column to which you wish to move the cursor. A positive number specifies how many columns to the right you wish to move the cursor; a negative number specifies how many columns to the left.

<row number> is a decimal number specifying the relative row to which you wish to move the cursor. A positive number specifies how many rows down you wish to move the cursor; a negative number specifies how many rows up you wish to move the cursor.

When using the above escape sequences, the data visible on the screen rolls up or down (if necessary) in order to position the cursor at the specified data character. The cursor and data movement occur as follows:

- If a specified character position lies within the boundaries of the screen, the cursor moves to that position; the data on the screen does not move.
- If the specified cursor relative row precedes the top line currently visible on the screen, the cursor moves to the specified column in the top row of the screen; the data then rolls down until the specified row appears in the top line of the screen.
- If the specified cursor relative row exceeds the bottom line currently visible on the screen, the cursor moves to the specified column in the bottom row of the screen; the data then rolls up until the specified row appears in the bottom line of the screen.

If you specify only a <column number>, the cursor remains in the current row. Similarly, if you specify only a <row number>, the cursor remains in the current column.

Example

To position the cursor (rolling the data if necessary) at the character residing 15 columns to the right and 25 rows above the current cursor position (within display memory), use the escape sequence:

```
^c&a+15c-25R
```

Combining Absolute and Relative Addressing

You may use a combination of screen relative, absolute and cursor relative addressing within a single escape sequence.

For example, to move the cursor (and roll the text if necessary) so that it is positioned at the character residing in the 70th column of the 18th row below the current cursor position, use the escape sequence:

```
^c&a69c+18R
```

Next, to move the cursor so that it is positioned at the character residing 15 columns to the left of the current cursor position in the 4th row currently visible on the screen, use the escape sequence:

```
^c&a-15c 3Y
```

Similarly, to move the cursor (and roll the text up or down if necessary) so that it is positioned at the character residing in the 10th column of absolute row 48 in display memory, use the escape sequence:

```
^c&a9c 47R
```

Display Enhancements

The terminal includes as a standard feature the following display enhancement capabilities:

- Inverse Video - black characters are displayed against a white background.
- Underline Video - characters are underscored.
- Blink Video - characters blink on and off.

Note

The Model 226 computer doesn't provide display enhancements. The Model 220 computer provides display enhancements if it has an HP 98204A composite video card. The Model 236 computer with color video doesn't have the half bright enhancement.

The display enhancements are used on a field basis. The field can not span more than one line. The field scrolls with display memory. Overwriting a displayable character in a field preserves the display enhancement. The enhancements may be used separately or in any combination. When used, they cause control bits to be set within display memory.

From a program or from the keyboard, you enable and disable the various video enhancements by embedding escape sequences within the data. The general form of the escape sequence is:

$\text{E}_c \& \text{d} \langle \text{enhancement code} \rangle$

where enhancement code is one of the uppercase letters A through O specifying the desired enhancement(s) or an @ to specify end of enhancement:

Enhancement Character

	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Half-Bright									x	x	x	x	x	x	x	x
Underline					x	x	x	x					x	x	x	x
Inverse Video			x	x			x	x			x	x			x	x
Blinking		x		x		x		x		x		x		x		x
End Enhancement	x															

Note that the escape sequence for "end enhancement" ($\text{E}_c \& \text{d} @$) or the escape sequence for another video enhancement, ends the previous enhancement.

Raster Control

The terminal provides the ability to enable and disable the alphanumeric display. The escape sequences for these capabilities are:

$\text{E}_c * \text{d} \text{E}$ Turn on the alphanumeric display; enable writing to the alphanumeric display.

$\text{E}_c * \text{d} \text{F}$ Turn off the alphanumeric display; disable writing to the alphanumeric display.

Accessing Color (Series 200 Model 236 with Color Video only)

To access color on the Series 200 Model 236, you must understand some simple terms.

Color pair - two colors which define the foreground color (color of the characters) and the background color, respectively. At least one of the color pair must be black; displaying color on color is not possible. A total of 64 color pairs are possible, but only eight can be displayed at any one time.

Pen # - one of eight predefined color pairs. Pen 0 through pen 7 are initially defined as follows (re-defining a color pair is discribed later):

Pen #	foreground color	background color
0	white	black
1	red	black
2	green	black
3	yellow	black
4	blue	black
5	magenta	black
6	cyan	black
7	black	yellow

Pen #0 is the default pen selected by the terminal when writing to the display.

Pen #7 is always used for displaying the softkey labels.

Selecting a Pen (Color Pair)

By using an escape sequence, you can select a pen number other than pen #0 when writing to the display. Like other display enhancements, pen selection is used on a field basis. The field cannot span more than one line. That is, the pen selection is only active until a new-line character is encountered; then the default pen is re-selected. The escape sequence for selecting a pen is:

```
ESC & v n <parameter>
```

where n is the pen number you wish to use, and <parameter> is a single character that specifies the action as described below. To select a pre-defined pen number, the necessary <parameter> is s. Thus,

```
ESC & v 4s
```

selects the pre-defined pen number 4.

Changing Pen Definitions

You may change the pre-defined color pair for any of the eight existing display pens. The three primary colors (red, green and blue) are used in various combinations to achieve the desired color.

The combinations of red, green, and blue that define foreground and background colors can be specified in two notations. The first is RGB (Red-Green-Blue), and the second is HSL (Hue-Saturation-Luminosity). The notation must be selected before you can redefine pens (if no notation type is specified, the “terminal” uses the last notation specified, or RGB notation at power-up). To select a notation type, use the $E_{c\&.v}$ escape sequence used above:

$E_{c\&.v} n<parameter>$

where n is 0 (for RGB) or 1 (for HSL), and $<parameter>$ is the letter **m**. Thus, the sequence

$E_{c\&.v} 1M$

selects HSL notation. It does nothing more.

To specify the quantity of red (hue), green (saturation), and blue (luminosity) to appear in your background and foreground colors, the a , b , c , x , y , and z parameters are used. These parameters have the following meanings:

- a specifies the amount of red (hue) used in the foreground.
- b specifies the amount of green (saturation) used in the foreground.
- c specifies the amount of blue (luminosity) used in the foreground.
- x specifies the amount of red (hue) used in the background.
- y specifies the amount of green (saturation) used in the background.
- z specifies the amount of blue (luminosity) used in the background.

Each a , b , c , x , y , and z parameter specified is preceded by a number in the range 0 through 1, in increments of 0.01. The following table gives the values needed to define the eight principle colors:

Sample RGB/HSL Color Definition Values

R	G	B	Color	H	S	L
0	0	0	Black	X	X	0
0	0	1	Blue	.66	1	1
0	1	0	Green	.33	1	1
0	1	1	Cyan	.5	1	1
1	0	0	Red	1	1	1
1	0	1	Magenta	.83	1	1
1	1	0	Yellow	.16	1	1
1	1	1	White	X	0	1

X = don't care (may be any value between 0 and 1)

The following tables provide algorithms for explicitly defining the ranges of the parameters mentioned in the previous table for the Model 236 computer with color video.

HSL Definition Algorithm

COLOR SELECTED	parm. 1 H RANGE	parm. 2 S RANGE	parm. 3 L RANGE
BLACK	don't care	don't care	< 0.25
WHITE	don't care	<0.25	>= 0.25
RED	.00- .08	>= 0.25	>= 0.25
YELLOW	.09- .24	>= 0.25	>= 0.25
GREEN	.25- .41	>= 0.25	>= 0.25
CYAN	.42- .58	>= 0.25	>= 0.25
BLUE	.59- .74	>= 0.25	>= 0.25
MAGENTA	.75- .91	>= 0.25	>= 0.25
RED	.92-1.00	>= 0.25	>= 0.25

In the RGB color method, when N represents the largest-valued (most intense) color of the three color specifications, colors are selected as follows:

RGB Definition Algorithm

COLOR SELECTED	parm.1 RED RANGE	parm. 2 GREEN RANGE	parm. 3 BLUE RANGE
BLACK	<.25 or <N/2	<.25 or <N/2	<.25 or <N/2
WHITE	>=.25 and >=N/2	>=.25 and >=N/2	>=.25 and >=N/2
YELLOW	>=.25 and >=N/2	>=.25 and >=N/2	<.25 or <N/2
GREEN	<.25 or <N/2	>=.25 and >=N/2	<.25 or <N/2
CYAN	<.25 or <N/2	>=.25 and >=N/2	>=.25 and >=N/2
BLUE	<.25 or <N/2	<.25 or <N/2	>=.25 and >=N/2
MAGENTA	>=.25 and >=N/2	<.25 or <N/2	>=.25 and >=N/2
RED	>=.25 and >=N/2	<.25 or <N/2	<.25 or <N/2

One final parameter, *i*, is needed. It is used to assign a pen number to the newly-defined color pair. Thus, the escape sequence for changing a color pair definition is:

```
Esc & v <0|1> m n a n b n c n x n y n z <pen#> I
```

where either a 0 or a 1 precedes the *m* parameter (selecting either RGB or HSL notation, respectively), and *n* is one of the legal values from the tables. <pen#> is an integer in the range 0 thru 7 which precedes the *i* parameter, and defines that pen number to be the color pair specified by the preceding *a*, *b*, *c*, *x*, *y*, and *z* parameters. Omitting any *a*, *b*, *c*, *x*, *y*, or *z* parameter causes a value of 0 to be assigned to the omitted parameter by default. Also, the background parameters can be specified before the foreground parameters.

Examples

```
^c&v 0m 1a 0b 0c 0x 1y 0z 5I
```

This example re-defines pen 5 to specify red characters on a green background. (Note that 236 computers with color video ignore the green background specification and assign a black one instead.) This example is equivalent to

```
^c&v 0m 1a 1y 5I
```

since omitted parameters (a, b, c, x, y, z) are given default values of 0.

```
^c&v 1m .66a 1b 1c 3i 0m 1c 1x 1y 6I
```

This example re-defines pen 3 to specify blue characters on a black background (HSL notation), and pen 6 to specify blue characters on a yellow background (RGB notation). This example illustrates how multiple pens can be defined on a single line using different notations. (Again, note that the Model 236 with color video will reject the background specification of pen 6, and will use black instead.)

If you should specify color on color when setting up color definitions on the Model 236 computer with color video, you will find that the foreground color will remain as chosen and the background color will default to black.

```
^c&v 5I
```

This example re-defines pen 5 to specify a black foreground and a black background, using the previous notation type.

Note

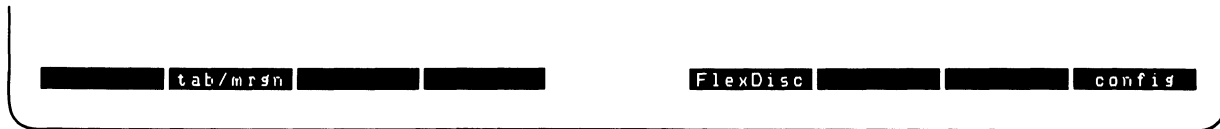
Supplying neither a foreground nor a background color when defining a color pair causes both the foreground and background to be black.

Configuring the ITE

The Internal Terminal Emulator is designed so that the various ITE characteristics can be configured quickly by displaying configure “menus” on the screen and then using system function keys to change the content of these menus.

Configuration Function Keys

To gain access to the configuration menus through the keyboard, press the function key **F9** (labeled AIDS on the system console overlay). This causes the following softkey display to appear at the bottom of the screen:



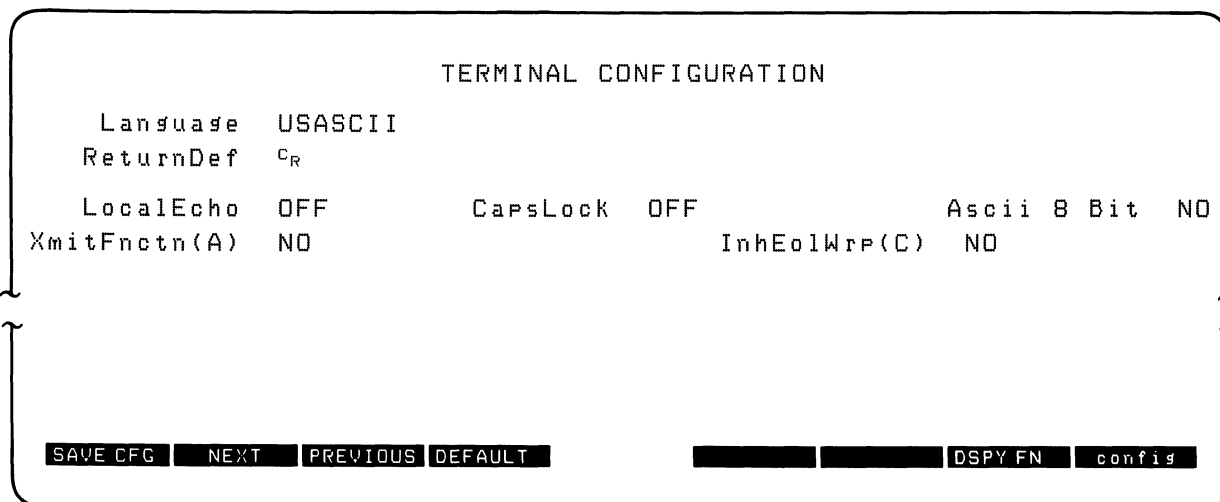
The function keys **F2** (tab/mrgn) and **F5** (FlexDisc) were covered in the section “Using the Internal Terminal Emulator” along with a brief discussion on the function key **F8** (config). When you press **F8** (config) a new softkey display appears at the bottom of your screen.



Pressing **F5** (terminal) fills the display with the **Terminal Configuration Menu** which is covered next.

Terminal Configuration Menu

After pressing **F5** (terminal) your display should look like this:



Description of Fields

The TERMINAL CONFIGURATION menu contains a set of unprotected fields that you access using the **TAB** key. Note that all fields can be changed using function keys **f2** (NEXT) and **f3** (PREVIOUS) with the exception of **ReturnDef** which is changed using the function key **f7** (DSPY FN).

There are seven fields which can be changed using the function keys as defined in the next section. These fields are described as follows:

ReturnDef	<p>specifies the definition of the ENTER key (labeled RETURN key on the overlay). The default definition is an ASCII c_R. The definition may consist of up to two characters. If the second character is a space, it is ignored and only the first character is used.</p> <p>Default: c_R space</p>
LocalEcho	<p>specifies whether characters entered through the keyboard are both displayed on the screen and transmitted to the host computer.</p> <p>ON ($\text{E}_C\&K$ 1L)</p> <p>Characters entered through the keyboard are both displayed on the screen and transmitted to the host computer.</p>
CapSLock	<p>determines whether the ITE generates the full 128-character ASCII set or only Teletype-compatible codes.</p> <p>ON ($\text{E}_C\&K$ 1C)</p> <p>The ITE generates only Teletype-compatible codes: uppercase ASCII (00-5F, hex) and DEL (7F, hex). Unshifted alphabetic keys (a-z) generate the codes for their uppercase equivalents. The {, and } keys generate the codes for [, \, and] respectively. The keys for generating ~ and ` are disabled.</p> <p>OFF ($\text{E}_C\&K$ 0C)</p> <p>The ITE generates the full 128-character ASCII set of codes.</p> <p>Default: OFF</p>
XmitFctn(A)	<p>determines whether the escape code functions are executed at the ITE and transmitted to the host computer.</p> <p>YES ($\text{E}_C\&S$ 1A)</p> <p>The escape code sequences generated by control keys such as ↑ and ↓ are transmitted to the host computer. If LocalEcho is ON, the function is also performed locally.</p> <p>NO ($\text{E}_C\&S$ 0A)</p> <p>The escape code sequences for the major function keys are executed locally but NOT transmitted to the host computer.</p> <p>Note that display functions will emit E_C Z and E_C Y to a host computer.</p> <p>Default: NO</p>

`InhEolWrP(C)` designates whether or not the end-of-line wrap is inhibited.

NO (`E_C&S 0C`)

When the cursor reaches the right margin it automatically moves to the left margin in the next lower line (a local carriage return and line feed are generated).

YES (`E_C&S 1C`)

When the cursor reaches the right margin it remains in that screen column until an explicit carriage return or other cursor movement function is performed (succeeding characters overwrite the existing character in that screen column).

`Language` changes the character set on your keyboard to one of the following when you press function key `f2` or `f3`:

USASCII (United States)

SVENSK/SUOMI (Swedish/Finnish)

FRANCAIS azM (French AZERTY¹ layout with mutes)

FRANCAIS qwM (French QWERTY layout with mutes)

FRANCAIS az (French AZERTY¹ layout)

FRANCAIS qw (French QWERTY layout)

DEUTSCH (German)

ESPAÑOL M (Spanish with mutes)

ESPAÑOL (Spanish)

KATAKANA (Japanese)

The system console overlay works the same on all the keyboards listed. Diagrams of the previously mentioned keyboards can be found at the end of this article.

Series 200 computers support two character sets which contains the special characters associated with all of the international languages. These character sets are: Extended Roman and KATAKANA. The following charts show these character sets. Note that blank spaces in the chart are given the character **hp**; however, they have not been shown on the charts so that the left half of the chart or standard 7-bit ASCII code could be shown as separate from the right half of the chart or 8-bit code. The 8-bit code can be accessed by configuring the field **ASCII 8 Bit** to **YES**.

¹ The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers.

COL BIT				8	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
ROW BIT				7	0	0	0	1	1	1	1	1	0	0	0	1	1	0	1	1	1	
4	3	2	1	5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
					0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
0	0	0	0	0	N _U	D _L	(SP)	ø	@	P	`	p					—	â	À			
0	0	0	1	1	S _M	D _I	!	1	A	Q	a	q						ê	î			
0	0	1	0	2	S _X	D ₂	"	2	B	R	b	r						ô	Ø			
0	0	1	1	3	E _X	D ₃	#	3	C	S	c	s					°	û	Æ			
0	1	0	0	4	E _T	D ₄	\$	4	D	T	d	t						á	à			
0	1	0	1	5	E _O	N _K	%	5	E	U	e	u					ç	é	í			
0	1	1	0	6	R _K	S _T	&	6	F	V	f	v					ñ	ó	ø			
0	1	1	1	7	Q	E _B	'	7	G	W	g	w					ñ	ú	æ			
1	0	0	0	8	E _S	S _N	()	8	H	X	h	x				'	i	à	Ä			
1	0	0	1	9	N _T	E _M)	9	I	Y	i	y				'	ç	è	ì			
1	0	1	0	10	L _T	S _B	*	:	J	Z	j	z				^	x	ò	ö			
1	0	1	1	11	Y _T	E _C	+	;	K	[k	{				~	£	ù	ü			
1	1	0	0	12	F _T	F _S	,	<	L	\	l					~		ä	é			
1	1	0	1	13	S _R	E _S	-	=	M	J	m	}					§	ë	ï			
1	1	1	0	14	S _O	R _S	.	>	N	^	n	~						ö	ß			
1	1	1	1	15	S _T	U _S	/	?	O	_	o	*				£		ü				☒

Extended Roman Character Set

To use the Extended Roman Character Set, configure your TERMINAL CONFIGURATION menu to the language you want, ASCII 8 Bit should be set to YES and your terminal (ITE) should be in the remote mode. All characters for the various languages mentioned in the menu are now available to you except KATAKANA.

COL BIT		8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
ROW BIT		7	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	1
		6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
		5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
		4	3	2	1														
		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
0	0	0	1																
0	0	1	0																
0	0	1	1																
0	1	0	0																
0	1	0	1																
0	1	1	0																
0	1	1	1																
1	0	0	0																
1	0	0	1																
1	0	1	0																
1	0	1	1																
1	1	0	0																
1	1	0	1																
1	1	1	0																
1	1	1	1																

KATAKANA Character Set

To use the KATAKANA Character Set, configure your TERMINAL CONFIGURATION menu to the language KATAKANA, ASCII 8 Bit should be set to YES and your terminal (ITE) should be in the remote mode. To type Japanese ASCII characters, press **CTRL** - **'**. If you want KATAKANA characters, press **CTRL** - **.**

For the French keyboard layouts, the AZERTY² and QWERTY designations refer to the location of the A, Z, Q, and W keys as follows:

AZERTY: Row 3 = A Z E R T Y
 Row 2 = Q S D (etc.)
 Row 1 = W X C (etc.)

QWERTY Row 3 = Q W E R T Y
 Row 2 = A S D
 Row 1 = Z X C (etc.)

For the French and Spanish keyboard layouts, the mutes designation refers to the manner in which certain accent character keystrokes are handled (^ and `` on the French layout and ` on the Spanish). If the mutes are enabled, those keystrokes will generate the particular accent character but will NOT move the cursor. If you then type an applicable vowel, the vowel will appear in the same character position as the accent and the cursor then moves to the next column (if you type any character other than an applicable vowel, however, the character will replace the accent character).

ASCII 8 Bit transmits from full set of 8-bit codes when enabled (YES) and transmits only codes less than 128 when disabled (NO).

Values: YES (E_c&k 1I) = 8-bit codes.
 NO (E_c&k 0I) = Standard 7-bit codes.

² The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers.

Changing the Fields

To change the fields in the display, press the **TAB** key until the cursor is located under the field you wish to change. Next, use the following function keys to change the state of the field.

- SAVE CFG saves the fields on the configuration menu which you have altered.
- NEXT changes the setting of the field you are presently in to the next setting in that field. For example, if you press **TAB** until you are located at the field **LocalEcho OFF** and then press **f2** the **LocalEcho** field changes to **ON**.
- PREVIOUS changes the setting of the field you are presently in to the previous setting in that field. For example, if you press **TAB** until you are located at the field **LocalEcho ON** and then press **f3** the **LocalEcho** field changes to **OFF**.
- DEFAULT causes the fields in the menu to be filled with their default value. The default values are as shown in the TERMINAL CONFIGURATION display at the beginning of this section. The only exception is the "language" field it defaults to the language option shipped with your system.
- DSPY FN enables and disables the display functions mode. Pressing the key once enables the display functions mode and pressing it a second time disables it. When enabled * appears in the function key label box. You use the display function mode for entering ASCII control characters in the **ReturnDef** field. Note that this implementation of display function is separate from that which is enabled/disabled via the mode selection keys. Enabling or disabling display functions mode using this function key does **NOT** alter the effect of the DISPLAY FCTNS mode selection key (and vice versa).
- config removes the menu from the screen and changes the function key labels to the following:



ITE Escape Sequences

Several Internal Terminal Emulator (ITE) keyboard functions can be activated or controlled by a remote computer by use of escape-code sequences. The effect is identical to using non-ASCII keys on the ITE keyboard. Escape sequences consist of the escape-code character followed by one or more visible (non-control) ASCII characters.

The sequences listed in this section are recognized and executed by the ITE whether they are received from the data communication link or from the keyboard, although the keyboard is seldom used for escape sequences. If an illegal or unrecognized sequence is received, the ITE ignores the message and all subsequent data until one of the following characters is received: @, A thru Z, [, \, ^, _, carriage-return, escape-code or any ASCII 7-bit code less than the character space.

Sequence Types

There are two general categories of escape sequences. The two-character ITE control sequences are used primarily for ITE, screen, and cursor control. Most of these sequences are equivalent to keyboard operations that involve a single keystroke or the simultaneous pressing of two keys.

The extended escape sequences consist of the escape-code character followed by at least two subsequent characters. They are used, either for functions that are not included in the two-character sequences, or for sequences whose inherent complexity requires two or more characters in addition to the escape-code in order to define the operation. Absolute and relative cursor addressing are examples of operations that require longer control sequences.

Escape Sequences for ITE Control

Escape Code	Function
E _c 1	Set tab
E _c 2	Clear tab
E _c 3	Clear all tabs
E _c 4	NOT IMPLEMENTED (Set left margin)
E _c 5	NOT IMPLEMENTED (Set right margin)
E _c 9	NOT IMPLEMENTED (Clear all margins)
E _c @	NOT IMPLEMENTED (Makes the terminal program wait approximately one second.)
E _c A	Cursor up
E _c B	Cursor down
E _c C	Cursor right
E _c D	Cursor left
E _c E	Hard reset (power on reset of ITE)
E _c F	Cursor home down
E _c G	Move cursor to the left margin
E _c H	Cursor home up
E _c I	Horizontal tab
E _c J	Clear screen from cursor to the end of memory.

Escape Sequences for ITE Control (continued)

Escape Code	Function
E _c K	Clear line from cursor to end of line
E _c L	Insert line
E _c M	Delete line
E _c P	Delete character
E _c Q	Start insert character mode
E _c R	End insert character mode
E _c S	Roll up
E _c T	Roll down
E _c U	Next page
E _c V	Previous page
E _c W	NOT IMPLEMENTED (Format mode on)
E _c X	NOT IMPLEMENTED (Format mode off)
E _c Y	Enables display functions mode
E _c Z	Disables display functions mode
E _c [NOT IMPLEMENTED (Start unprotected field)
E _c]	NOT IMPLEMENTED (End unprotected/transmit-only field)
E _c ^	NOT IMPLEMENTED (Primary terminal status request)
E _c `	Sense cursor position(relative)
E _c a	Sense cursor position (absolute)
E _c b	NOT IMPLEMENTED (Unlock keyboard)
E _c c	NOT IMPLEMENTED (Lock keyboard)
E _c d	NOT IMPLEMENTED (Transmit a block of text to computer)
E _c f	NOT IMPLEMENTED (Modem disconnect)
E _c g	Soft reset (of ITE)
E _c h	Cursor home up.
E _c i	Backtab
E _c j	NOT IMPLEMENTED (Begin User Key Definition mode)
E _c k	NOT IMPLEMENTED (End User Keys Definition mode)
E _c l	NOT IMPLEMENTED (Begin Memory Lock mode)
E _c m	NOT IMPLEMENTED (End Memory Lock Mode)
E _c P	Default value for softkey <input type="button" value="f1"/>
E _c q	Default value for softkey <input type="button" value="f2"/>
E _c r	Default value for softkey <input type="button" value="f3"/>
E _c s	Default value for softkey <input type="button" value="f4"/>
E _c t	Default value for softkey <input type="button" value="f5"/>
E _c u	Default value for softkey <input type="button" value="f6"/>
E _c v	Default value for softkey <input type="button" value="f7"/>
E _c w	Default value for softkey <input type="button" value="f8"/>
E _c z	NOT IMPLEMENTED (Initiate terminal self test)
E _c ~	NOT IMPLEMENTED (Secondary terminal status request)

Extended Escape Sequences

Escape Code Sequence	Function
$E_c \& a \langle col \rangle c \langle row \rangle Y$	Moves the cursor to column “col” and screen row “row” or the screen (screen relative addressing).
$E_c \& a \langle col \rangle c \langle row \rangle R$	Moves the cursor to column “col” and row “row” in memory (absolute addressing).
$E_c \& a \pm \langle col \rangle c \pm \langle row \rangle Y$	Moves the cursor to column “col” and row “row” (on the screen) relative to its present position (“col” and “row” are signed integers). A positive number indicates right or downward movement and a negative number indicates left or upward movement.
$E_c \& a \pm \langle col \rangle c \pm \langle row \rangle R$	Moves the cursor to column “col” and row “row” (on the screen) relative to its present position (“col” and “row” are signed integers). A positive number indicates right or downward movement and a negative number indicates left or upward movement.
$E_c \& q 0 L$	NOT IMPLEMENTED (Unlock configuration)
$E_c \& q 1 L$	NOT IMPLEMENTED (Lock configuration)
$E_c \& d \langle char \rangle$	Selects the display enhancement indicated by <char> to begin at the present cursor position. <char> can be E or A thru Q .
$E_c \& k \langle x \rangle A$	AUTO LF enable: $x = 1$; disable: $x = 0$
$E_c \& k \langle x \rangle B$	NOT IMPLEMENTED (BLOCK enable: $x = 1$; disable: $x = 0$)
$E_c \& k \langle x \rangle C$	Caps Lock enable: $x = 1$; disable: $x = 0$
$E_c \& k \langle x \rangle I$	ASCII8Bits enable: $x = 1$; disable: $x = 0$
$E_c \& k \langle x \rangle J$	NOT IMPLEMENTED (FrameRate 50 Hz : $x = 1$; 60 Hz : $x = 0$)
$E_c \& k \langle x \rangle L$	LocalEcho enable: $x = 1$; disable: $x = 0$
$E_c \& k \langle x \rangle M$	NOT IMPLEMENTED (MODIFY ALL enable: $x = 1$; disable: $x = 0$)
$E_c \& k \langle x \rangle N$	NOT IMPLEMENTED (SPOWLatch enable: $x = 1$; disable: $x = 0$)
$E_c \& k \langle x \rangle P$	Caps Mode is primarily used as a typing convenience and affects only the 26 alphabetic keys. When it is enabled, all unshifted alphabetic keys generate uppercase letters and all shifted alphabetic keys generate lowercase letters. enable: $x = 1$; disable: $x = 0$
$E_c \& k \langle x \rangle R$	REMOTE enable: $x = 1$; disable: $x = 0$

Extended Escape Sequences (continued)

Escape Code Sequence	Function
$E_c \& s \langle x \rangle A$	xmitFnctn(A) enable: x = 1; disable: x = 0
$E_c \& s \langle x \rangle B$	NOT IMPLEMENTED (SPOW(B) enable: x = 1; disable: x = 0)
$E_c \& s \langle x \rangle C$	InhEolWrp(C) enable: x = 1; disable: x = 0
$E_c \& s \langle x \rangle D$	NOT IMPLEMENTED (Line/Page(D) enable: x = 1; disable: x = 0)
$E_c \& s \langle x \rangle G$	NOT IMPLEMENTED (InfHndShk(G) enable: x = 1; disable: x = 0)
$E_c \& s \langle x \rangle H$	NOT IMPLEMENTED (Inh DC2(H) enable: x = 1; disable: x = 0)
$E_c \& w 12F$	Turns on the display window (top 24 rows)
$E_c \& w 13F$	Turns off the display window
$E_c * d \langle \text{parameters} \rangle$	List of $\langle \text{parameters} \rangle$ for display control: E Turns on alphanumeric display; F Turns off alphanumeric display. When terminating a string of escape sequences with these parameters use a capital letter.
$E_c * s \langle \text{Parameter} \rangle *$	Read device I.D. Status is parameter number 1.
$E_c \& j \langle x \rangle$	Enables and disables the function keys (f1 thru f8). If x equals: A Display the Modes set of function key labels, B Enable the User function keys. (The user key labels are displayed.) @ Remove the function key labels from the screen. The User function keys, however, are still active.
$E_c \& f \langle \text{attribute} \rangle a$ $\langle \text{key} \rangle k$ $\langle \text{label length} \rangle d$ $\langle \text{string length} \rangle L$	Defines the function keys. Information on how this is done can be found in the function key group section of this manual under the definition of user keys.

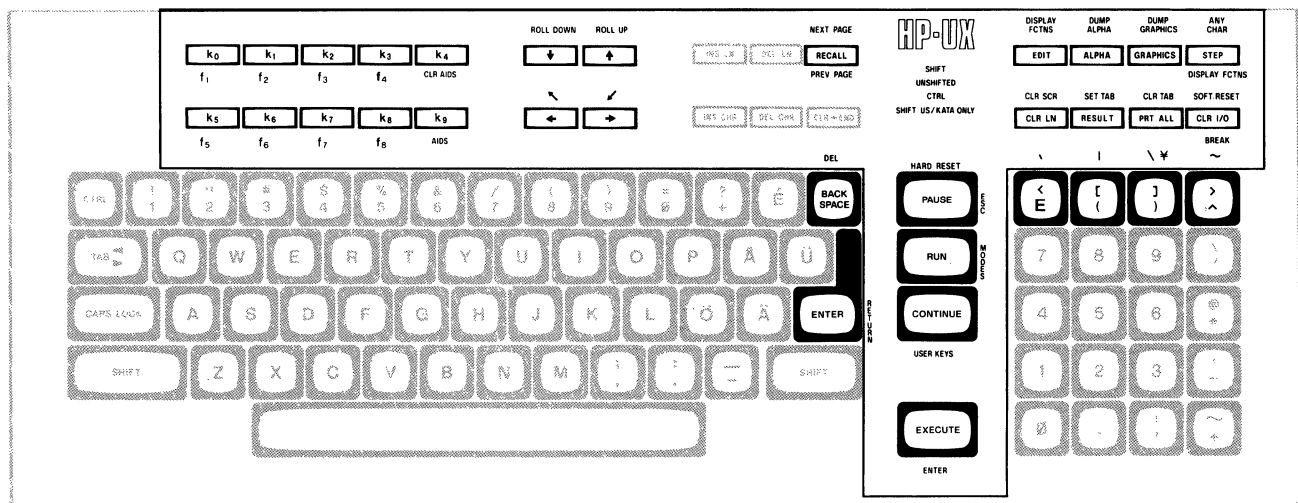
If an escape sequence is not recognized, the terminal ignores subsequent characters until ASCII decimal characters 0 thru 31 or 64 thru 95 is received, terminating the sequence. Note that E_c will terminate the old sequence and start a new one.

Keyboard Diagrams for Other Languages

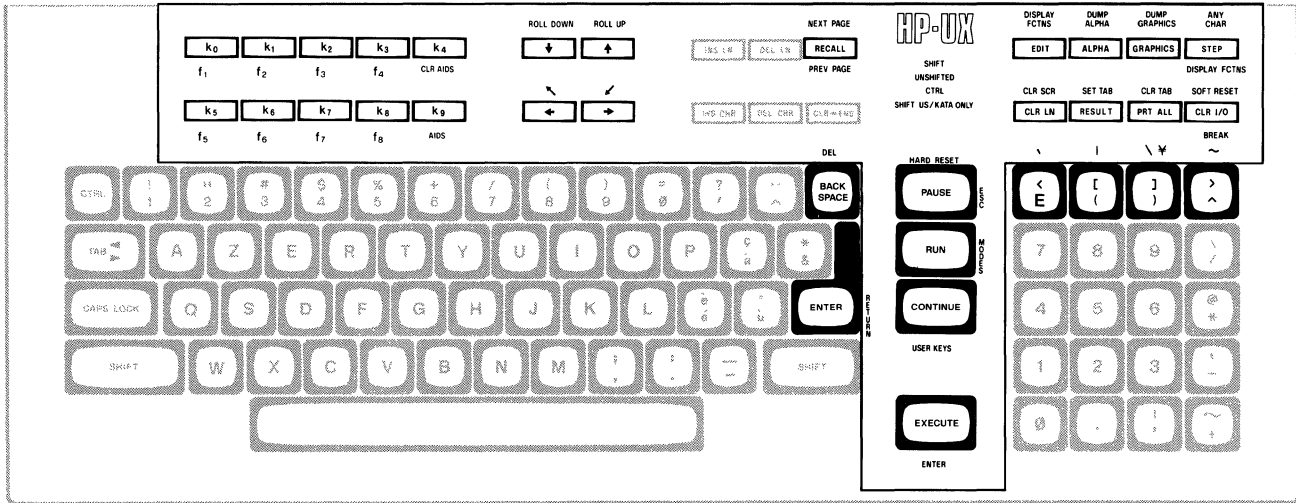
This section shows diagrams of the types of keyboards which are available to the Series 200 user. The keyboard option you now have can be configured to any one of the languages by use of the **TERMINAL CONFIGURATION** menu previously mentioned in this article.

To change to another keyboard language use the terminal configuration menu and select the **Language** field you want. Then change the **ASCII 8 Bit** field to YES. Next, save the changed configuration menu. Note that the languages accessed through the terminal configuration menu are not available in the vi editor. The vi editor only uses the first 128 ASCII characters of your systems particular character set. For more information on your keyboard, read the appropriate manual sent with your system.

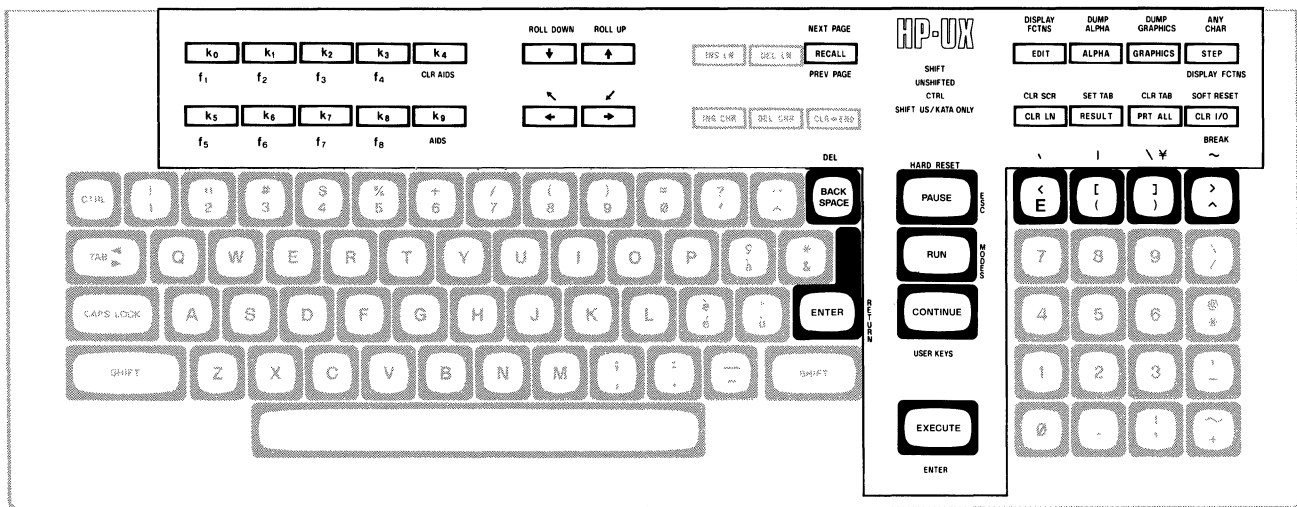
You will notice as you look at the keyboards that a majority of the character key and numeric key labels have changed. To use your keyboard effectively in anyone of these languages, you will have to re-label the key caps.



SVENSK/SUOMI (Swedish/Finnish)

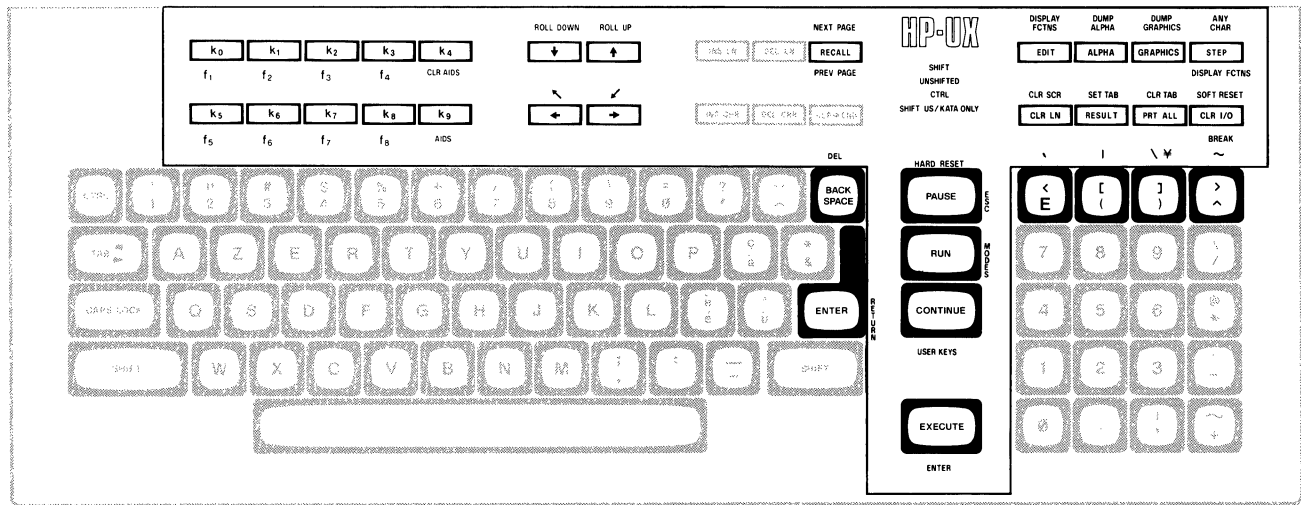


FRANCAIS azM (French AZERTY³ layout with mutes is not available)

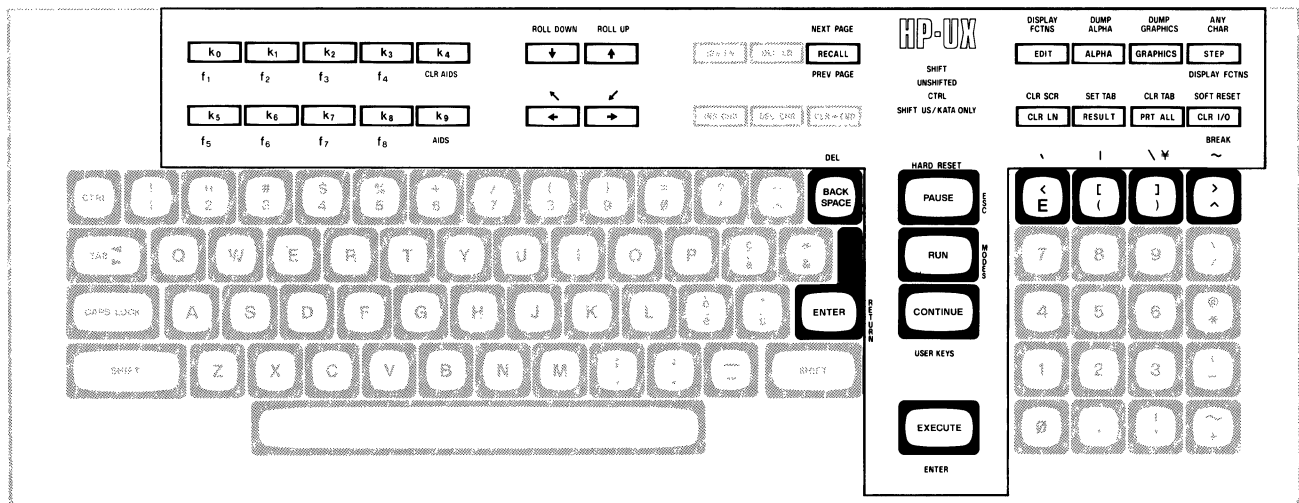


FRANCAIS qwM (French QWERTY layout with mutes)

³ The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers.



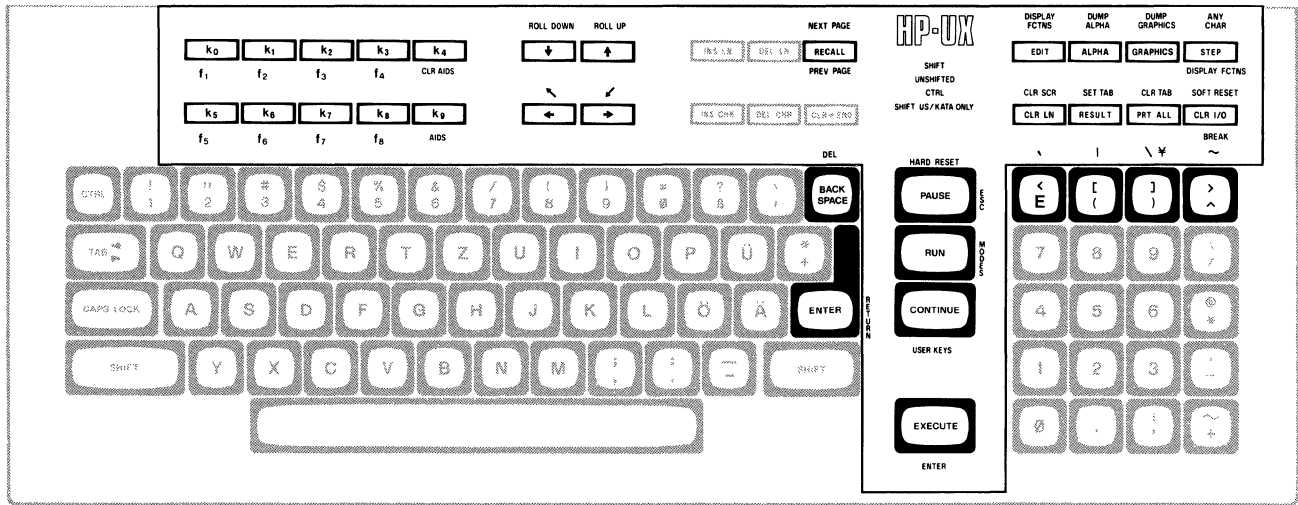
FRANCAIS az (French AZERTY⁴ layout is not available)



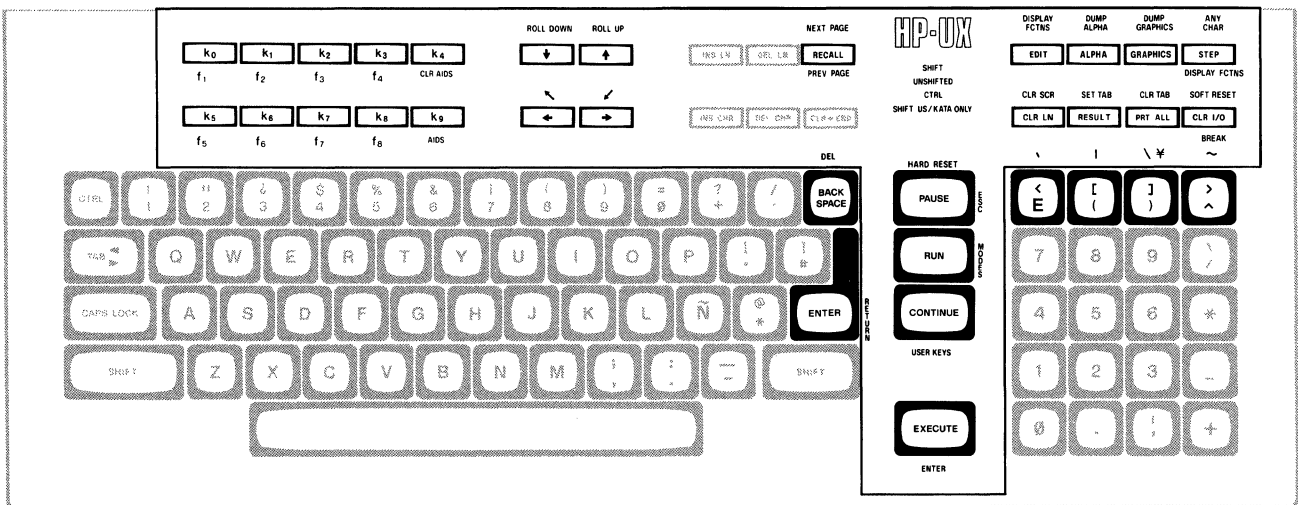
FRANCAIS qw (French QWERTY layout)

⁴ The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers.

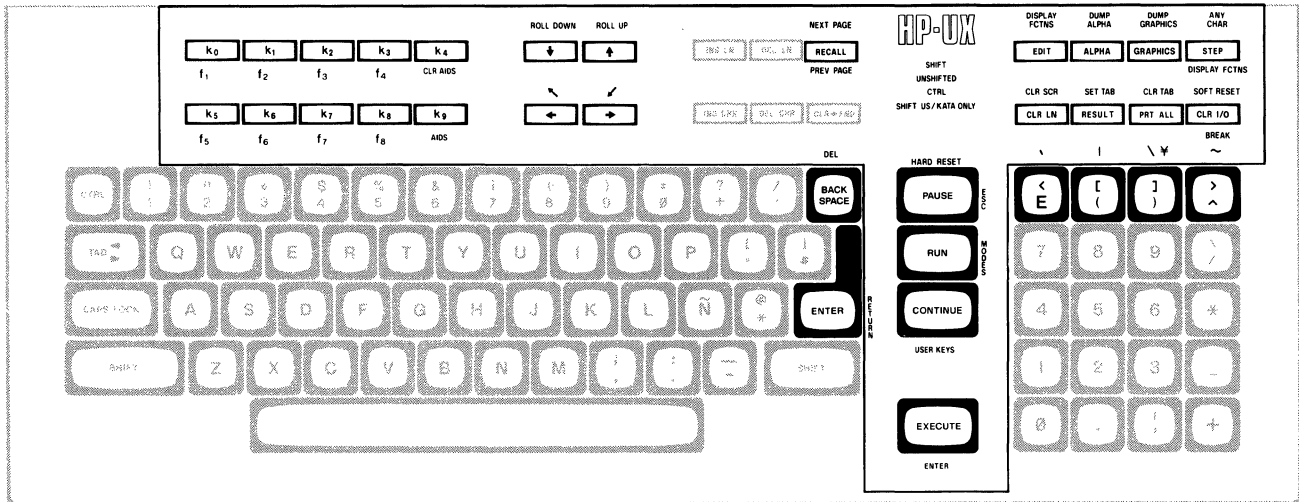
40 Series 200 Console



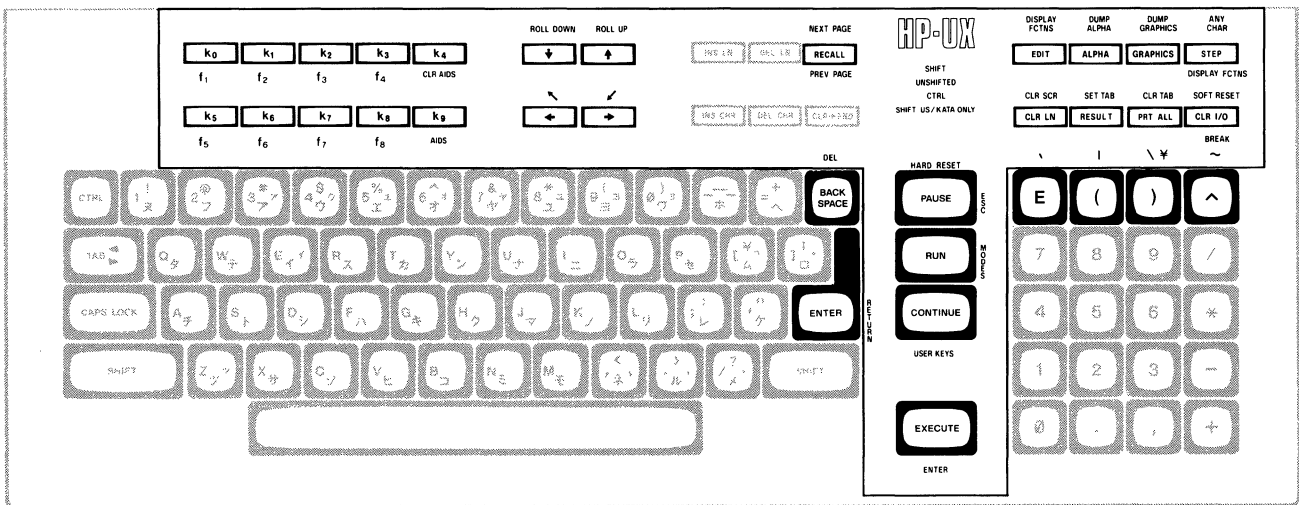
DEUTSCH (German)



ESPAÑOL M (Spanish with mutes)



ESPAÑOL (Spanish)



KATAKANA (Japanese)

Notes

Table of Contents

HP-UX and the HP 9000 Model 520 as System Console

The Keyboard.....	2
Alphanumeric Group.....	2
Numeric Pad Group.....	2
Display Control Group.....	2
Setting and Clearing Margins.....	2
Setting and Clearing Tab Stops.....	3
Cursor Control.....	3
Edit Group.....	6
Insert Character Mode.....	7
Function Key Group.....	7
Modes.....	7
AIDS.....	8
User Keys.....	9
User-definable Keys.....	9
The Display.....	11
Memory Addressing Scheme.....	11
Row Addressing.....	11
Column Addressing.....	11
Cursor Sensing.....	12
Absolute Sensing.....	12
Relative Sensing.....	12
Cursor Positioning.....	12
Screen Relative Addressing.....	13
Example.....	13
Absolute Addressing.....	13
Example.....	14
Cursor Relative Addressing.....	14
Example.....	15
Combining Absolute and Relative Addressing.....	15
Display Enhancements.....	15
Raster Control.....	16
Accessing Color.....	16
Color Pair.....	16
Pen#.....	17
Selecting a Pen (Color Pair).....	17
Changing Pen Definitions.....	17
Examples.....	19

Controlling Configuration and Status.....	20
Re-configuring the Terminal.....	20
Sending Terminal Status.....	21
Primary Terminal Status	21
Secondary Terminal Status.....	23

HP-UX and the HP 9000 Model 520 As System Console

The **system console** is the terminal to which HP-UX sends system loader messages and soft system error messages. Like other terminals on an HP-UX system, it is also used for general system access (such as logging in, running programs, and entering data). The system console:

- must be connected to the computer via select code 0.
- must not be connected via a modem.
- must have a device file named */dev/console*.

Each system must have a system console. When HP-UX is run on the HP 9000 Model 520, the computer's keyboard and display act as the system console. This article describes the HP 9000 Model 520 as a terminal and as system console. It also discusses the methods of accessing the "terminal's" features: from the keyboard and from a program or command (via escape sequences).

HP-UX treats your HP 9000 Model 520 as six independent devices. The first device is a 32-bit mini-computer composed of a central processing unit, an I/O processor and memory. The second, third, fourth and fifth devices are: the built-in thermal printer, the built-in flexible disc drive, the built-in Winchester disc drive, and the graphics display. The sixth device is the computer's keyboard and display. This last device is the terminal and system console discussed in this article.

The display portion of the "terminal" consists of a display screen and display memory. The display cursor (a blinking underscore on the screen) indicates where the next character entered appears. As you enter characters, each is displayed at the cursor position, the ASCII code for the character is recorded at the associated position in display memory, and the cursor moves to the next character position on the screen. As the screen becomes full, newly entered data causes existing lines to be forced off the screen. Data lines forced off the screen are still maintained in display memory and can subsequently be moved back onto the screen. The size of display memory is determined by the HP-UX configuration. Once the display memory is full, additional data entered causes the older data in display memory to be lost.

Throughout this article, the sequence `\E` represents the escape character. Supplying an invalid escape sequence causes that sequence to be ignored. Escape sequences with optional or required parameters (referred to as "parameterized escape sequences") must be terminated by an upper case character before the sequence is implemented.

The Keyboard

The Model 520's keyboard is divided into major functional groups: the alphanumeric group, the numeric pad group, the display control group, the edit group, and the function group. Each function group is discussed in the sections below, with an emphasis on features and their access.

Alphanumeric Group

This group of keys is similar to a standard typewriter keyboard and consists of the alphabetic, numeric, and symbol keys. Included are lower and uppercase alphabetic characters, ASCII control codes, punctuation characters, and some commercial symbols.

Numeric Pad Group

The numeric group of keys is located to the right of the alphanumeric keys. The layout of the numeric key pad is similar to that of a standard office calculator. These keys are convenient for high-speed entry of large quantities of numeric data.

Display Control Group

The display control group consists of the keys that control the location of the cursor on the display. Each display control key and its function is described in the sections that follow. The escape code for accessing each display control feature is provided with each display control key. Some display control features can only be accessed via an escape sequence; no key is associated with the feature. The escape code for such features is also provided in the sections that follow.

Setting and Clearing Margins

You can redefine the left and/or right margin. These margins affect the cursor positioning for certain functions (such as carriage-return, home up, home down, etc.) and establish operational bounds for the insert character and delete character functions. In addition, the left margin is always an implicit tab stop. Data to the left of the left margin or to the right of the right margin is still accessible.

When you are entering data through the keyboard and the cursor reaches the right margin, it automatically moves to the left margin in the next lower line. When you press **RETURN** the cursor moves to the left margin in the current line if auto line feed mode is disabled or to the left margin in the next lower line if auto line feed mode is enabled.

Margins can be set with the AIDS keys (discussed in a later section) or with escape sequences:

- \E4 - set the left margin at the current cursor location.
- \E5 - set the right margin at the current cursor location.
- \E9 - clear both margins; by default the left margin becomes 1, the right margin becomes 80.

Attempting to set the left margin to the right of the right margin (or the right margin to the left of the left margin) causes the new margin to be rejected; the system beeps to notify you that the new margin was not accepted.

Setting and Clearing Tab Stops







You can define a series of tab stops to which you can move the cursor using the tab and back tab functions shown below. From the keyboard you set and clear tab stops using the **TAB SET** and **TAB CLEAR** keys. To set a tab stop, move the cursor to the desired location and press **TAB SET**. To clear a tab stop, move the cursor to the tab stop position and press **TAB CLEAR**. Additionally, you may use the functions provided with the AIDS keys (discussed in a later section) to set and clear tab stops.

Note that the left margin is always an implicit tab stop and cannot be cleared. The escape sequences to set and clear tab stops are:

- \E1 - set a tab stop at the current cursor position.
- \E2 - clear a tab stop previously set at the current cursor position.
- \E3 - clear all tab stops currently set. Note that this feature is available only from softkeys (as are the margin functions described above).

Cursor Control

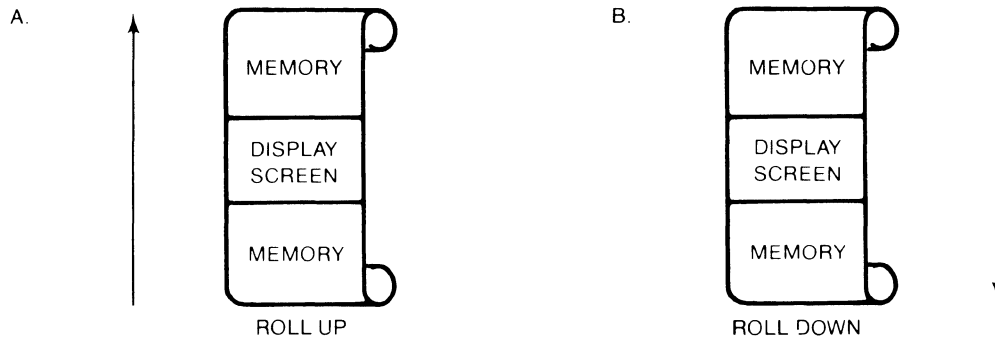
Several keys exist on keyboard for changing the location of the cursor:

Key	Escape Sequence	Feature
	\EA	Move the cursor up one row in the current column position. Holding the key down causes the cursor to move continuously, row by row, until the key is released. When the cursor is in the top row of the screen, moving the cursor up actually moves the cursor to the same column position in the bottom row of the screen.
	\EB	Move the cursor down one row in the current column position. Holding the key down causes the cursor to move continuously, row by row, until the key is released. When the cursor is in the bottom row of the screen, moving the cursor down actually moves the cursor to the same column position in the top row of the screen.
	\EC	Move the cursor right one position in the current line; if the current position is the right margin, the cursor is moved to the left margin of the next line. Holding the key down causes the cursor to move continuously, column by column, until the key is released.
	\ED	Move the cursor left one position in the current line; if the current position is the left margin, the cursor is moved to the right margin of the previous line. Holding the key down causes the cursor to move continuously, column by column, until the key is released.
SHIFT 	\EH or \Eh	Home up: moves the cursor to the left margin in top row of the screen and rolls the text in display memory down as far as possible so that the first line in display memory appears in the top row of the screen.
SHIFT 	\EF	Home down: moves the cursor to the left margin in the bottom line of the screen and rolls the text in display memory up as far as necessary so that the last line in display memory appears immediately above the cursor position.

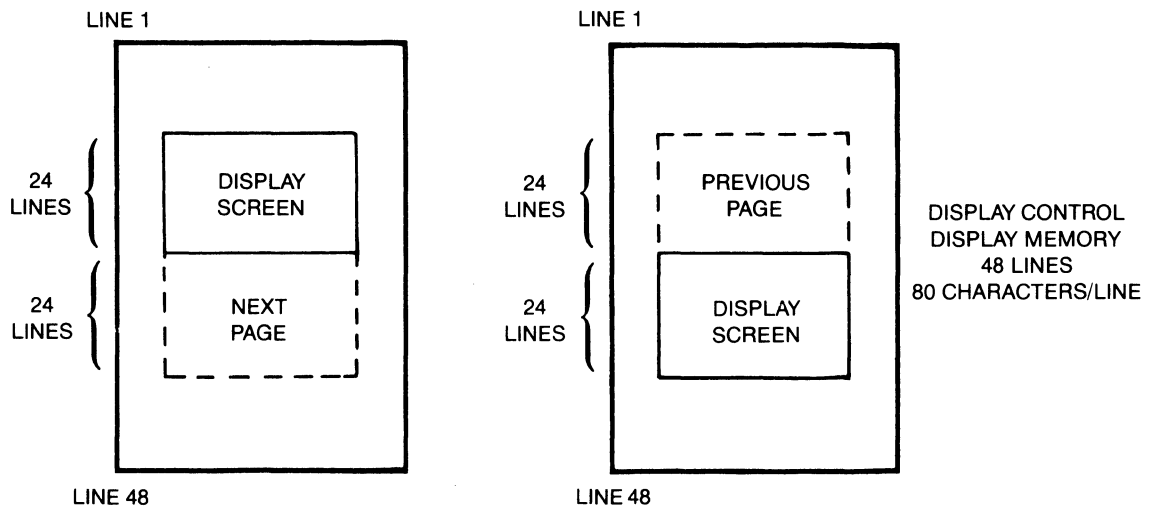
4 Model 520 Console

Key	Escape Sequence	Feature
none	\EG	Move the cursor to the left margin.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">TAB</div>	\EI	Move the cursor forward to the next tab stop.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">SHIFT</div> <div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block; margin-left: 5px;">TAB</div>	\Ei	Move the cursor backwards to the previous tab stop.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">ROLL</div> <div style="font-size: 0.8em; vertical-align: middle;">↑</div>	\ES	Roll the text in display memory up one row on the screen. The top row rolls off the screen, the remaining data rolls up one line on the screen, and a new line of data rolls from display memory into the bottom line of the screen. When the key is held down, the text continues to roll upward until the key is released or until the final line of data in display memory appears in the top row of the screen. In the latter case, pressing or continuing to press down the key has no further effect. The roll up and roll down functions are shown in the illustrations at the end of this section.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">ROLL</div> <div style="font-size: 0.8em; vertical-align: middle;">↓</div>	\ET	Roll the text in display memory down one row on the screen. The bottom row rolls off the screen, the remaining data rolls down one line on the screen, and a new line of data rolls from the display memory into the top line of the screen. When the key is held down, the text continues to roll down until the key is released or until the first line of data in display memory appears in the top row of the screen. In the latter case, pressing or continuing to press down the key has no further effect. The roll up and roll down functions are shown in the illustrations at the end of this section.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">SHIFT</div> <div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block; margin-left: 5px;">ROLL</div> <div style="font-size: 0.8em; vertical-align: middle;">↑</div>	\EU	Roll the text in display memory up so that the next page (see the explanation below) of data replaces the current page on the screen. If the key is held down, the operation is repeated until the key is released or until the final line in display memory appears in the top line of the screen. In the latter case, pressing or continuing to hold down the key has no further effect.
<div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block;">SHIFT</div> <div style="border: 1px solid black; border-radius: 5px; padding: 2px; display: inline-block; margin-left: 5px;">ROLL</div> <div style="font-size: 0.8em; vertical-align: middle;">↓</div>	\EV	<p>The cursor is placed at the left margin, at the top row of the display.</p> <p>Roll the text in display memory down so that the previous page (see the explanation below) of data replaces the current page on the screen. If the key is held down, the operation is repeated until the key is released or until the first line in display memory appears in the top line of the screen. In the latter case, pressing or continuing to hold down the key has no further effect.</p> <p>The cursor is placed at the left margin, at the top row of the display.</p>

The data in display memory can be accessed (displayed on the screen) in blocks that are known as "pages". A page consists of 24 lines of data. The current page is that sequence of lines which appears on the screen at any given time. The **previous page** is the preceding 24 lines in display memory. The **next page** is the succeeding 24 lines in display memory. This concept, along with the concept of rolling data through the display screen and memory, are shown in the following illustrations.



The "Roll" Data Functions



Previous Page and Next Page Concepts

Edit Group

The edit group consists of the keys that allow you to modify the data presented on the screen. Currently, however, the edited data cannot be read back by the system. Typically, these features are used to modify data presented by programs. For example, the *vi* text editor program uses these features.

You can edit data on the screen by simply overstriking the old data. In addition, the following edit keys and escape sequences may be used:

Key	Escape Sequence	Function
CLEAR SCN	\EJ	Removes from display memory, all characters from the current location of the cursor to the end of display memory.
CLEAR LINE	\EK	Removes from display memory, all characters from the current location of the cursor to the end of the current line.
INS LN	\EL	The text line containing the cursor and all text lines below it roll downward one line, a blank line is inserted in the screen row containing the cursor, and the cursor moves to the left margin of the blank line. Holding the key down causes the operation to be repeated until the key is released.
DEL LN	\EM	The text line containing the cursor is deleted from display memory, all text lines below it roll upward one row, and the cursor moves to the left margin. Holding the key down causes the operation to be repeated until the key is released or until there are no subsequent lines of text remaining in display memory. In the latter case, pressing or continuing to hold down this key has no further effect.
DEL CHR	\EP	The cursor remains stationary while the character at the current cursor location is deleted. All characters between the cursor and the right margin move left one column and a blank moves into the line at the right margin. This function is meant to be used within that portion of the screen delineated by the left and right margins. If the cursor is positioned to the left of the left margin, the delete character function works as previously described. If the cursor is positioned beyond the right margin, the delete character function affects those characters from the current cursor position through the right boundary of the screen. If the key is held down, the terminal continues to delete characters until either the key is released or no characters remain between the cursor position and the right margin. In the latter case, pressing or continuing to hold down this key has no further effect.
INS CHR	\EQ	Turn on the insert character mode (see the description that follows).
INS CHR	\ER	Turn off the insert character mode (see the description that follows).

Insert Character Mode

When the “insert character” editing mode is enabled, characters entered through the keyboard or received from the computer are inserted into display memory at the cursor position. Each time a character is inserted, the cursor and all characters from the current cursor position through the right margin move one column to the right. Characters that are forced past the right margin are lost. When the cursor reaches the right margin, it moves to the left margin in the next lower line and the insert character function continues from that point.

The edit function is meant to be used within that portion of the screen delineated by the left and right margins. If the cursor is positioned to the left of the left margin, the insert character function works as previously described. If the cursor is positioned beyond the right margin, however, the insert character function affects those characters between the current cursor position and the right boundary of the screen. In such a case, when the cursor reaches the right boundary of the screen, it moves to the left margin in the next lower line and the insert character function continues from that point as described in the previous paragraph.

When the insert character mode is enabled (and softkey labels are displayed), the characters IC are displayed between the fourth and fifth function key labels. These characters are displayed to remind you that you are in the insert character mode.

Function Key Group

Across the top right of the keyboard are 16 keys labeled (0 16) through (15 31). HP-UX recognizes only the first 8 keys, (0 16) through (7 23), as function keys. The functions performed by these keys change dynamically as you use the terminal. At any given time the applicable function labels for these keys appear across the bottom of the display screen. However, softkeys are not supported by HP-UX (softkeys are those keys physically located on the display).

Modes

When you press the “MODES” key (12 28), the eight function keys are redefined. Pressing a redefined key allows access to one of the “modes” described in the following sections. The labels for the redefined keys are shown below (keys without labels are undefined):

(0 16)	(1 17)	(2 18)	(3 19)	(4 20)	(5 21)	(6 22)	(7 23)
			REMOTE			DISPLAY AUTO	
			MODE			FUNCT	LF*

You may use these function keys to enable and disable various terminal operating modes. Each defined mode selection key alternately enables and disables a particular mode. When the mode is enabled, an asterisk (*) appears in the associated key label on the screen (for example, auto line feed mode is enabled in the key menu above).

When the **remote mode** is enabled and a key is pressed, the terminal transmits the associated ASCII code to HP-UX. In **local mode** (remote mode is disabled), when an alphanumeric key is pressed the associated character is displayed at the current cursor position on the screen (nothing is transmitted to HP-UX).

When the **auto line feed mode** is enabled, an ASCII line feed control code is automatically appended to each ASCII carriage return control code generated through the keyboard. ASCII carriage return control codes can be generated through the keyboard in any of the following ways:

- By pressing either **EXECUTE** or **RETURN** (HP-UX treats these keys identically).
- By simultaneously pressing the keys **CTRL** and **M**.
- By pressing any of the user keys (**0 16** through **7 23**), provided that a carriage-return code is included in the particular key definition.

When the **display functions mode** is enabled, the terminal operates as follows:

- In local mode, it displays ASCII control codes and escape sequences but does not execute them. For example, if you press **←**, the terminal displays `\ED` on the screen but does not move the cursor one character to the left.
- In remote mode, it transmits ASCII control codes and escape sequences but does not execute them locally. For example, if you press **ROLL ↑**, the terminal transmits `\ES` but does not perform the “roll up” function. If local echo is enabled (ON) then the `\ES` is also displayed on the screen. **Local echo** specifies that the character is not only transmitted, but displayed on the terminal as well.

These same mode selection functions can be accessed via the escape sequences:

- `\E&k <x>R` – when x is 0, the remote mode is off; when x is 1, the remote mode is on.
- `\E&k <x>A` – when x is 0, auto line feed mode is off; when x is 1, auto line feed mode is on.
- `\EY` – enables display functions; when enabled, all printing and non-printing characters are displayed.
- `\EZ` – disables display functions; when disabled, only printing characters are displayed.

AIDS

When you press the AIDS key **12 28**, another menu is displayed, showing a single, defined key (the MARGINS/TABS key). When this key is pressed, the eight function keys become general control keys that you use for setting and clearing margins and tabs from the keyboard. Pressing one of the defined keys causes the terminal to issue the appropriate escape sequence for the function selected. These escape sequences and their function are discussed with the Display Control Group, earlier in this section.

Note that the MARGINS and TABS keys only send their associated escape sequences to HP-UX when display functions are enabled and when the A Strap is set (discussed later in this article). If these conditions are not met, the escape sequence is executed locally but is not sent to HP-UX.

User Keys

When you press the USER KEYS key (14 30), the eight function keys display the user defined key labels. In the following section (“User-definable Keys”), the function of the user keys and the procedure for defining them is described. To remove the user key labels from the screen (while still retaining their defined functions), press (12 28) (the AIDS key) while holding the (SHIFT) key depressed.

The USER KEYS key always toggles between displaying the current key labels and the user key labels.

User-definable Keys

The eight function keys (0 16) through (7 23), besides performing the terminal control functions described above, can be defined by a program. In this context, “defined” means:

- You can assign to each key a string of ASCII alphanumeric characters and/or control codes (such as carriage return or line feed).
- You can specify each key’s operation attribute: whether its key definition is to be executed locally at the terminal, transmitted to the computer, or both.
- You can assign to each key an alphanumeric label (up to 16 characters) which, in user keys mode (i.e. when the USER KEYS key (14 30) is pressed), is displayed across the bottom of the screen.

The definition of each user key may contain up to 80 characters (alphanumeric characters, ASCII control characters, and explicit escape sequence characters).

To define a user-definable key, enter the escape sequence:

```
\E&f <attribute><key><label length><string length><label><string>
```

where <attribute> is a two character combination from the list 0a, 1a, or 2a. The default value for <attribute> is 0a. The attribute character specifies whether the definition of the particular user key is to be:

- a. Treated in the same manner as the alphanumeric keys (0a).

If the terminal is in local mode, the definition of the key is executed locally. If the terminal is in remote mode and local echo is disabled (OFF), the definition of the key is transmitted to the computer. If the terminal is in remote mode and local echo is enabled (ON), the definition of the key is both transmitted to the computer and executed locally.

- b. Executed locally only (1a).
- c. Transmitted to the computer only (2a).

When the transmit-only attribute (2a) is designated, the particular user key has no effect unless the terminal is in remote mode. A transmit-only user key appends the appropriate terminator to the string (either carriage-return or carriage-return/line feed, depending on the state of Auto Line Feed).

<key> is a two character identifier specifying the key to be defined. The key is specified by a value in the range 1k through 8k (1k is the default). For example, to specify the fifth user key, enter 5k for <key>. Note that this differs from the physical key labels on the HP 9000 Model 520’s keyboard (they are labeled 0 through 7).

`<label length>` is the number of characters in the key label. Acceptable values are in the range 0d through 16d. Specifying a zero length causes the key label to remain unchanged. 0d is the default value for the label length.

`<string length>` is the length of the string forming the key definition. Acceptable values are in the range -1L through 80L; 1L is the default. Entering a string length value of zero causes the key definition to remain unchanged. A string length value of -1 causes the key definition to be erased.

`<label>` is the character sequence for the label.

`<string>` is the character sequence for the key definition.

The `<attribute>`, `<key>`, `<label length>`, and `<string length>` parameters may appear in any sequence but must precede the label and key definition strings. You must use an uppercase identifier (A, K, D, or L) for the final parameter and a lowercase identifier (a, k, d, or l) for all preceding parameters. If any of the four fields are omitted, their default values are used. At least one of the parameters must be specified because its uppercase identifier is needed to terminate the sequence.

Following the parameters, the first 0 through 16 characters, as designated by `<label length>`, constitute the key's label and the next 0 through 80 characters, as designated by `<string length>`, constitute the key's definition string. The total number of characters (alphanumeric data, ASCII control codes such as carriage-return and line feed, and explicit escape sequence characters) in the label string can exceed 16, but only the first 16 characters are used. The same is true for the destination string; only the first 80 characters are used.

The initial (power-on) definition of the user keys is:

- all keys are transmit-only (attribute is 2a).
- the user key labels are f1 through f8.
- definitions are \Ep, \Eq, \Er, \Es, \Et, \Eu, \Ev, and \Ew for keys 0 through 7 23, respectively.

The Display

The “terminal’s” display has many features of its own, such as video highlights (inverse video and blinking), raster control, cursor sensing and addressing, and color highlight control (for Model 520 Computers equipped with a color display). These functions are accessed only through escape sequences and are discussed in the sections that follow.

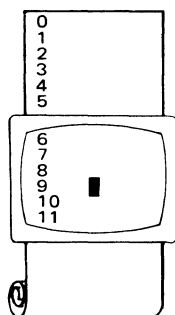
Memory Addressing Scheme

Display memory positions can be addressed using absolute or relative coordinate values. Display memory is made up of 80 columns (0 - 79) and any number of 24 line pages (specified by the HP-UX configuration). As shipped to you, the display memory has 48 lines (0 - 47) of 80 characters (2 screens). The amount of display memory can be determined from byte 0 of the primary terminal status (discussed in the section entitled “The Terminal”, later in this article). The types of addressing available are absolute (memory relative), screen relative, and cursor relative.

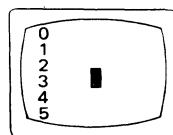
Row Addressing

The figure below illustrates the way that the three types of addressing affect row or line numbers. The cursor is shown positioned in the fourth row on the screen. Screen row 0 is currently at row 6 of display memory. In order to reposition the cursor to the first line of the screen the following three destination rows could be used:

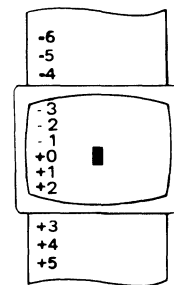
- Absolute: row 6
- Screen Relative: row 0
- Cursor Relative: row -3



a.) Absolute: row 6



b.) Screen Relative: row 0



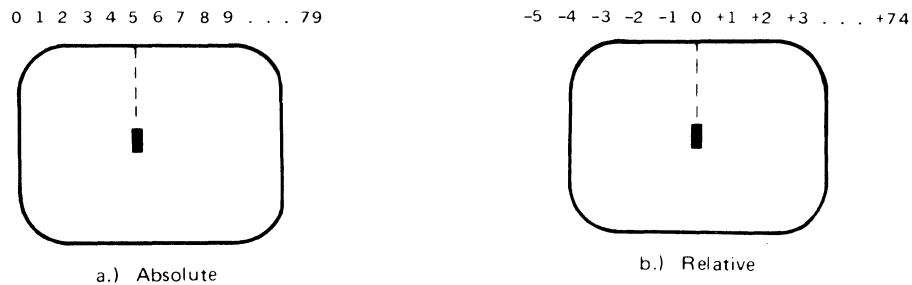
c.) Cursor Relative: row -3

Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen and cursor relative column addressing. The figure below illustrates the difference between absolute and relative addressing. The cursor is shown in column 5.

Whenever the row or column addresses exceed those available, the largest possible value is substituted. In screen relative addressing, the cursor cannot be moved to a row position that is not currently displayed. For example, in the cursor relative portion of the figure above (showing row addressing), a relative row address of -10 would cause the cursor to be positioned at the top of the current screen (relative to row -3). Column positions are limited to the available screen positions. For example, in the following illustration, the absolute column addressing example shows limits of 0 and 79, while the relative column addressing example shows limits of -5 and 74. The cursor cannot be wrapped around from column 0 to column 79 by specifying large negative values for relative column positions.



Column Addressing

Cursor Sensing

The current position of the screen cursor can be sensed. The position returned can be the absolute position in the display memory or the location relative to the current screen position. (Absolute and relative addresses are discussed in the section “Cursor Addressing”.)

Cursor sensing is available only when the “terminal” is in remote mode.

Absolute Sensing

When a program sends the escape sequence `\Ea` to the terminal, the terminal returns to the program an escape sequence of the form `\E&a xxxc yyyR`, where `xxx` is the absolute column number and `yyy` is the absolute row number of the current cursor position. You will later see that this escape sequence is identical to the escape sequence for an absolute move of the cursor.

Relative Sensing

When a program sends the escape sequence `\E'`, the terminal returns to the program an escape sequence of the form `\E&a xxxcyyyY` where `xxx` is the column number of the cursor and `yyy` is row position of the cursor relative to screen row 0. This escape sequence is identical to the escape sequence for a relative move of the cursor (discussed later in this article).

Cursor Positioning

The cursor can be positioned directly by giving memory or screen coordinates, or by sending the escape codes for any of the keyboard cursor positioning operations.

Screen Relative Addressing

To move the cursor to any character position on the screen, use any of the following escape sequences:

```
\E&a<column number> c <row number>Y
\E&a<row number> y <column number>C
\E&a<column number>C
\E&a<row number>Y
```

where <column number> is a decimal number specifying the screen column to which you wish to move the cursor. Zero specifies the leftmost column.

<row number> is a decimal number specifying the screen row (0 - 23) to which you wish to move the cursor. Zero specifies the top row of the screen; 23 specifies the bottom row.

When using the escape sequences for screen relative addressing, the data on the screen is not affected (the cursor may only be moved around in the 24 rows and 80 columns currently displayed, thus data is not scrolled up or down).

If you specify only <column number>, the cursor remains in the current row. Similarly, if you specify only <row number>, the cursor remains in the current column.

Example

The following escape sequence moves the cursor to the 20th column of the 7th row on the screen:

```
\E&a6y19C
```

Absolute Addressing

You can specify the location of any character within display memory by supplying absolute row and column coordinates. To move the cursor to another character position using absolute addressing, use any of the following escape sequences:

```
\E&a<column number> c <row number>R
\E&a<row number> r <column number>C
\E&a<column number>C
\E&a<row number>R
```

where <column number> is a decimal number (0 - 79) specifying the column coordinate (within display memory) of the character at which you want the cursor positioned. Zero specifies the first (leftmost) column in display memory, 79 the rightmost column.

<row number> is a decimal number (0-max) specifying the row coordinate (within display memory) of the character at which you want the cursor positioned. Zero specifies the first (top) row in display memory, max specifies the last. The value of max is specified as:

$$[24 (\text{lines/page}) \times \text{num_page} (\text{pages})] - 1$$

where num_page is the number of pages of display memory specified by the system configuration. As shipped to you, the configuration dictates that 2 pages of display memory be allocated. Thus, the last row that can be addressed is 47.

When using the above escape sequences, the data visible on the screen rolls up or down (if necessary) in order to position the cursor at the specified data character. The cursor and data movements occur as follows:

- If a specified character position lies within the boundaries of the screen, the cursor moves to that position; the data on the screen does not move.
- If the absolute row coordinate is less than that of the top line currently visible on the screen, the cursor moves to the specified column in the top row of the screen; the data then rolls down until the specified row appears in the top line of the screen.
- If the absolute row coordinate exceeds that of the bottom line currently visible on the screen, the cursor moves to the specified column in the bottom row of the screen; the data then rolls up until the specified row appears in the bottom line of the screen.

If you specify only a <column number>, the cursor remains in the current row. Similarly, if you specify only a <row number>, the cursor remains in the current column.

Example

To position the cursor (rolling the data if necessary) at the character residing in the 60th column of the 27th row in display memory, the escape sequence is:

```
\E&a26r59C
```

Cursor Relative Addressing

You can specify the location of any character within display memory by supplying row and column coordinates that are relative to the current cursor position. To move the cursor to another character position using cursor relative addressing, use any of the following escape sequences:

```
\E&a +/- <column number> c +/- <row number>R
\E&a +/- <row number> r +/- <column number>C
\E&a +/- <column number>C
\E&a +/- <row number>R
```

where <column number> is a decimal number specifying the relative column to which you wish to move the cursor. A positive number specifies how many columns to the right you wish to move the cursor; a negative number specifies how many columns to the left.

<row number> is a decimal number specifying the relative row to which you wish to move the cursor. A positive number specifies how many rows to the right you wish to move the cursor; a negative number specifies how many rows to the left.

When using the above escape sequences, the data visible on the screen rolls up or down (if necessary) in order to position the cursor at the specified data character. The cursor and data movements occur as follows:

- If a specified character position lies within the boundaries of the screen, the cursor moves to that position; the data on the screen does not move.
- If the specified cursor relative row precedes the top line currently visible on the screen, the cursor moves to the specified column in the top row of the screen; the data then rolls down until the specified row appears in the top line of the screen.
- If the specified cursor relative row precedes the bottom line currently visible on the screen, the cursor moves to the specified column in the bottom row of the screen; the data then rolls up until the specified row appears in the bottom line of the screen.

If you specify only a <column number>, the cursor remains in the current row. Similarly, if you specify only a <row number>, the cursor remains in the current column.

Example

To position the cursor (rolling the data if necessary) at the character residing 15 columns to the right and 25 rows above the current cursor position (within display memory), use the escape sequence:

```
\E&a+15c-25R
```

Combining Absolute and Relative Addressing

You may use a combination of screen relative, absolute and cursor relative addressing within a single escape sequence.

For example, to move the cursor (and roll the text if necessary) so that it is positioned at the character residing in the 70th column of the 18th row below the current cursor position, use the escape sequence:

```
\E&a69c+18R
```

Similarly, to move the cursor (and roll the text up or down if necessary) so that it is positioned at the character residing in the 10th column of absolute row 48 in display memory, use the escape sequence:

```
\E&a9c47R
```

Display Enhancements

The terminal includes as a standard feature the following display enhancement capabilities:

- Inverse Video - black characters are displayed against a white background.
- Underline Video - characters are underscored.
- Blink Video - characters blink on and off.

Note

The half bright display enhancement is not implemented on this terminal. When the half bright enhancement is selected on the HP 9000 Model 520 with a black-and-white display, it is ignored. Selecting the half bright enhancement on the HP 9000 Model 520 with a color display causes the terminal to select pen 3. (See the section "Accessing Color" later in this article).

The display enhancements are used on a field basis. The field cannot span more than one line. The field scrolls with display memory. Overwriting a displayable character in a field preserves the display enhancement. The enhancements may be used separately or in any combination. When used, they cause control bits to be set within display memory.

From a program or from the keyboard, you enable and disable the various video enhancements by embedding escape sequences within the data. The general form of the escape sequence is:

```
\E&d<enhancement code>
```

where enhancement code is one of the uppercase letters A through O specifying the desired enhancement(s) or an @ to specify end of enhancement:

Enhancement Character

	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Half-Bright									x	x	x	x	x	x	x	x
Underline					x	x	x	x					x	x	x	x
Inverse Video			x	x			x	x			x	x			x	x
Blinking		x		x		x		x		x		x		x		x
End Enhancement	x															

Note that the escape sequence for “end enhancement” (`\E&d@`) or the escape sequence for another video enhancement, ends the previous enhancement.

Raster Control

The terminal provides the ability to enable and disable both its alphanumeric display and its graphic display. The escape sequences for these capabilities are:

- `\E*dc` - Turn on graphics display; enable writing to the graphics display.
- `\E*dd` - Turn off graphics display; disable writing to the graphics display.
- `\E*de` - Turn on the alphanumeric display; enable writing to the alphanumeric display.
- `\E*df` - Turn off the alphanumeric display; disable writing to the alphanumeric display.

Whether used individually or in combination, the last character of the escape sequence must be uppercase. For example, to turn off the graphics display, use the escape sequence `\E*dD`. When these sequences are combined, an uppercase specifier must terminate the sequence. To turn on the graphics display and turns off the alphanumeric display, use the escape sequence `\E*dcF`.

Accessing Color

If your Model 520 computer is equipped with a color display, you may access its color capabilities from HP-UX. First, you need to understand some simple terms.

Color pair - two colors which define the foreground color (color of the characters) and the background color, respectively. At least one color of the color pair must be black; displaying color on color is not possible. A total of 15 color pairs are possible, but only eight can be displayed at any one time.

Pen # - one of eight predefined color pairs. Pen 0 through pen 7 are initially defined as follows (re-defining a color pair is described later):

Pen #	foreground color	background color
0	white	black
1	red	black
2	green	black
3	yellow	black
4	blue	black
5	magenta	black
6	cyan	black
7	black	yellow

Pen #0 is the default pen selected by the terminal when writing to the display.

Pen #7 is always used for displaying the softkey labels.

Selecting a Pen (Color Pair)

By using an escape sequence, you can select a pen number other than pen #0 when writing to the display. Like other display enhancements, pen selection is used on a field basis. The field cannot span more than one line. That is, the pen selection is only active until a new-line character is encountered; then the default pen is re-selected. The escape sequence for selecting a pen is:

```
\E&v n<parameter>
```

where *n* is the pen number you wish to use, and *<parameter>* is a single character that specifies what action you want to take. To select a pre-defined pen number, the necessary *<parameter>* is *s*. If *n > 7*, HP-UX performs the calculation (*n* Modulo 8) on the supplied value to determine the actual pen number. Thus,

```
\E&v 4S
```

selects the pre-defined pen number 4. Note that *s* is capitalized in the preceding escape sequence. This is because escape sequences are terminated by a capital letter. Thus, the last character of any escape sequence *must* be uppercase. However, if a parameter is *not* the last character of the escape sequence, it may appear in lower-case.

Changing Pen Definitions

You may change the pre-defined color pair for any of the eight existing display pens. The three primary colors (red, green and blue) are used in various combinations to achieve the desired color.

The combinations of red, green, and blue that define foreground and background colors can be specified in two notations. The first is RGB (Red-Green-Blue), and the second is HSL (Hue-Saturation-Luminosity). The notation must be selected before you can redefine pens (if no notation type is specified, the “terminal” uses the last notation specified, or RGB notation at power-up). To select a notation type, use the `\E&v` escape sequence used above:

```
\E&v n<parameter>
```

where *n* is 0 (for RGB) or 1 (for HSL), and *<parameter>* is the letter *m*. Thus, the sequence

```
\E&v 1M
```

selects HSL notation. It does nothing more.

To specify the quantity of red (hue), green (saturation), and blue (luminosity) to appear in your background and foreground colors, the a, b, c, x, y, and z parameters are used. These parameters have the following meanings:

- a specifies the amount of red (hue) used in the foreground.
- b specifies the amount of green (saturation) used in the foreground.
- c specifies the amount of blue (luminosity) used in the foreground.
- x specifies the amount of red (hue) used in the background.
- y specifies the amount of green (saturation) used in the background.
- z specifies the amount of blue (luminosity) used in the background.

Each a, b, c, x, y, and z parameter specified is preceded by a number in the range 0 through 1, in increments of 0.01. The following table gives the values needed to define the eight principle colors:

R	G	B	Color	H	S	L
0	0	0	Black	X	X	0
0	0	1	Blue	0.66	1	1
0	1	0	Green	0.33	1	1
0	1	1	Cyan	0.50	1	1
1	0	0	Red	1	1	1
1	0	1	Magenta	0.83	1	1
1	1	0	Yellow	0.16	1	1
1	1	1	White	X	0	1

(Note that X's in the above table represent "don't care" situations.)

One final parameter, i, is needed. It is used to assign a pen number to the newly-defined color pair. Thus, the escape sequence for changing a color pair definition is:

```
\E&v <0|1>m na nb nc nx ny nz <pen#>I
```

where either a 0 or a 1 precedes the **m** parameter (selecting either RGB or HSL notation, respectively), and n is one of the legal values from the table above. <pen#> is an integer in the range 0 - 7 which, when combined with the i parameter, defines that pen number to be the color pair specified by the preceding a, b, c, x, y, and z parameters. Omitting any a, b, c, x, y, or z parameter causes a value of 0 to be assigned to the omitted parameter by default.

Examples

```
\E&v 0m 1a 0b 0c 0x 1y 0z 5l
```

This example re-defines pen 5 to specify red characters on a green background. (Note that the Model 520 will ignore the green background specification and assign a black one instead.) This example is equivalent to

```
\E&v 0m 1a 1y 5l
```

since omitted parameters (a, b, c, x, y, z) are given default values of 0.

```
\E&v 1m .66a 1b 1c 3i 0m 1c 1x 1y 6l
```

This example re-defines pen 3 to specify blue characters on a black background (HSL notation), and pen 6 to specify blue characters on a yellow background (RGB notation). This example illustrates how multiple pens can be defined on a single line using different notations. (Again, note that the Model 520 will reject the background specification of pen 6, and will use black instead.)

```
\E&v 0m 1y 1z 1a 1c 4l
```

This example re-defines pen 4 to specify a cyan background with magenta characters. This example shows how background and foreground specifications can be reversed. The Model 520 will accept the magenta foreground, but will reject the cyan background; black will be used instead.

If the foreground and background colors are both non-black, the foreground color will be used, and the background color will be black, regardless of the order in which the parameters are specified.

```
\E&v 5l
```

This example re-defines pen 5 to specify a black foreground and a black background, using the previous notation type.

Note

Supplying neither a foreground nor a background color when defining a color pair causes both the foreground and background to be black. This is like typing on a typewriter without paper or ribbon; you can't see what is written.

Controlling Configuration and Status

The terminal provides additional escape sequences for managing its configuration and its status.

Re-configuring the Terminal

The terminal allows you to reset a few of its configuration parameters via escape sequences. These parameters and their escape sequences are:

Function	Escape Sequence	Description
Auto Line Feed Mode	\E&k nA	When n is 0, auto line feed mode is off. When n is 1, auto line feed mode is on. Default = OFF.
Local Echo	\E&k nL	Characters entered through the keyboard are displayed on the screen and transmitted to the computer when n = 1. When n = 0, characters entered through the keyboard are transmitted to the computer only; if they are to appear on the screen, the computer must "echo" them back to the terminal. Default = OFF.
Remote Mode	\E&k nR	When n is 0, the remote mode is off. When n is 1, the remote mode is on. Default = ON.
Caps Mode	\E&k nP	When caps mode is enabled, all unshifted alphabetic keys generate uppercase letters and all shifted alphabetic keys generate lowercase letters. This mode is used primarily as a typing convenience and affects only the 26 alphabetic keys. When n = 1, the caps mode is enabled. When n = 0, the caps mode is disabled. Default = OFF. From the keyboard, you enable and disable caps mode using the ⓈCAPS key. This key alternately enables and disables caps mode.
Transmit Function (STRAP A)	\E&s <x>A	This escape sequence specifies whether or not escape code sequences are both executed at the terminal and transmitted to HP-UX. When x = 1, the escape code sequences generated by control keys such as ⓈROLL (up) and ⓈROLL (down) are transmitted to HP-UX. If local echo is ON, the function is also performed locally. When x = 0, the escape sequences for the major function keys are executed locally, but are not transmitted to HP-UX. The default is x = 0.
Enable End Of Line Wrap (STRAP C)	\E&s <x>C	This field specifies whether or not the end-of-line wrap is inhibited. When x = 0 and the cursor reaches the right margin, it automatically moves to the left margin in the next lower line (a local carriage return and line feed are generated). When x = 1 and the cursor reaches the right margin, it remains in that screen column until an explicit carriage return or other cursor movement function is performed (succeeding characters overwrite the existing character in that screen column). Default = OFF.

Sending Terminal Status

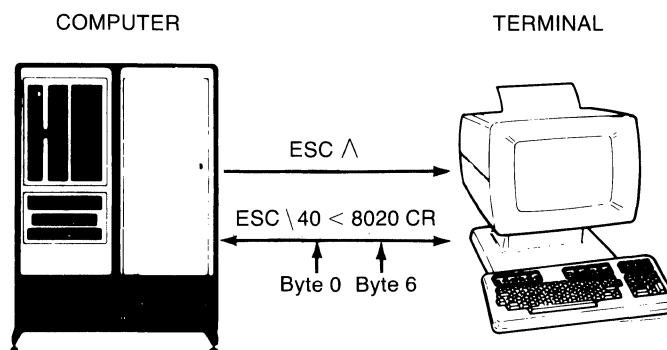
Terminal status is made up of 14 status bytes (bytes 0 through 13) containing information such as display memory size, switch settings, configuration menu settings, and terminal errors. There are two terminal status requests: primary and secondary. Each returns a set of seven status bytes.

Primary Terminal Status

You can request the first set of terminal status bytes (bytes 0 through 6) by issuing the following escape sequence:

`\E^`

The terminal responds with an `\E\`, and seven status bytes followed by a terminator (a carriage-return character). A typical primary terminal status request and response is shown in the following illustration.

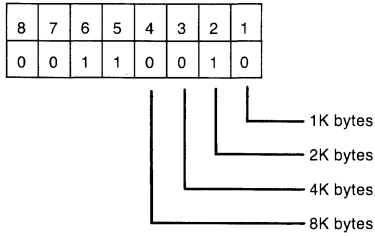


BYTE	ASCII	BINARY	STATUS
0	4	0011 0100	4K bytes of display memory (2 pages of 24 lines)
1	0	0011 0000	Function key transmission disabled
			Cursor wraparound disabled
2	<	0011 1100	Configuration Straps A-H
3	8	0011 1000	
			Terminal sends secondary status
4	0	0011 0000	
5	2	0011 0010	Last Self-Test ok
6	0	0011 0000	

Primary Terminal Status Example

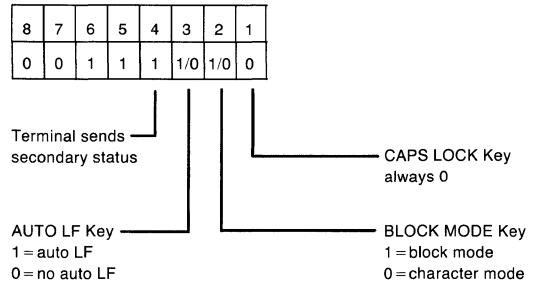
PRIMARY STATUS BYTES

BYTE 0 DISPLAY MEMORY SIZE

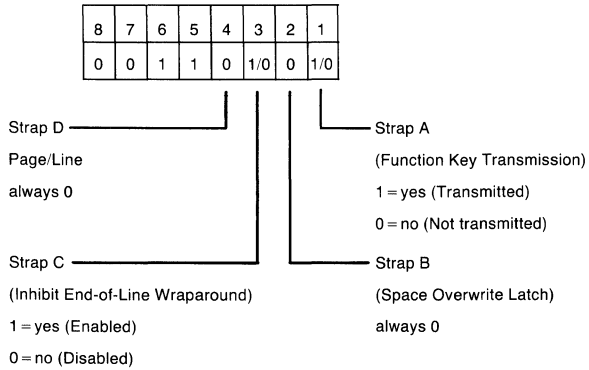


This byte specifies the amount of display memory available in the terminal.
(roughly 2K per page)

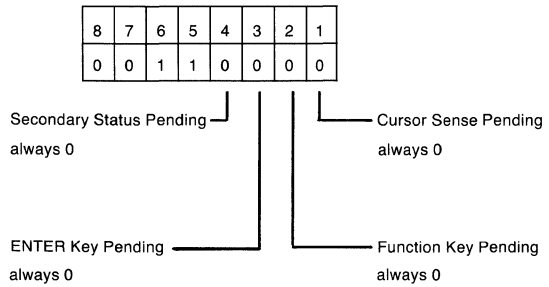
BYTE 3 LATCHING KEYS



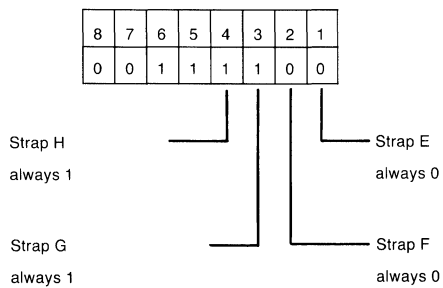
BYTE 1 CONFIGURATION STRAPS A-D



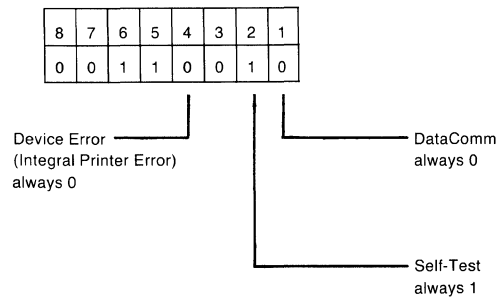
BYTE 4 TRANSFER PENDING FLAGS



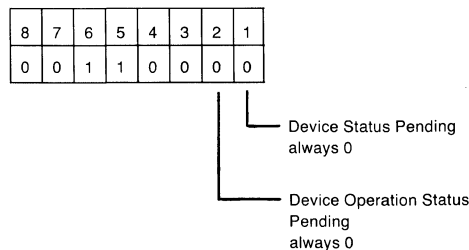
BYTE 2 CONFIGURATION STRAPS E-H



BYTE 5 ERROR FLAGS



BYTE 6 DEVICE TRANSFER PENDING FLAGS



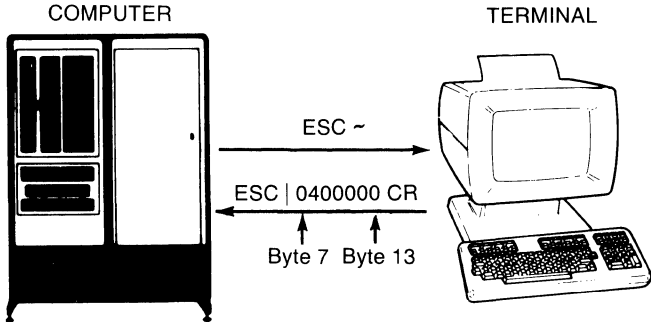
(tracks "S", "F", or "U" completion codes associated with E&p device control sequences.)

Secondary Terminal Status

You can request the second set of terminal status bytes (bytes 7 through 13) by issuing the following escape sequence:

```
\E~
```

The terminal responds with an \E|, and seven status bytes followed by a terminator (a carriage-return character). A typical secondary terminal status request and response is shown in the following illustration.



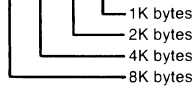
BYTE	ASCII	BINARY	STATUS
7	0	0011 0000	
8	4	0011 0101	Terminal identifies self
9	0	0011 0000	
10	0	0011 0000	
11	0	0011 0000	
12	0	0011 0000	
13	0	0011 0000	

Secondary Terminal Status Example

Secondary Status Bytes

BYTE 7 Buffer Memory (always zero)

8	7	6	5	4	3	2	1
0	0	1	1	0	0	1	0



Memory installed in addition to display memory that is available for use as data buffers. Note that the HP 9000 Model 20 terminals always return a 0 value.

BYTE 8 TERMINAL FIRMWARE CONFIGURATION

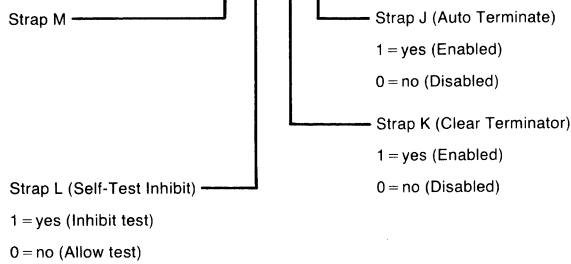
8	7	6	5	4	3	2	1
0	0	1	1	0	1	0	0



APL Firmware does not apply.

BYTE 9 CONFIGURATION STRAPS J-M (always zero)

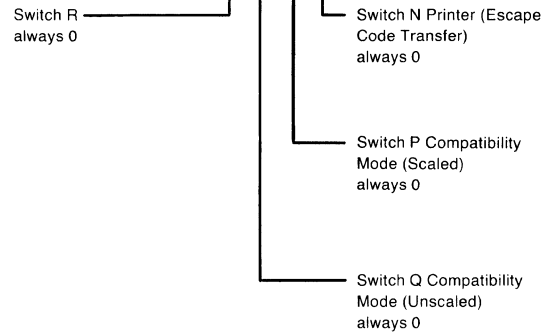
8	7	6	5	4	3	2	1
0	0	1	1	0	0	0	0



Straps J-M do not apply to the terminal.

BYTE 10 KEYBOARD INTERFACE KEYS (N-R)

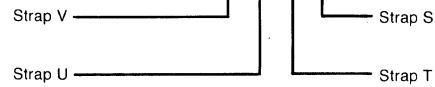
8	7	6	5	4	3	2	1
0	0	1	1	0	0	0	0



Straps N-R do not apply

BYTE 11 CONFIGURATION STRAPS S-V (always zero)

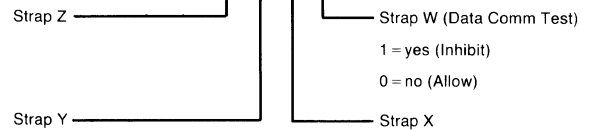
8	7	6	5	4	3	2	1
0	0	1	1	0	0	0	0



Straps S-V do not apply to the terminal.

BYTE 12 CONFIGURATION STRAPS W-Z (always zero)

8	7	6	5	4	3	2	1
0	0	1	1	0	0	1	0



Straps X, X, Y, and Z do not apply to the terminal.

BYTE 13 MEMORY LOCK MODE (always zero)

8	7	6	5	4	3	2	1
0	0	1	1	0	0	0	0

Table of Contents

Mail: Mail Handler

Introduction	1
Common Usage	2
Maintaining Folders	8
More About Sending Mail	9
Tilde Escapes	9
Network Access	12
Special Recipients	13
Additional Features	14
Message Lists	14
List of Commands	15
Custom Options	21
Command Line Options	23
Message Format	24
Glossary	25
Summary of Commands, Options, and Escapes	26
Command Summary	26
Options Summary	27
Tilde Escapes Summary	28
Command Line Flags	28

Mailx Mail Handler

Introduction

Mailx provides a simple and friendly environment for sending and receiving mail. It divides incoming mail into its constituent messages and allows the user to deal with them in any order. In addition, it provides a set of *ed*-like commands for manipulating messages and sending mail. *Mailx* offers the user simple editing capabilities to ease the composition of outgoing messages, as well as providing the ability to define and send to names which address groups of users. Finally, *mailx* is able to send and receive messages across HP-UX-supported networks such as UUCP.

This document describes how to use the *mailx* program to send and receive messages. You do not need to be familiar with other message handling systems, but you should be familiar with the HP-UX shell, the text editor, and some of the common HP-UX commands. The *HP-UX Reference* and *HP-UX Concepts and Tutorials* manuals covering text editors and *csh* can be consulted for more information on these topics.

Here is how messages are handled: the mail system accepts incoming *messages* for you from other people and collects them in a file, called your *system mailbox*. When you login, the system notifies you if there are any messages waiting in your system mailbox. If you are a *csh* user, you will be notified when new mail arrives if you inform the shell of the location of your mailbox. Your system mailbox is located in the directory `/usr/mail` in a file with your login name. For example, if your login name is *sam*, you can make *csh* notify you of new mail by including the following line in your `.cshrc` file:

```
set mail=/usr/mail/sam
```

When you read your mail using *mailx* it reads your system mailbox and separates that file into the individual messages that have been sent to you. You can then read, reply to, delete, or save these messages. Each message is marked with its author and the date sent.

Common Usage

The *mailx* command has two distinct usages, depending on whether you want to send or receive mail. Sending mail is simple: to send a message to a user whose login name is, say, *root*, use the shell command:

```
% mailx root
```

then type your message. When you reach the end of the message, type an EOT (**CTRL**-**D**) at the beginning of a line, causing *mailx* to echo an EOT and return you to the Shell. The next time the person you sent mail to next logs in, he will receive the message:

```
You have mail.
```

indicating the availability of your message.

If, while you are composing the message you decide that you do not wish to send it after all, you can abort the letter by pressing **DEL** (or **RUN**). Pressing **RUN** once causes *mailx* to print (or display):

```
(Interrupt -- one more to kill letter)
```

Pressing **DEL** a second time causes *mailx* to save your partial letter on the file *dead.letter* in your home directory and abort the letter. Once you have sent mail to someone, there is no way to undo the act, so be careful.

The message your recipient reads will consist of the message you typed, preceded by one or more lines telling who sent the message (your login name), the date and time it was sent, and other information about the letter.

If you want to send the same message to several other people, you can list their login names on the command line. Thus,

```
% Mail sam bob john
Tuition fees are due next Friday. Don't forget!!
Control\~d>
EOT
%
```

sends the reminder to sam, bob, and john.

If, when you log in, you see the message,

```
You have mail.
```

you can read the mail by typing:

```
% mailx
```

Mailx responds by typing (displaying) its version number and date, then listing the messages you have waiting. It then sends a prompt and awaits your next command. Messages are assigned numbers starting with 1--, and each message is accessed by using the assigned message number.

Mailx keeps track of which messages are **new** (have been sent since you last read your mail) and **read** (have been read by you). New messages have an N next to them in the header listing, while old, but unread, messages have a U next to them. *mailx* keeps track of new/old and read/unread messages by putting a header field called `Status` into your messages. To look at a specific message, use the *type* command (abbreviated *t*). For example, if you had the following messages:

```
N 1 root      Wed Sep 21 09:21 "Tuition fees"
N 2 sam      Tue Sep 20 22:55
```

you could examine the first message by giving the command:

```
type 1
```

causing *mailx* to respond with, for example:

```
Message 1:
>From root Wed Sep 21 09:21:45 1978
Subject: Tuition fees
Status: R
Tuition fees are due next Wednesday. Don't forget!!
```

Many *mailx* commands such as *type* that operate on messages take a message number as an argument. For these commands, there is a notion of a current message. When you enter the *mailx* program, the current message is initially the first one. Thus, you can often omit the message number and use, for example,

```
t
```

to type (display) the current message. As a further shorthand, you can type a message by simply giving its message number. Hence,

```
1
```

would display the first message.

Frequently, it is useful to read the messages in your mailbox in order, one after another. You can read the next message in *mailx* by simply typing a newline (press **RETURN**). As a special case, you can type a newline (**RETURN**) as your first command to *mailx* to type (display) the first message.

After the message has been typed or displayed, if you wish to send an immediate reply, you can do so with the *reply* command. *Reply*, like *type*, takes a message number as its argument. *Mailx* then begins a message addressed to the user who sent you the message. You can then type in your letter of reply, followed by a **CTRL-D** (EOT) at the beginning of a line, as before. *Mailx* then sends EOT followed by the ampersand prompt to indicate it is ready for another command. In our example, if, after reading the first message, you wished to reply to it, you might give the command:

```
reply
```


4 Mailx

Mailx responds with:

```
To: root
Subject: Re: Tuition fees
```

and waits for you to enter your letter. You are now in the message-collection mode described at the beginning of this section so *mailx* gathers your message up to an EOT (**CTRL**-**D**).

Note that *mailx* copies the subject header from the original message because correspondence about a particular matter tends to retain the same subject heading, making it easy to recognize. If there are other header fields in the message, that information is also used. For example, if the letter had a `To:` header listing several recipients, *mailx* would arrange to send your reply to the each of them. Similarly, if the original message contained a `Cc:` (carbon copies to) field, *mailx* would send your reply to **those** users. However, *mailx* does not send the message to you, even if you appear in the `To:` or `Cc:` field, unless you explicitly ask to be included. See Tilde Escapes and Special Recipients sections of this article for more details.

After typing in your letter, the dialog with *mailx* might look like the following:

```
reply
To: root
Subject: Re: Tuition fees
Thanks for the reminder
EOT
&
```

The *reply* command is especially useful for sustaining extended conversations over the message system, with other “listening” users receiving copies of the conversation. The *reply* command can be abbreviated to *r*.

Sometimes you will receive a message that has been sent to several people and wish to reply **only** to the person who sent it. *Reply* with an uppercase *R* replies to a message, but sends a copy to the sender only.

If, while reading your mail, you wish to send a message to someone, but not as a reply to one of your messages, you can send the message directly using the *mail* command, which takes as arguments the names of the recipients you want to send to. For example, to send a message to “frank”,

```
mail frank
This is to confirm our meeting next Friday at 4.
EOT
&
```

The *mail* command can be abbreviated to *m*.

Normally, each message you receive is saved in the file *mbox* in your login directory at the time you leave *mailx*. Often, however, you will not want to save a particular message you have received because it is only of passing interest. To avoid saving a message in *mbox*, delete it using the *delete* command. In our example,

```
delete 1
```

prevents *mailx* from saving message 1 (from root) in *mbox*. In addition to not saving deleted messages, *mailx* does not let you type (display) them either. The effect is to make the message disappear altogether, along with its number. The *delete* command can be abbreviated to *d*. Many features of *mail* can be tailored to your liking with the *set* command. *Set* has two forms, depending on whether you are setting a **binary** or **valued** option. Binary options are either on or off. For example, the *ask* option informs *mailx* that each time you send a message, you want it to prompt you for a subject header, to be included in the message. To set the *ask* option, type

```
set ask
```

Another useful *mailx* option is *hold*. Unless told otherwise, *mailx* moves the messages from your system mailbox to the file *mbox* in your home directory when you leave *mailx*. If you want *mailx* to keep your letters in the system mailbox instead, set the *hold* option:

```
set hold
```

Valued options tailor *mailx* to match your needs. For example, the *shell* option tells *mailx* which shell you like to use. For example to select the shell */bin/csh*, type:

```
set SHELL=/bin/csh
```

Note that no spaces are allowed in *SHELL=/bin/csh*. A complete list of the *mailx* options appears at the end of this article.

Another important valued option is *crt*. If you use a fast video terminal to print long messages, they fly by too quickly for you to read them. You can use the *crt* option to force *mailx* to send messages longer than a given number of lines through the paging program *more*. For most CRT displays, use the following command:

```
set crt=24
```

to paginate messages that will not fit on a 25-line screen. *More* prints a screenful of information, then displays *--MORE--* on the remaining line. Type a space to see the next screenful.

Mailx also provides an *alias* option where the specified alias is a name that stands for one or more real user names. Mail sent to an alias is then sent to the list of real users associated with the alias. For example, an alias can be defined for the members of a project, so that you can send mail to the whole project by sending mail to just a single name. The *alias* command in *mailx* defines an alias. Suppose that the users in a project are named Sam, Sally, Steve, and Susan. To define an alias called *project* for them, use:

```
alias project sam sally steve susan
```

6 Mailx

Alias can also be used to provide a convenient name for someone whose user name is inconvenient. For example, if a user named “Bob Anderson” had the login name “anderson”, you might want to use:

```
alias bob anderson
```

so that you could send mail to the shorter name, “bob”.

While *alias* and *set* commands enable you to customize *mailx*, they must be retyped each time you enter *mailx*. To make them more convenient to use, *mailx* always looks for two files when it is invoked. It first reads a system-wide file `/usr/lib/mailx/mailx.rc`, then a user-specific file, `.mailrc` which is found in the user’s home directory. The system-wide file is maintained by the system administrator and contains *set* commands that are applicable to all system users. The `.mailrc` file is usually set up by each user to select options to fit his preference and to define individual aliases. Here is an example `.mailrc` file:

```
set ask nosave SHELL=/bin/csh
```

As you can see, it is possible to set many options in the same *set* command. The *nosave* option is described in the Additional Features section of this article.

Mail aliasing is implemented at the system-wide level by the mail delivery system *sendmail*. These aliases are stored in the file `/usr/lib/aliases` and are accessible to all system users. The lines in `/usr/lib/aliases` are of the form:

```
alias: <alias>, <name 1>, <name 2>, <name 3>, ...
```

where `<alias>` is the mailing list name and the `<names>` are the members of the list. Long lists can be continued onto the next line by starting the next line with a space or tab. Remember that you must execute the shell command *newaliases* after editing `/usr/lib/aliases` because the delivery system uses an indexed file created by *newaliases*.

Note

Mailx supports *alias* **only** for mail originators on the system as defined here. System-wide aliasing requires the *sendmail* facility which is not presently available on HP-UX.

We have seen that *mailx* can be invoked with command line arguments (people to send the message to), or with no arguments (to read mail). Specifying the `-f` flag on the command line causes *mailx* to read messages from a file other than your system mailbox. For example, if you have a collection of messages in the file *letters* you can use *mailx* to read them with:

```
% mailx -f letters
```

You can use all the *mailx* commands described in this article to examine, modify, or delete messages from your *letters* file which will be rewritten when you leave *mailx* with the *quit* command described below.

Since mail that you read is saved in the file *mbox* in your home directory by default, you can read the file from your home directory by typing

```
% mailx -f
```

Normally, messages that you examine using the *type* command are saved in *mbox* in your home directory if you leave *mailx* with the *quit* command described below. If you wish to retain a message in your system mailbox you can use the *preserve* command to tell *mailx* to leave it there. *Preserve* accepts a list of message numbers, just like *type* and can be abbreviated to *pre*.

Messages in your system mailbox that you do not examine are normally retained in your system mailbox automatically. To save such a message saved in *mbox* without reading it, use the *mbox* command. For example,

```
mbox 2
```

in our example would cause the second message (from sam) to be saved in *mbox* when the *quit* command is executed. *Mbox* can also be used to direct messages to your *mbox* file if you have set the *hold* option described previously. *Mbox* can be abbreviated to *mb*.

When you have perused all messages of interest, use the *quit* command to leave *mailx*. Any messages you have typed but not deleted are saved in the file *mbox* in your login directory. Deleted messages are discarded irretrievably, and messages left untouched are preserved in your system mailbox so that you will see them the next time you type:

```
% mailx
```

Quit can be abbreviated to *q*.

If, for some reason, you want to leave *mailx* quickly without altering either your system mailbox or *mbox*, type *x* (short for *exit*), which immediately returns you to the Shell without changing anything.

If, instead, you want to execute a Shell command without leaving *mailx*, type the command preceded by an exclamation point, just as in the text editor. For instance:

```
!date
```

prints the current date without leaving *mail*.

The *help* command prints out a brief summary of the *mailx* commands, using only single-character command abbreviations.

Maintaining Folders

This section describes a simple *mailx* facility for maintaining groups of messages together in folders.

To use the folder facility, you must tell *mailx* where you want to keep your folders. Each folder of messages will be a single file. For convenience, all of your folders are kept in a single directory of your choosing. To tell *mailx* where your folder directory is, put a line of the form

```
set folder=letters
```

in your *.mailrc* file. If, as in the example above, your folder directory does not begin with a “/”, *mailx* assumes that your folder directory is to be found starting from your home directory. Thus, if your home directory is */usr/person* the above example told *mailx* to find your folder directory in */usr/person/letters*.

Anywhere a file name is expected, you can use a folder name, preceded by a “+” with no intervening spaces. For example, to put a message into a folder with the *save* command, use:

```
save +classwork
```

to save the current message in the *classwork* folder. If the *classwork* folder does not yet exist, it will be created. Note that messages saved by use of the *save* command are automatically removed from your system mailbox.

In order to make a copy of a message in a folder without causing that message to be removed from your system mailbox, use the *copy* command, which is identical in all other respects to the *save* command. For example,

```
copy +classwork
```

copies the current message into the *classwork* folder and leaves a copy in your system mailbox.

The *folder* command can be used to direct *mailx* to the contents of a different folder. For example,

```
folder +classwork
```

directs *mail* to read the contents of the *classwork* folder. All of the commands that you can use on your system mailbox are also applicable to folders, including *type*, *delete*, and *reply*. To inquire which folder you are currently editing, type:

```
folder
```

To list your current set of folders, use the *folders* command.

To start reading one of your folders, use the *-f* option described in earlier in this section. For example,

```
% mailx -f +classwork
```

causes *mailx* to read your *classwork* folder without looking at your system mailbox.

More About Sending Mail

Tilde Escapes

While typing in a message to be sent to others, it is often useful to be able to invoke the text editor on the partial message, print the message, execute a shell command, or do some other auxiliary function. *mailx* provides these capabilities through “tilde escapes” which consist of a tilde (~) at the beginning of a line, followed by a single character indicating the function to be performed. For example, to print the text of the message so far, use:

```
~p
```

which will print a line of dashes, the recipients of your message, and the text of the message so far. Since *mailx* requires two consecutive DELs (or f12p11w8RUBOUTs) to abort a letter, you can use a single DEL to abort the output of ~p or any other ~ escape without killing your letter.

If you are dissatisfied with the message as it stands, you can invoke the text editor on it by using the escape:

```
~e
```

which causes the message to be copied into a temporary file and an instance of the editor to be spawned. After modifying the message to your satisfaction, write it out and quit the editor. *mailx* then responds by typing (or displaying):

```
(continue)
```

after which you can continue typing text to be appended to your message, or you can type **CTRL-D** to end the message. A standard text editor is provided by *mailx*.

To override the default editor, set the valued option EDITOR to specify a different shell file such as:

```
set EDITOR=/usr/bin/ex
```

or

```
set EDITOR=/usr/bin/vi
```

To use the screen or *visual* editor as an alternative to the standard text editor on your current message, you can use the escape,

```
~v
```

~v works like ~e, except that the screen editor is invoked instead. A default screen editor is defined by *mailx*. To select a different visual (screen) editor, set the valued option VISUAL to the path name of a different editor.

10 Mailx

It is sometimes useful to be able to include the contents of some file in your message. The escape

```
~r filename
```

is provided for this purpose, and causes the named file to be appended to your current message. *Mailx* complains if the file doesn't exist or can't be read. If the read is successful, the number of lines and characters appended to your message is printed, after which you can continue appending text. The filename may contain shell metacharacters like `*` and `?` which are expanded according to the conventions of your shell.

As a special case of `~r`, the escape

```
~d
```

reads in the file *dead.letter* from your home directory. This is often useful because *mailx* copies the text of your message there when you abort a message with **DEL**.

To save the current text of your message on a file, use the escape:

```
~w filename
```

Mailx then prints out the number of lines and characters written to the file, after which you can continue appending text to your message. Shell metacharacters can be used in the filename, as in `~r` and are expanded with the conventions of your shell.

If you are sending mail from within *mailx*'s command mode, you can read a message sent to you into the message you are constructing with the escape:

```
~m 4
```

which reads message 4 into the current message, shifted right by one tab stop. You can name any non-deleted message, or list of messages. To forward messages without shifting by a tab stop, use `~f` (this is the usual way to forward a message).

If, in the process of composing a message, you decide to add more people to the list of message recipients, you can do so with the escape:

```
~t <name1> <name2> ...
```

You can name as few or many additional recipients as you wish. Note that the users originally on the recipient list will still receive the message because you cannot remove someone from the recipient list with `~t`.

To associate a subject with your message, use the escape:

```
~s <arbitrary string of text>
```

which replaces any previous subject with `<arbitrary string of text>`. The subject, if given, is sent near the top of the message prefixed with `SUBJECT:`. To see what the message will look like, use `~p`.

For political reasons, one occasionally prefers to list certain people as recipients of carbon copies of a message rather than direct recipients. The escape

```
~c <name1> <name2>...
```

adds the named people to the Cc: list as when using `~t`. Again, you can execute `~p` to see what the message will look like.

The recipients of the message together constitute the To: field, the subject the Subject: field, and the carbon copies the Cc: field. If you wish to edit these in ways impossible with the `~t`, `~s`, and `~c` escapes, you can use the escape:

```
~h
```

which prints To: followed by the current list of recipients and leaves the cursor (or printhead) at the end of the line. If you type ordinary characters, they are appended to the end of the current list of recipients. You can also use your erase character to erase back into the list of recipients, or your kill character to erase them altogether. Thus, for example, if your erase and kill characters are the standard # and @ symbols,

```
~h
To: root kurt####bill
```

changes the initial recipients `root kurt` to `root bill`. When you type a newline (**RETURN** or **ENTER**), *mailx* advances to the Subject: field, where the same rules apply. Another newline brings you to the Cc: field, which can be edited in the same fashion. Another newline leaves you appending text to the end of your message. You can use `~p` to print the current text of the header fields and the body of the message.

To temporarily escape to the shell, use the sequence:

```
~!<command>
```

is used, which executes `<command>` and returns you to mailing mode without altering the text of your message. If you wish, instead, to filter the body of your message through a shell command, use:

```
~|<command>
```

which pipes your message through the command and uses the output as the new text of your message. If the command produces no output, *mailx* assumes that something is amiss and retains the old version of your message. A frequently-used filter is the command *fmt*, designed to format outgoing mail.

To effect a temporary escape to *mailx* command mode instead, use:

```
~:mailx <command>
```

This is especially useful for retyping the message you are replying to, using, for example:

```
~:t
```


12 Mailx

It is also useful for setting options and modifying aliases.

If you wish (for some reason) to send a message that contains a line beginning with a tilde, a double tilde must be used. For example,

```
~~This line begins with a tilde.
```

sends the line:

```
~This line begins with a tilde.
```

Finally, the escape

```
~?
```

prints out a brief summary of the available tilde escapes.

On some terminals (particularly those with no lower case) tildes are difficult to type. *Mailx* enables you to change the escape character by using the **escape option**. For example, to use a right bracket, type:

```
set escape=]
```

As with the tilde, if you need to send a line starting with the escape character, type a pair of adjacent escape characters as when using tilde. Redefining the escape character removes the special significance of ~.

Network Access

This section describes how to send mail to people on other machines. Recall that sending to a plain login name sends mail to that person on your machine only. If your recipient logs in on a different machine connected to yours by UUCP, You must know the list of machines through which your message must travel to arrive at his site. If his machine has a continuous (modem or direct-connect) datacomm link to yours, you can send mail to him using the syntax:

```
host!name
```

where *host* is the name of his machine and *name* is his login name. If your message must go through an intermediate machine first, you must use the syntax:

```
intermediate!host!name
```

and so on. It is actually a feature of UUCP that the map of all the systems in the network is not known anywhere (except where people decide to write it down for convenience). Talk to your system administrator about the machines connected to your site.

If you need to use an HP-UX-supported network to access recipients on other networks, contact the System Administrator of the system providing the link between networks for procedures.

When you use the *reply* command to respond to a letter, there is a problem of figuring out the names of the users in the `To:` and `Cc:` lists **relative to the current machine**. If the original letter was sent to you by someone on the local machine, then this problem does not exist, but if the message came from a remote machine, the problem must be dealt with. *mailx* uses a heuristic to build the correct name for each user relative to the local machine. So, when you *reply* to remote mail, the names in the `To:` and `Cc:` lists may change somewhat.

Special Recipients

As described previously, you can send mail to either user names or *alias* names. It is also possible to send messages directly to files or to programs, using special conventions. If a recipient name has a `/` in it or begins with a `+`, it is assumed to be the path name of a file into which to send the message. If the file already exists, the message is appended to the end of the file. If you want to name a file in your current directory (i.e., one for which a `/` would not usually be needed) you can precede the name with `./`. For example, to send mail to the file *memo* in the current directory, use the command:

```
% mailx ./memo
```

If the name begins with a `+`, it is expanded into the full path name of the folder name in your folder directory. This ability to send mail to files can be used for a variety of purposes, such as maintaining a journal and keeping a record of mail sent to a certain group of users. The second example can be done automatically by including the full pathname of the record file in the *alias* command for the group. Using our previous *alias* example, you could use the command:

```
alias project sam sally steve susan /usr/project/mail_record
```

to save all mail sent to `project` would be saved on the file `/usr/project/mail_record` as well as being sent to the members of the project. This file can be examined using `mailx -f`.

Sometimes it is useful to send mail directly to a program (such as a project billboard program). To use *mailx* to send messages to the program, use a vertical bar followed by the program file name: `|billboard`, for example.

Mailx treats recipient names that begin with a `!` as a program to send the mail to. An *alias* can be set up to reference a `!`-prefaced name if desired. **Caveats:** the *mailx* shell treats `!` specially, so it must be quoted on the command line. Also, the `! program` must be presented as a single argument to *mailx*. The safest course is to surround the entire name with double quotes. This also applies to usage in the *alias* command. For example, to alias *rmsg*s to *rmsg*s `-s` type:

```
alias rmsg " ! rmsg -s "
```

Additional Features

This section describes some additional commands of use for reading your mail, setting options, and handling lists of messages.

Message Lists

Several *mailx* commands accept a list of messages as an argument. Along with *type* and *delete*, described earlier, the *from* command prints the message headers associated with the message list passed to it. *From* is particularly useful in conjunction with some of the message list features described below.

A <message list> consists of a list of message numbers, ranges, and names, separated by spaces or tabs. Message numbers may be either decimal numbers, which directly specify messages, or one of the special characters <up arrow>, <.>, or <\$> to specify the first relevant, current, or last relevant message, respectively. For most commands, **relevant** here means **not deleted**, (or **deleted** for the *undelete* command).

A range of messages consists of two message numbers (of the form described in the previous paragraph) separated by a hyphen (dash). Thus, to print the first four messages, use:

```
type 1-4
```

and to print all the messages from the current message to the last message, use

```
type .-$
```

A <name> is a user name. The user names given in the message list are collected together and each message selected by other means is checked to make sure it was sent by one of the named users. If the message consists entirely of user names, then every message sent by one those users that is **relevant** (in the sense described earlier) is selected. Thus, to print every message sent to you by *root*, use the command:

```
type root
```

As a shorthand notation, you can specify *** to get every **relevant** (same sense) message. Thus,

```
type *
```

prints all undeleted messages,

```
delete *
```

deletes all undeleted messages, and

```
undelete *
```

undeletes all deleted messages.

You can search for the presence of a word in subject lines with */*. For example, to print the headers of all messages that contain the word *Pascal*, use the command:

```
from /Pascal
```

Note that subject searching ignores upper/lowercase differences.

List of Commands

This section describes all the *mailx* commands available when receiving mail.

- !** Used to preface a command to be executed by the shell.
- The `-` command goes to the previous message and prints it. The `-` command may be given a decimal number `<n>` as an argument, in which case the `<n>`th previous message is gone to and printed.
- Print** (abbr: *P*) Like *print*, but also print out ignored header fields. See also *print* and *ignore*.
- Reply** (abbr: *R*) Note the capital R in the name. Frame a reply to a one or more messages. The reply (or replies if you are using this on multiple messages) will be sent **ONLY** to the person who sent you the message (respectively, the set of people who sent the messages you are replying to). You can add people using the `~t` and `~c` tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with "Re:" unless it already began thus. If the original message included a "reply-to" header field, the reply will go **only** to the recipient named by "reply-to." Type in your message using the same conventions available through the *mail* command.
- The *Reply* command is especially useful for replying to messages that were sent to distribution groups when you really just want to send a message to the originator. Use it often.
- Type** (abbr: *T*) Identical to the *Print* command.
- alias** (abbr: *a*) Define a name to stand for a set of other names. This is used when you want to send messages to a certain group of people and want to avoid retyping their names. For example
- ```
alias project john sue willie kathryn
```
- creates an alias *project* which expands to the four people John, Sue, Willie, and Kathryn.
- If no argument is given, all aliases are printed; one argument prints only the alias specified.
- alternates** (abbr: *alt*) If you have accounts on several machines, you may find it convenient to use the `/usr/lib/aliases` on all the machines except one to direct your mail to a single account.
- The *alternates* command informs *mailx* that each of these other addresses is really **you**, so that when you *reply* with messages to one of these alternate names, *mailx* will not bother to send a copy of the message to this other address (which would simply be directed back to you by the alias mechanism).
- If *alternates* is given no argument, it lists the current set of alternate names. *Alternates* is usually used in the *.mailrc* file.
- chdir** (abbr: *cd*) The *chdir* command allows you to change your current directory. *Chdir* takes a single argument, which is taken to be the pathname of the directory to change to. If no argument is given, *chdir* changes to your home directory.

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>copy</i>            | (abbr: <i>co</i> ) The <i>copy</i> command is identical to <i>save</i> except that copied messages are not marked for deletion when you quit.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <i>delete</i>          | (abbr: <i>d</i> ) Deletes a list of messages. Deleted messages can be reclaimed with the <i>undelete</i> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <i>dp</i> or <i>dt</i> | <i>dpt</i> or <i>dt</i> deletes the current message and prints the next message. It is useful for quickly reading and disposing of mail. Displays <code>At EOF</code> if no messages remaining.                                                                                                                                                                                                                                                                                                                                                                     |
| <i>edit</i>            | (abbr: <i>e</i> ) The <i>edit</i> command provides editing capabilities for individual messages. It takes a list of messages as described under the <i>type</i> command and processes each by writing the message into a file <i>message</i> < <i>x</i> > (where < <i>x</i> > is the message number being edited) then executing the text editor on the file. When you have edited the message to your satisfaction, write the message out and quit the editor. <i>Mailx</i> then reads the message back and removes the file. <i>Edit can be abbreviated to e.</i> |
| <i>else</i>            | Marks the end of the <b>then</b> part of an <i>if</i> statement and the beginning of the part to take effect if the condition of the <i>if</i> statement is false.                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>endif</i>           | Marks the end of an <i>if</i> statement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>exit</i>            | (abbr: <i>ex</i> or <i>x</i> ) Leaves <i>mailx</i> without updating the system mailbox or the file you were reading. Thus, if you accidentally delete several messages, you can use <i>exit</i> to avoid scrambling your mailbox.                                                                                                                                                                                                                                                                                                                                   |
| <i>file</i>            | (abbr: <i>fi</i> ) Identical to <i>folder</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <i>folders</i>         | List the names of the folders in your folder directory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>folder</i>          | (abbr: <i>fo</i> ) The <i>folder</i> command switches to a new mail file or folder. With no arguments, it tells you which file you are currently reading. If you give it an argument, it will write out changes (such as deletions) you have made in the current file and read the new file. Some special conventions are recognized for the file/folder name:                                                                                                                                                                                                      |

| Name      | Meaning                         |
|-----------|---------------------------------|
| #         | Previous file read              |
| %         | Your system mailbox             |
| %<name>   | <i>Name</i> 's system mailbox   |
| &         | Your <code>~/mbox</code> file   |
| +<folder> | A file in your folder directory |

*from* (abbr: *f*) The *from* command takes a list of messages and prints out the header lines for each one; hence

```
from joe
```

is the easy way to display all the message headers from `joe`.

*headers* (abbr: *h*) When you start up *mailx* to read your mail, it lists the headers from each message in your mailbox. These headers tell you who each message is from, when they were sent, how many lines and characters each message is, and the `SUBJECT:` header field of each message, if present. In addition, *mailx* tags the message header of each message that has been the object of the *preserve* command with a `P`.

Messages that have been *saved* or *written* are flagged with a `*`. *Deleted* messages are not printed at all. To reprint the current list of message headers, use the *headers* command.

*Headers* (and thus the initial header listing) only lists the first so many message headers. The number of headers listed depends on the speed of your terminal. This can be overridden by specifying the number of headers you want with the `WINDOW` option. *mailx* maintains a notion of the current `WINDOW` into your messages for the purposes of printing headers.

Use the *z* command to move forward or back one window. You can move *mailx*'s notion of the current window directly to a particular message by using, for example,

```
headers 40
```

to move *mailx*'s attention to the messages around message 40. The *headers* command can be abbreviated to *h*.

*help* Print a brief (and usually out-of-date) help message about the commands in *mailx*. Refer to this manual instead.

*hold* (abbr: *ho*; also *preserve*) Arrange to hold a list of messages in the system mailbox, instead of moving them to the file *mbox* in your home directory. If you set the binary option *hold*, this will happen by default.

*if* The *if* command is used to conditionally execute commands in your *.mailrc* file, depending on whether you are sending or receiving mail. Here is an example of the general structure used:

```
if receive
 commands..
endif
```

An *else* form is also available:

```
if send
 commands..
else
 commands..
endif
```

Note that the only allowed conditions are `receive` and `send`.

- ignore* Add the list of header fields named to the `ignore list`. Header fields in the ignore list are not printed on your terminal when you print a message. This allows you to suppress printing of certain machine-generated header fields, such as `Via` which are not usually of interest. The *Type* and *Print* commands can be used to print a message in its entirety, including ignored fields. If *ignore* is executed with no arguments, it lists the current set of ignored fields.
- list* List the valid *mailx* commands.
- mailx* (abbr: *m*) Send mail to one or more people. If you have the *ask* option set, *mailx* will prompt you for a subject to your message. Type in your message, using tilde escapes described earlier to edit, print, or modify your message. To signal your satisfaction with the message and send it, type control-d at the beginning of a line, or a "." alone on a line if you set the option *dot*.
- To abort the message, type two interrupt characters (**DEL** or **RUBOUT** by default) in a row or use the `~q` escape.
- mbox* Indicate that a list of messages be sent to *mbox* in your home directory when you quit. This is the default action for messages if you do **not** have the **hold** option set.
- next* (abbr: *n*; also + or **RETURN**) The *next* command goes to the next message and types it. If given a message list, *next* goes to the first such message and types it. Thus,
- ```
next root
```
- goes to the next message sent by *root* and types it. *Next* can be abbreviated to simply a newline, which means that one can go to and type a message by simply giving its message number or one of the magic characters: `<up arrow>`, `< . >`, or `< $ >`. Thus,
- ```
. (period)
```
- prints the current message and
- ```
4
```
- prints message 4, as described previously.
- preserve* Same as *hold*. Preserves listed messages in your system mailbox when you quit.
- print* (abbr: *p*; similar to *type*) Prints all messages specified by the message list, but does not print *ignored* header fields.
- quit* (abbr: *q*) Leave *mailx* and update the file, folder, or system mailbox you were reading. Messages that you have examined are marked as `read` and messages that existed when you started are marked as `old`. If you were editing your system mailbox and the binary option `hold` is **set**, all messages which have not been deleted, saved, or mboxed will be retained in your system mailbox. If you were editing your system mailbox and `hold` is **not set**, all messages which have not been deleted, saved, or preserved are moved to the file *mbox* in your home directory.

reply (abbr: *r*) Frame a reply to the originator of a single message and send it to the originator plus all the people who received the original message, except you. You can add people using the `~t` and `~c` tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with `Re:` unless it already began thus.

If the original message included a `reply-to` header field, the reply will go **only** to the recipient named by `reply-to`. Type in your message using the same conventions as with the *mail* command.

respond Same as *reply*.

save (abbr: *s*) It is often useful to be able to save messages on related topics in a file. The *save* command gives you ability to do this. The *save* command takes as its argument a list of message numbers, followed by the name of the file on which to save the messages. The messages are appended to the named file, thus allowing one to keep several messages in the file, stored in the order they were put there. *Save* can be abbreviated *s*. Here is how *save* can be used relative to our running example:

```
s 1 2 tuitionmail
```

Saved messages are not automatically saved in *mbox* at quit time, nor are they selected by the *next* command described above, unless explicitly specified.

If the filename is preceded by a vertical bar (`|`), *mailx* treats that file as a pipe. For example,

```
s 2 | lp
```

submits message 2 to the printer.

set (abbr: *se*) Set an option or give an option a value. Used to customize *mailx*. Options are listed near the end of this article. Binary options are *on* or *off*; valued options require an accompanying `<value>` parameter. To set a binary option, type:

```
set <option>
```

For valued options:

```
set <option>=<value>
```

Several options can be specified in a single *set* command. Use *unset* to disable options.

shell (abbr: *sh*) The *shell* command enables you to escape to the shell. *Shell* invokes an interactive shell and allows you to type commands to it. When you leave the shell, return is to *mailx*. The shell used is a default assumed by *mailx* which can be overridden by setting the valued option *SHELL*. For example,

```
set SHELL=/bin/csh
```

source (abbr: *so*) The *source* command reads *mailx* commands from a file. It is useful when you are trying to fix your *.mailrc* file and you need to re-read it.

top The *top* command takes a message list and prints the first five lines of each addressed message. It can be abbreviated to *to*. If you wish, you can change the number of lines that *top* prints out by setting the valued option *toplines*. On a CRT terminal,

```
set toPlines=10
```

might be preferred.

type (abbr: *t*; similar to *print*) Print a list of messages on your terminal. If the *crt* option is **set** to a given value, and the total number of lines in the messages to be printed exceeds the *crt* value, the messages are printed by a terminal paging program such as *more*.

unalias Deletes the specified *alias(es)* from the alias list.

undelete (abbr: *u*) The *undelete* command causes a message that had been deleted previously to regain its initial status. Only messages that have been deleted can be undeleted. This command can be abbreviated to *u*.

unset Reverse the action of setting a binary or valued option.

visual (abbr: *v*) It is sometimes useful to be able to select between two editors, based on the type of terminal being used. To invoke a display-oriented editor, use the *visual* command. Except for the type of editor being used, *visual* is identical to *edit*.

Edit and *visual* commands both assume some default text editor. The default for each can be overridden by the valued options `EDITOR` and `VISUAL` for the standard and screen editors. For example, you could use:

```
set EDITOR=/usr/bin/ex VISUAL=/usr/bin/vi
```

write (abbr: *w*) The *save* command always writes the entire message, including the headers, into the file. If you want to write just the message itself, you can use the *write* command. *Write* has the same syntax as *save*, and can be abbreviated to *w*. For example, to write the second message in *file.c*, type:

```
w 2 file.c
```

As suggested by this example, *write* is useful for such tasks as sending and receiving source program text over the message system.

xit (abbr: *x*) Same as *exit*.

z *mailx* presents message headers in windowfuls as described under the *headers* command. You can move *mailx*'s attention forward to the next window by typing:

```
z+
```

command. Analogously, you can move to the previous window with:

```
z-
```

Custom Options

Throughout this article, we have seen examples of binary and valued options. This section describes each of the options in alphabetical order, including some that you have not seen yet. Options should be typed as all uppercase or all lowercase letters as listed; don't mix letter case within an option.

- EDITOR** The valued option `EDITOR` defines the pathname of the text editor to be used in the `edit` command and `~e`. If not defined, a standard default editor is used.
- SHELL** The valued option `SHELL` gives the path name of your shell. This shell is used for the `!` command and `~!` escape. In addition, this shell expands file names with shell metacharacters like `*` and `?` in them.
- VISUAL** The valued option `VISUAL` defines the pathname of your screen editor for use in the `visual` command and `~v` escape. A standard screen editor is used if you do not define one.
- append** The `append` option is binary and causes messages saved in `mbox` to be appended to the end rather than prepended. Normally, `mailx` will put messages in `mbox` in the same order that the system puts messages in your system mailbox. By setting `append`, you are requesting that `mbox` be appended to, regardless. It is in any event quicker to append.
- ask** `ASK` is a binary option which causes `mailx` to prompt you for the subject of each message you send. If you respond with simply a newline, no subject field will be sent.
- askcc** `ASKCC` is a binary option which causes you to be prompted for additional carbon copy recipients at the end of each message. Responding with a newline shows your satisfaction with the current list.
- autoprint** `AUTOPRINT` is a binary option which causes the `delete` command to behave like `dp`. Thus, after deleting a message, the next one will be typed automatically. This is useful for quickly scanning and deleting messages in your mailbox.
- debug** The binary option `debug` causes debugging information to be displayed. Use of this option is the same as using the `-d` command line flag.
- dot** `DOT` is a binary option which, if set, causes `mailx` to interpret a period alone on a line as the terminator of a message you are sending.
- escape** To change the escape character used when sending mail, use the valued option `ESCAPE`. Only the first character of the `ESCAPE` option is used, and it must be doubled if it is to appear as the first character of a line of your message. If you change your escape character, then `~` loses all its special meaning, and need no longer be doubled at the beginning of a line.
- <folder>** The name of the directory to use for storing folders of messages. If this name begins with a `"/`, `mailx` considers it to be an absolute pathname; otherwise, the folder directory is found relative to your home directory.
- hold** The binary option `hold` causes messages that have been read but not manually dealt with to be held in the system mailbox. This prevents such messages from being automatically swept into your `mbox`.

- ignore* The binary option `ignore` causes **DEL** (or **RUBOUT**) characters from your terminal to be ignored and echoed as `@`'s while you are sending mail. **DEL** characters retain their original meaning in *mailx* command mode. Setting the `ignore` option is equivalent to supplying the `-i` flag on the command line as described under Command Line Options which follows.
- ignoreeof* This option is related to `dot`, and causes *mailx* to refuse to accept a `(CTRL)-D` as the end of a message. `ignoreeof` also applies to *mailx* command mode.
- keep* The `keep` option causes *mailx* to truncate your system mailbox instead of deleting it when it is empty. This is useful if you elect to protect your mailbox, which you would do with the shell command:
- ```
chmod 600 /usr/mail/<yourname>
```
- where `<yourname>` is your login name. If you do not do this, anyone can probably read your mail, although people usually don't.
- keepsave* When you *save* a message, *mailx* usually discards it when you *quit*. To retain all saved messages, set the `keepsave` option.
- metoo* When sending mail to an alias, *mailx* makes sure that if you are included in the alias, that mail will not be sent to you. This is useful if a single alias is being used by all members of the group. If however, you wish to receive a copy of all the messages you send to the alias, you can set the binary option `metoo`.
- noheader* The binary option `noheader` suppresses the printing of the version and headers when *mailx* is first invoked. Setting this option is the same as using `-N` on the command line.
- nosave* Normally, when you abort a message with two **DELS** (or **RUBOUTS**), *mailx* copies the partial letter to the file *dead.letter* in your home directory. Setting the binary option `nosave` prevents this.
- quiet* The binary option `quiet` suppresses the printing of the version when *mailx* is first invoked, as well as printing the *type* command message number with each message.
- record* If you love to keep records, then the valued option `record` can be set to the name of a file to save your outgoing mail. Each new message you send is appended to the end of the file.
- screen* When *mailx* initially prints the message headers, it determines how many headers to print by looking at the speed of your terminal; the faster your terminal, the more it prints. The valued option `screen` overrides this calculation and specifies how many message headers you want printed. This number is also used for scrolling with the `z` command.
- sendmail* To select an alternate delivery system, set the `sendmail` option to the full pathname of the program to use. **Note: this is not for everyone! Most people should use the default delivery system.**
- toplines* The valued option `toplines` defines the number of lines that the *top* command will print out instead of the default five lines.
- verbose* The binary option "verbose" causes *mailx* to invoke *sendmail* with the `-v` flag, which causes it to go into verbose mode and announce expansion of aliases, etc. Setting the "verbose" option is equivalent to invoking *mailx* with the `-v` flag as described earlier.

## Command Line Options

This section describes command line options for *mailx* and what they are used for.

- N* Suppress the initial printing of headers.
  - d* Turn on debugging information. Not of general interest.
  - f* <file> Show the messages in <file> instead of your system mailbox. If <file> is omitted, *mailx* reads *mbox* in your home directory.
  - i* Ignore tty interrupt signals. Useful on noisy phone lines, which generate spurious RUBOUT or DELETE characters. It's usually more effective to change your interrupt character to CTRL-C (see the *stty* shell command for more information).
  - n* Inhibit reading of */usr/lib/Mail.rc*. Not generally useful, since */usr/lib/Mail.rc* is usually empty.
  - s* <string> Used for sending mail. <String> is used as the subject of the message being composed. If <string> contains blanks, you must surround the string with quote marks.
  - u* <name> Read <name>'s mail instead of your own. Other unwitting systems users often neglect to protect their mailboxes, but discretion is advised. Essentially, `-u kathy` is a shorthand way of doing `-f /usr/mail/kathy`.
  - v* Use the `–v` flag when invoking *sendmail*. This feature can also be enabled by setting the the option “verbose”.
- The following command line flags are also recognized, but are intended for use by programs invoking *mailx* and not for people.
- T* <file>. Arrange to print on <file> the contents of the `article-id` fields of all messages that were either read or deleted. `–T` is for the *readnews* program and should NOT be used for reading your mail.
  - h* <number> Pass on hop count information. *Mailx* takes <number>, increments it, and passes it with `–h` to the mail delivery system. `–h` has effect only when sending mail and is used for network mail forwarding.
  - r* <name> Used for network mail forwarding where <name> is the sender of the message. <name> and `–r` are simply sent along to the mail delivery system. *Mailx* waits for the message to be sent and the exit status returned. Also restricts formatting of message.

Note that `–h` and `–r` (which are for network mail forwarding) are not used in practice since mail forwarding is now handled separately. They may disappear in future HP-UX versions.

## Message Format

This section describes message formats. Messages begin with a `From` line, which consists of the word `From` followed by a user name, followed by anything, followed by a date in the format returned by the `ctime` library routine described in section 3 of the HP-UX Reference manual. A possible `ctime` format date is:

```
Tue Dec 1 10:58:23 1981
```

The `ctime` date may be optionally followed by a single space and a time zone indication, which should be three capital letters, such as MDT.

Following the `From` line are zero or more `header field` lines. Each header field line is of the form:

```
name: information
```

`Name` can be anything, but only certain header fields are recognized as having any meaning. The recognized header fields are: `article-id`, `bcc`, `cc`, `from`, `reply-to`, `sender`, `subject`, and `to`.

Other header fields may be significant to various networks. Refer to the message standards documentation for the network being used for more information. A header field can be continued onto following lines by making the first character on the following line a space or tab character.

If any headers are present, they must be followed by a blank line. The part that follows is called the `body` of the message, and must be ASCII text containing no null characters. Each line in the message body must be terminated with an ASCII newline character and no line can be longer than 512 characters. If binary data must be passed through the mail system, it is suggested that this data be encoded in a format that encodes six bits into a printable character.

For example, one could use the upper- and lowercase letters, the digits, comma and period to make up a set of 64 characters. Thus, a 16-bit binary number could be sent as three characters. These characters should be packed into lines, preferably lines about 70 characters long because long lines are transmitted more efficiently.

The message delivery system always adds a blank line to the end of each message. This blank line must not be deleted.

The UUCP message delivery system sometimes adds a blank line to the end of a message each time it is forwarded through a machine.

Note that some network transport protocols enforce message length limits.

## Glossary

This section contains the definitions of a few phrases peculiar to *mailx*.

|                     |                                                                                                                                                                                                         |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>alias</i>        | An alternative name for a person or list of people.                                                                                                                                                     |
| <i>flag</i>         | An option, given on the command line of <i>mailx</i> , prefaced with a <code>-</code> . For example, <code>-f</code> is a flag.                                                                         |
| <i>header field</i> | At the beginning of a message, a line containing information that is part of the structure of the message. Popular header fields include <code>to</code> , <code>cc</code> , and <code>subject</code> . |
| <i>mail</i>         | A collection of messages. Often used as in the phrase, “Have you read your mail?”                                                                                                                       |
| <i>mailbox</i>      | The place where your mail is stored, typically in the directory <code>/usr/mail</code> .                                                                                                                |
| <i>message</i>      | A single letter from someone, initially stored in your <i>mailbox</i> .                                                                                                                                 |
| <i>message list</i> | A string used in <i>mailx</i> command mode to describe a sequence of messages.                                                                                                                          |
| <i>option</i>       | A piece of special purpose information used to tailor <i>mailx</i> to your taste. Options are specified with the <i>set</i> command.                                                                    |

## Summary of Commands, Options, and Escapes

### Command Summary

The following tables provide a quick summary of the *mailx* commands, binary and valued options, and tilde escapes. Command abbreviations, where applicable, are shown in bold type in parentheses at the beginning of the description.

| Command                | Description                                                                            |
|------------------------|----------------------------------------------------------------------------------------|
| <i>!</i>               | Single command escape to shell                                                         |
| <i>-</i>               | Back up to previous message                                                            |
| <i>Print</i>           | ( <b>P</b> ) Type message with ignored fields                                          |
| <i>Reply</i>           | ( <b>R</b> ) Reply to author of message only                                           |
| <i>Type</i>            | ( <b>T</b> ) Type message with ignored fields                                          |
| <i>alias</i>           | ( <b>a</b> ) Define an alias as a set of user names                                    |
| <i>alternates</i>      | ( <b>alt</b> ) List other names you are known by                                       |
| <i>chdir</i>           | ( <b>cd</b> ) Change working directory, home by default                                |
| <i>copy</i>            | ( <b>co</b> ) Copy a message to a file or folder                                       |
| <i>delete</i>          | ( <b>d</b> ) Delete a list of messages                                                 |
| <i>dp</i> or <i>dt</i> | Delete current message, type next message                                              |
| <i>endif</i>           | End of conditional statement; see <i>if</i>                                            |
| <i>edit</i>            | ( <b>e</b> ) Edit a list of messages                                                   |
| <i>else</i>            | Start of else part of conditional; see <i>if</i>                                       |
| <i>exit</i>            | ( <b>ex</b> or <b>x</b> ) Leave mail without changing anything                         |
| <i>file</i>            | ( <b>fi</b> ) Interrogate/change current mail file                                     |
| <i>folder</i>          | ( <b>fo</b> ) Same as <i>file</i>                                                      |
| <i>folders</i>         | List the folders in your folder directory                                              |
| <i>from</i>            | ( <b>f</b> ) List headers of a list of messages                                        |
| <i>headers</i>         | ( <b>h</b> ) List current window of messages                                           |
| <i>help</i>            | Print brief summary of <i>mailx</i> commands                                           |
| <i>hold</i>            | ( <b>ho</b> ) Same as <i>preserve</i>                                                  |
| <i>if</i>              | Conditional execution of <i>mailx</i> commands                                         |
| <i>ignore</i>          | Set/examine list of ignored header fields                                              |
| <i>list</i>            | List valid <i>mailx</i> commands                                                       |
| <i>local</i>           | List other names for the local host                                                    |
| <i>mailx</i>           | ( <b>m</b> ) Send mail to specified names                                              |
| <i>mbox</i>            | Arrange to save a list of messages in <i>mbox</i>                                      |
| <i>next</i>            | ( <b>n</b> , <b>+</b> , or <b>RETURN</b> ) Go to next message and type it              |
| <i>preserve</i>        | Arrange to leave list of messages in system mailbox                                    |
| <i>print</i>           | Print specified messages without <i>ignored</i> headers                                |
| <i>quit</i>            | ( <b>q</b> ) Leave <i>mailx</i> ; update system mailbox and <i>mbox</i> as appropriate |
| <i>reply</i>           | ( <b>r</b> ) Compose a reply to a message                                              |

| Command         | Description                                                  |
|-----------------|--------------------------------------------------------------|
| <i>save</i>     | (s) Append messages, headers included, on a file             |
| <i>set</i>      | (se) Set binary or valued options                            |
| <i>shell</i>    | (sh) Invoke an interactive shell                             |
| <i>source</i>   | (so) Reads <i>mailx</i> commands from a file.                |
| <i>top</i>      | Print first so many (5 by default) lines of list of messages |
| <i>type</i>     | (t) Print messages                                           |
| <i>unalias</i>  | Remove one or more <i>alias</i> groups                       |
| <i>undelete</i> | (u) Undelete list of messages                                |
| <i>unset</i>    | Undo the operation of a <i>set</i>                           |
| <i>visual</i>   | (v) Invoke visual editor on a list of messages               |
| <i>write</i>    | (w) Append messages to a file, don't include headers         |
| <i>xit</i>      | (x) Synonym for <i>exit</i>                                  |
| <i>z</i>        | Scroll to next/previous screenful of headers                 |

## Options Summary

The following table describes the options. Each option is shown as being either a binary or valued option.

| Option    | Type   | Description                                                   |
|-----------|--------|---------------------------------------------------------------|
| EDITOR    | valued | Pathname of editor for <i>~e</i> and <i>edit</i>              |
| SHELL     | valued | Pathname of shell for <i>shell</i> , <i>~!</i> , and <i>!</i> |
| VISUAL    | valued | Pathname of screen editor for <i>~v</i> and <i>visual</i>     |
| append    | binary | Always append messages to end of <i>mbox</i>                  |
| ask       | binary | Prompt user for Subject: field when sending                   |
| askcc     | binary | Prompt user for additional Cc's at end of message             |
| autoPrint | binary | Print next message after <i>delete</i>                        |
| crt       | valued | Minimum number of lines before using <i>more</i>              |
| debug     | binary | Print out debugging information                               |
| dot       | binary | Accept . alone on line to terminate message input             |
| escape    | valued | Escape character to be used instead of <i>~</i>               |
| folder    | valued | Directory to store folders in                                 |
| hold      | binary | Hold messages in system mailbox by default                    |
| ignore    | binary | Ignore <b>DEL</b> (or <b>RUBOUT</b> ) while sending mail      |
| ignoreeof | binary | Don't terminate letters/command input with EOF                |
| keep      | binary | Don't unlink system mailbox when empty                        |
| keepsave  | binary | Don't delete <i>saved</i> messages by default                 |
| metoo     | binary | Include sending user in aliases                               |
| noheader  | binary | Suppress initial printing of version and headers              |
| nosave    | binary | Don't save partial letter in <i>dead.letter</i>               |
| quiet     | binary | Suppress printing of <i>mailx</i> version and message numbers |
| record    | valued | File to save all outgoing mail in                             |
| screen    | valued | Size of window of message headers for <i>z</i> , etc.         |
| sendmail  | valued | Choose alternate mail delivery system                         |
| toplines  | valued | Number of lines to print in <i>top</i>                        |
| verbose   | binary | Invoke sendmail with the <i>-v</i> flag                       |



## Tilde Escapes Summary

The following table summarizes the tilde escapes available while sending mail.

| Escape | Arguments | Description                                                |
|--------|-----------|------------------------------------------------------------|
| ~/     | command   | Execute shell command                                      |
| ~c     | name ...  | Add names to Cc: field                                     |
| ~d     |           | Read <i>dead.letter</i> into message                       |
| ~e     |           | Invoke text editor on partial message                      |
| ~f     | messages  | Read named messages                                        |
| ~h     |           | Edit the header fields                                     |
| ~m     | messages  | Read named messages, right shift by tab                    |
| ~p     |           | Print message entered so far                               |
| ~q     |           | Abort entry of letter; like <b>DEL</b> (or <b>RUBOUT</b> ) |
| ~r     | filename  | Read file into message                                     |
| ~s     | string    | Set Subject: field to <i>string</i>                        |
| ~t     | name ...  | Add names to To: field                                     |
| ~v     |           | Invoke screen editor on message                            |
| ~w     | filename  | Write message on file                                      |
| ~      | command   | Pipe message through <i>command</i>                        |
| ~~     | string    | Quote a ~ in front of <i>string</i>                        |

## Command Line Flags

The following table shows the command line flags that *mailx* accepts:

| Flag        | Description                                     |
|-------------|-------------------------------------------------|
| -N          | Suppress the initial printing of headers        |
| -T <file>   | Article-id's of read/deleted messages to <file> |
| -d          | Turn on debugging                               |
| -f <file>   | Show messages in <file> or <~/mbox>             |
| -h <number> | Pass on hop count for mail forwarding           |
| -i          | Ignore tty interrupt signals                    |
| -n          | Inhibit reading of <i>/usr/lib/Mail.rc</i>      |
| -r <name>   | Pass on <name> for mail forwarding              |
| -s <string> | Use <string> as subject in outgoing mail        |
| -u <name>   | Read <name's> mail instead of your own          |
| -v          | Invoke sendmail with the -v flag                |

Note that -T, -d, -h, and -r are not for human use.

